Insider Threat Detection Using Neural Networks:
A Statistical Comparison of the CERT Insider Threat Database

by Wendell H. Walter

B.S. Telecommunications and Electronics Engineering, July 2007, University of Oriente
M.S. in Communication Technology and Policy, August 2009, Ohio University
MBA in Business Administration and Business Intelligence and Analytics Management,
May 2018, Keller Graduate School of Management of DeVry 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

May 18, 2025

Praxis directed by

Evan Wehr
Professorial Lecturer of Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that Wendell H Walter has passed the Final Examination for the degree of Doctor of Engineering as of March 26, 2025. This is the final and approved form of the Praxis.

Insider Threat Detection Using Neural Networks:
A Statistical Comparison of the CERT Insider Threat Database

Wendell H Walter

Praxis Research Committee:

Evan Wehr, Professorial Lecturer in Engineering and Applied Science, Praxis Director

Amir Etemadi, Associate Professor of Engineering and Applied Science, Committee Member

John Morton, Professorial Professor of Engineering and Applied Science, Committee Member

© Copyright 2025 by Wendell Walter
All rights reserved

Dedication

The author wishes to dedicate this work to his late father, grandmother, his living mother, brothers, sisters, wife and kids.

Acknowledgements

Several thanks are given to the following individuals:

1. Dr. David A. Noever for making his modified dataset available for conducting additional research.
2. My advisor Evan Wehr for all the hours of undivided attention and constant reminders.
3. My wife Gizelle Walter, for her encouragement, my daughter Gabrielle and my son Evan, for their patience when I couldn't join them in all the fun activities.
4. Michelle Seitz, for applying all the formatting and finishing touches to this work.
5. The Knuth Agency, for providing a comprehensive content review.

Abstract of Praxis

Insider Threat Detection Using Neural Networks: A Statistical Comparison of the CERT Insider Threat Database

This research compares the efficacy of various classifier TensorFlow Neural Networks (NNs) at detecting Insider Threats to guide designers of security solutions on the selection of appropriate NN algorithms for Insider Threat detection. The praxis utilized a modified version of the CERT r4.2 dataset provided by David A. Noever. Fifteen TensorFlow NN models were compared, as were different rescaling techniques and epoch counts.

A Kruskal-Wallis test and Dunn's post hoc test were used to compare NN models, which revealed differences in performance. The top-performing model on average was non-rescaled ELM-klm 10, which achieved an accuracy of 0.9804. That is in contrast with the top performer of Noever's research, Random Forest, which achieved an accuracy of 0.9835. It should also be noted that this praxis utilized 10 risk indicators to achieve this performance by saving processing time with only a small loss of accuracy.

Furthermore, there was a significant difference across rescaling techniques such as Standardize, Normalized, Non-Rescaled or Robust, as was also the case for the utilization of 10 Epoch counts or 35 Epoch counts. While these findings align with previous literature in terms of showcasing different levels of accuracy across models using various techniques, the specific outcomes differ when compared to other literature, with accuracy differences being small. Future research could expand on this and similar studies by testing other datasets, as well as evaluating additional NN models and platforms.

Declarations

1. The dataset used in this praxis was originally obtained from CERT and modified by the author (Noever, 2019).
2. All figures, tables and graphics obtained from other sources are cited based on the original creators.
3. There are no conflicts of interest to be declared by the author.

Table of Contents

Dedication ..... **iv**

Acknowledgements ..... **v**

Abstract of Praxis ..... **vi**

Declarations ..... **vii**

Table of Contents ..... **viii**

List of Figures ..... **xii**

List of Tables ..... **xiv**

List of Symbols ..... **xv**

List of Acronyms ..... **xvi**

List of Equations ..... **xix**

Chapter 1: Introduction **1**

1.1 Background 1

1.2 Research Motivation 2

1.3 Problem Statement 3

1.4 Thesis Statement 4

1.5 Research Objectives 4

1.6 Research Questions and Hypotheses 5

1.6.1 Research Questions 5

1.6.2 Hypotheses 5

1.7 Scope of Research 6

1.8 Research Limitations 7

1.9 Organization of Praxis 7

Chapter 2: Literature Review 9

2.1 Introduction 9

2.2 Machine Learning 9

2.3 Machine Learning Methods 10

2.3.1 Neural Networks 11

2.4 Insider Threat Datasets 13

2.4.1 The Wolf Of STUD (TWOS) Dataset 14

2.4.2 ENRON Dataset 14

2.4.3 APEX '07' Dataset 15

2.4.4 DARP 1998 Dataset 15

2.4.5 NSL-KDD Dataset 15

2.4.6 CERT Dataset 16

2.4.7 Dr. Noever's Dataset 16

2.5 Types of Insiders 17

2.5.1 Insider Threat Classification 18

2.6 Research Related to Insider Threat Detection and Model Performance. 19

2.7 Research Related to TensorFlow 24

Chapter 3: Methodology 26

3.1 Introduction 26

3.2 Dataset Preprocessing and Constructions in Dr. Noevers' Research 27

3.2.1 Feature Selection Using Random Forest 29

3.2.2 Use of SMOTE During Feature Selection 32

3.3 Model Generation, TensorFlow, Rescaling, and Performance Evaluation 32

3.3.1 TensorFlow NN Learners 32
3.3.1.1 Radial Basis Function (RBF) 33
3.3.1.2 Radial Basis Function with Dynamic Decay Adjustment (RBF-DDA).....34
3.3.1.3 Radial Basis Function Network (RBFN).....35
3.3.1.4 Determinantal Point Process (DPP) 35
3.3.1.5 Extreme Learning Machine (ELM).....36
3.3.1.6 Extreme Learning Machine with Gaussian Kernel (ELM-klm).....36
3.3.1.7 Neural Network (NN).....37
3.3.1.8 Averaged Neural Network (ANN) 38
3.3.1.9 Principal Component Neural Network PCNN 39
3.3.1.10 Multi-Layer Perceptron (MLP) 40
3.3.1.11 Multi-layer Perceptron with Weight Decay (MLP-wd) 41
3.3.1.12 Direct Kernel Perceptron (DKP) 41
3.3.1.13 Learning Vector Quantization (LVQ) 42
3.3.1.14 Cascade Correlation (CASCOR).....43
3.3.1.15 Bi-directional Kohonen (BDK).....43
3.3.2 Data Scaling Techniques.....44
3.3.3 Performance Model Metrics 45
3.4 One-Way Analysis of Variance (ANOVA).....47
3.4.1 Assumptions for One-Way Analysis of Variance.....48
3.5 Kruskal-Wallis and Dunns Test.....53
3.6 Experiment Design 54

Chapter 4: Results **55**

4.1 Introduction 55

4.2 Model Speed by Rescaling Technique 55

4.3 Model Accuracy by Rescaling Technique 58

4.4 Kruskal-Wallis Analysis 64

4.4.1 Dunn's Test Lettered Groupings 67

4.5 Addressing Research Questions and Hypotheses Outcomes 70

4.5.1 Research Question and Hypothesis, Set 1 71

4.5.2 Research Question and Hypothesis, Set 2 71

4.5.3 Research Question and Hypothesis, Set 3 72

4.5.4 Research Question and Hypothesis, Set 4 73

Chapter 5: Discussion and Conclusions **75**

5.1 Discussion 75

5.2 Conclusions 75

5.3 Contributions to Body of Knowledge 76

5.4 Recommendations for Future Research 77

References **79**

Appendix A **97**

List of Figures

Figure 1: Neural Network Architecture 12

Figure 2: Sigmoid Activation Function Plot 13

Figure 3: Random Forest Feature Selection Rank Order of Importance 31

Figure 4: Top 10 Features by Rank Order of Importance 31

Figure 5: Levene's Test Demonstrating Lack of Homogeneity of Variance 51

Figure 6: Standardized Accuracy Bar Plot 60

Figure 7: Normalized Accuracy Bar Plot 61

Figure 8: Non-Rescaled Accuracy Bar Plot 62

Figure 9: Robust Accuracy Comparison by Model and Epoch 63

Figure 10: Kruskal-Wallis Significance 64

Figure 11: Dunn's Test Lettered Groupings, One of Three 68

Figure 12: Dunn's Test Lettered Groupings, Two of Three 69

Figure 13: Dunn's Test Lettered Groupings, Three of Three 70

Figure 14: Correlation Matrix and Variance for the Dataset Feature Section 97

Figure 15: JB P-Value Test Results Summary by Learner and Scaler 99

Figure 16: AUC Comparison for Standardized- 10 and 35 Epochs 102

Figure 17: F1 Score Comparison for Standardized- 10 and 35 Epochs 103

Figure 18: Precision Comparison for Standardized- 10 and 35 Epochs 103

Figure 19: Recall Comparison for Standardized- 10 and 35 Epochs 104

Figure 20: AUC Comparison for Normalized- 10 and 35 Epochs 104

Figure 21: F1 Score Comparison for Normalized- 10 and 35 Epochs 105

Figure 22: Precision Comparison for Normalized- 10 and 35 Epochs 105

Figure 23: Recall Comparison for Normalized- 10 and 35 Epochs 106

Figure 24: AUC Comparison for Robust- 10 and 35 Epochs 106
Figure 25: F1 Score Comparison for Robust- 10 and 35 Epochs 107
Figure 26: Precision Comparison for Robust- 10 and 35 Epochs..... 107
Figure 27: Recall Comparison for Robust- 10 and 35 Epochs 108
Figure 28: AUC Comparison for Non-Rescaled- 10 and 35 Epochs..... 108
Figure 29: F1 Score Comparison for Non-Rescaled- 10 and 35 Epochs..... 109
Figure 30: Precision Comparison for Non-Rescaled- 10 and 35 Epochs 109
Figure 31: Recall Comparison for Non-Rescaled- 10 and 35 Epochs..... 110
Figure 32: Average Speed by Model 111

List of Tables

Table 1: Related Research on Insider Threats Model Performance 19

Table 2: Risk Indicators Extracted from CERT Dataset Features 28

Table 3: JB P-Value Test Results- Learner values less than 0.05 52

Table 4: Standardized Speed Rank Order by Epoch 10 and 35 56

Table 5: Normalized Speed Rank Order by Epoch 10 and 35 56

Table 6: Non-Rescaled Speed Rank Order by Epoch 10 and 35 57

Table 7: Robust Speed Rank Order by Epoch 10 and 35 58

Table 8: Standardized Accuracy Rank Order 59

Table 9: Normalized Accuracy Rank Order 60

Table 10: Non-Rescaled Accuracy Rank Order 61

Table 11: Robust Accuracy Rank Order 63

Table 12: Kruskal-Wallis Median Rank Order 64

Table 13: Top 7 Over-sampled Result by Model Accuracy 72

Table 14: JB P-Value Test Results- Learner Values More Than or Equal to 0.05 97

Table 15: Top 7 Under-sampled Result by Model Accuracy 99

Table 16: TensorFlow Neural Network Performance Metrics by Rescale Type 99

Table 17: Average Speed Rank by Model 110

Table 18: Praxis Top Performer comparison with Dr. Noever Top Seven Performers Rank Order by Accuracy 112

List of Symbols

α	Probability Value
σ^2	Variance
N	Number of observations
i	Index
$H_{i\text{-}n}$	Null Hypothesis
$H_{i\text{-}a}$	Alternate Hypothesis
( )	Open and close parentheses
[ ]	Open and close bracket
“ ”	Open and close quotation
=	Equal sign
N	Number of samples within a category used in hypothesis testing

List of Acronyms

AdaBoost	Adaptive Boosting
AD-DNN	Anomaly Detection Deep Neural Networks
AE	Autoencoders
AI	Artificial Intelligence
ANN	Averaged Neural Network
ANOVA	Analysis of Variance
AUC	Area Under the Curve
BDK	Bi-directional Kohonen
CASCOR	Cascade Correlation
CERT	Computer Emergency Response Team
CM	Confusion Matrix
CNN	Convolutional Neural Networks
DBLOF	Density-Based Local Outlier Factor
DBN	Deep Belief Network
DKP	Direct Kernel Perceptron
DMNAED	Dynamic Memory Network for Abnormal Event Detection
DPP	Determinantal Point Process
DT	Decision Tree
ELM	Extreme Learning Machine
ELM-KLM	Extreme Learning Machine with Gaussian Kernel
ET	Extra Tree
F1	f1-score

FN	False Negative
FP	False Positive
GB	Gradient Boosting
GE	General Electric
GPU	Graphic Processing Unit
H	Hypotheses
HMM	Hidden Markov Model
IBM	International Business Machine Corporation
IF	Isolation Forest
IoT	Internet of Things
KNN	K-Nearest Neighbors
KW	Kruskal-Wallis
LOF	Local Outlier Factor
LOR	Logic Regression
LR	Linear Regression
LSTM	Long Short-Term Memory
LVQ	Learning Vector Quantization
ML	Machine Learning
MLP	Multi-Layer Perceptron
MLP-WD	Multi-Layer Perceptron Weight Decay
NB	Naïve Bayes
NN	Neural Network
OCSVM	One-Class Support Vector Machine

PCA	Principal Component Analysis
PCNN	Principal Component Analysis Network
RBF	Radial Basis Function
RBF-DDA	Radial Basis Function- Dynamic Decay Adjustment
RBFN	Radial Basis Function Network
ReLU	Rectified Liner Unit
RF	Random Forest
RNN	Recurrent Neural Network
RO	Research Objective
ROC	Receiver Operating Characteristics
RQ	Research Question
SMOTE	Synthetic Minority Oversampling Technique
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TF-IDF	Term Frequency Inverse Document Frequency
TN	True Negative
TP	True Positive
XGB	eXtreme Gradient-Boosting

List of Equations

Equation 1: Sigmoid Equation 1 13

Equation 2: Accuracy Equation 2 46

Equation 3: Precision Equation 3 46

Equation 4: Recall Equation 4 46

Equation 5: f1 Score Equation 5 46

Chapter 1: Introduction

1.1 Background

Insiders pose a significant threat to security systems of government organizations, businesses, and institutions due to their knowledge and privileged access to information (Alzaabi & Mehmood, 2024). The CERT Guide to Insider Threats, which sets the background for this research, defines Malicious Insiders as:

> *"...a current or former employee, contractor, or business partner who has or had authorized access to an organization's network, system, or data, and who intentionally exceeded or misused that access in a manner that negatively affects the confidentiality, integrity, or availability of the organization's information or information systems" (Cappelli et al., 2012 p. 354).*

According to the CrowdStrike 2024 Threat Hunting Report, in April 2024, the CrowdStrike services responded to several Insider incidents targeted at more than 30 U.S.-based companies including aerospace, defense, retail and technology organizations. The report indicated that the Insiders were U.S residents hired for remote IT positions in 2023. This underlines the Insider Threat activity associated with remote workers access to internal resources (CrowdStrike, 2024).

According to a benchmark study published by the Ponemon Institute, in which a total of 1,075 IT and IT security practitioners from 309 organizations including North America, Europe, Middle East, Africa and Asia-Pacific were interviewed, a total of 7,343 Insider incidents were recorded with Malicious Insiders accounting for an average of 6.2 incidents annually across these regions at an average cost per incident of $701,500 (Ponemon Institute, 2023). Companies spent the majority of their incident response budget on the containment of Insider. Incidents taking less than 30 days to contain had an

average cost of $11.92 million in contrast to the average cost of $18.33 million for incidents taking more than 90 days to contain. The total average cost of activities to resolve Insider Threats over a 12-month period was $16.2 million for all events. North American companies spent more than the average cost at $19.09 million, followed by European companies at $17.47 million (Ponemon Institute, 2023).

1.2 Research Motivation

The Insider Threat research in this paper will examine evolving methods and motivations of malicious Insiders, given the increasing proliferation and ease-of-access to sensitive data. Several factors are driving this proliferation. The growth in big data analytics, cloud computing, and Internet of Things (IoT) has expanded the threat landscape needing protection from Insider Threats (Rahmani et al., 2022). These technologies generate more data.

In addition, the recent shift to remote and hybrid work environments, accelerated by the COVID-19 pandemic, has introduced challenges for Insider Threat detection (Rahmani et al., 2022). Employees can now access more digital resources than ever before, and from various locations, increasing the need for improved detection and classification of Insider activities. The improved classification of Insider activity with Machine Learning (ML) has excelled in the identification of changes in user behavior, which alerts security professionals to malicious activity. ML models for Insider Threat detection and classification offer a path to achieve these goals (Rahmani et al., 2022). The 2024 CrowdStrike Threat Hunting Report also underscores the presence of Insider incidents for over 30 companies across various industries which were linked to remote workers having access to company resources (CrowdStrike, 2024).

According to the June 2023 report on Insider Threat Indicators in User Activity Monitoring, by the Defense Counterintelligence and Security Agency, security and log capabilities have created a challenge due to data collections overloads. Security market vendors have responded by providing more security tool decisions (Defense Counterintelligence and Security Agency, 2023). However, this has presented a problem in terms of complexity and large-scale data analysis. To overcome these challenges, there is a need to identify exactly what information needs to be isolated from the large volumes of data to make quick and accurate decisions (Defense Counterintelligence and Security Agency, 2023).

1.3 Problem Statement

Insider Threats have traditionally been addressed using situational awareness, log monitoring, and observations of off-policy employee behavior; however, controls such as this have proved insufficient on their own for detecting Insider Threats (Defense Counterintelligence and Security Agency, 2023). However, digital tracking of user behaviors is increasingly used to flag unusual activities. The amount of data gathered on users may become overwhelming for security experts, and without the right detection technologies in place, may lead to an increase in insider behavior that could go unnoticed for an extended period (Defense Counterintelligence and Security Agency, 2023).

In addition, the 2024 Insider Threat report from Gurucul states that a recent survey of 413 IT and cybersecurity professionals in North America indicated 55% of respondents believe that Insider Attacks are as difficult to detect as external cyber-attacks compared to the 48% previously reported in 2023 (Gurucul, 2024). The report shows that 37% of respondents believe that Insider attacks are more difficult to detect than external

cyber-attacks, which is an improvement from the 44% previously reported in 2023.This is still a current concern and an organizational problem that should be addressed with better detection and monitoring tools that are focused on user activity (Gurucul, 2024).

1.4 Thesis Statement

An ML solution can be deployed to detect Insider Threats by shrinking the gap between the collection of data and timely, accurate decision making. Security professionals employ log monitoring tools to collect and track user activities. However, these tools fail to accurately classify data streams by correlating events, which would lead to more relevant alerts and timelier decisions related to the classification of Insider Threat (Defense Counterintelligence and Security Agency, 2023).

According to the 2024 Gurucul Insider Threat report, the faster an organization can recover from an Insider attack, the lower the operational impact and financial loss due to business continuity (Gurucul, 2024). The report reveals 55% of organizations report recovering from Insider attacks within a day, while 45% took a week or longer to recover. To reduce this recovery time, it is important for organizations to leverage security solutions that integrate ML, risk-based prioritization and contextual analysis to guide security teams to quickly identify and classify Insider events (Gurucul, 2024).

1.5 Research Objectives

This praxis is focused on the following research objectives (ROs):

RO$_1$: Acquire a dataset suitable for Insider Threat detection using Neural Networks (NNs).

RO_2: Produce insight into the differences in performance among the 15 types of NNs created to detect Insider Threats.

RO_3: Produce insight to guide security professionals designing solutions on the selection of appropriate NN algorithms for Insider Threat detection.

1.6 Research Questions and Hypotheses

The following Research Questions (RQs) will guide this praxis. The associated Hypotheses (H) below were identified as key elements for evaluation in this praxis:

1.6.1 Research Questions

RQ_1: Use TensorFlow to detect if there are differences in performance between the NN models created to detect Insider Threats?

RQ_2: Use TensorFlow Network to identify if the tested NN models are better at detecting Insider Threats than the top seven best-performing models from *Classifier Suites for Insider Threat Detection* (Noever, 2019).

RQ_3: Are there significant differences between data rescaling techniques used in NN models for Insider Threat detection?

RQ_4: Are there significant differences between different NN model's epoch counts used for Insider Threat detection?

1.6.2 Hypotheses

$H_{1\text{-}n}$: Of NN models created to detect Insider Threats, there is no significant difference between their performance.

$H_{1\text{-}a}$: Of NN models created to detect Insider Threats, there is at least one model with a significant difference between their performance as determined by a p-value threshold.

$H_{2\text{-}n}$: Of NN models created to detect Insider Threats, no models of this praxis performed better than the top seven models from *Classifier Suites for Insider Threat Detection* (Noever, 2019).

$H_{2\text{-}a}$ Of NN models created to detect Insider Threats, at least one model of this praxis performed better than the top seven models from *Classifier Suites for Insider Threat Detection* (Noever, 2019).

$H_{3\text{-}n}$: Of NN models created to detect Insider Threats, there is no significant difference among different data rescaling techniques.

$H_{3\text{-}a}$: Of NN models created to detect Insider Threats, there is a significant difference among different data rescaling techniques.

$H_{4\text{-}n}$: Of NN models created to detect Insider Threats, use of epochs makes no significant difference in learner performance.

$H_{4\text{-}a}$: Of NN models created to detect Insider Threats, use of epochs makes a significant difference in at least one learner's epoch performance.

1.7 Scope of Research

This praxis focuses on NN-based classification models for malicious Insider Threat detection using TensorFlow. Non-classification-based models will not be used as they do not contribute to the overall goal of identifying and classifying Insider Threat activity. TensorFlow is introduced in Section 2.7 and a total of 15 Neural Network learners is summarized in Section 3.3.1. A fundamental assumption of this research

includes the use of a modified version of the publicly available CERT r4.2 dataset, discussed in greater detail in Section 3.1.

1.8 Research Limitations

There is a dearth of available data for researchers to study the Insider Threats problem, as "organizations are often reluctant to report incidents of illicit Insider cyberactivity, even to law enforcement" (Cappelli et al., 2012, p. 20). Organizations are challenged to adhere to privacy standards and regulations, and due to the kinds of sensitive data that are a part of Insider Threat incidents, the ability to report summaries of Insider Threat incidents is limited to certain (non-public) audiences. That is particularly true in the case of research related to national security. As a result of these constraints, access to Insider Threat datasets is limited (Cappelli et al., 2012).

The research phase of this praxis was conducted within a year, which limited the number of options for creating or acquiring Insider Threat datasets suitable for ML. Real life considerations such as human resource capital and budget constraints associated with building a network environment with various Insider Threat events will therefore likely deviate from the data used within this praxis.

1.9 Organization of Praxis

The chapters herein are organized as follows:

Chapter 1 - Introduction: This chapter introduces the readers to the scope of the praxis.

Chapter 2 - Literature Review: This chapter reviews related research, dataset reviews, and related background which led to the creation of this research.

Chapter 3 - Methodology: This chapter reviews how the CERT dataset was modified and provides an overview of TensorFlow, Rescaling, and other information relevant to the experiment design.

Chapter 4 - Results: This chapter reveals performance measurements using TensorFlow and the evaluation of the research questions and hypotheses.

Chapter 5 - Discussions and Conclusions: This chapter reviews the findings of how this research contributes to the body of knowledge and shares ideas for future work.

Chapter 2: Literature Review

2.1 Introduction

This literature review will provide the context required to gain a better understanding of Insider Threat detection and classification using ML and guide the praxis to answer the research questions. This literature review will present a comprehensive survey of research regarding Insider Threats using ML models, and will review datasets used for Insider Threat research, and report on previous model performance. Examination of Insider Threats, as a focus of research, has matured over the last decade (Securonix, 2024; Luckey et al., 2019).

2.2 Machine Learning

IBM defines ML as a branch of Artificial Intelligence (AI) and Computer Science focusing on using data and algorithms to enable computers to imitate human interpretation of information. ML was defined in 1959 by pioneer Arthur Samuel as a field of study that gives computers the ability to learn without prior programming (Stewart, 2017). Some early adoptions of ML technology date to the 1990s with the introduction of the email spam filter, which was designed to sideline unwanted emails.

IBM breaks down ML into three main parts: (a) Decision Process, which is used to make predictions or classifications; (b) Error Function, which assesses the predictions of the model for accuracy; and (c) Model Optimization Process, which fits the model to the data points and makes necessary adjustments to reduce discrepancies until a performance threshold is achieved (IBM, 2021).

It is important to note the distinctions between ML, Deep Learning (DL), and Neural Networks (NNs). All three terms are sub-fields of AI; however, NNs are a subset of ML, and DL is a subset of NNs. NNs consist of node layers, with multiple inputs and an associated output. Each node is linked to a threshold which activates once the output on the specific node passes that threshold, with the data then sent to the next layer(s) within the network (IBM, 2021).

2.3 Machine Learning Methods

According to IBM, supervised machine learning comprises the use of labeled datasets to train algorithms to classify outcomes accurately (IBM, 2021). This is a cross-validation process which is repeated until the model fits appropriately. This praxis relies on the CERT r4.2 labeled dataset to build the Neural Network models in TensorFlow. The Multi-layer Perceptron (MLP) learner or model is a feedforward (one direction) NN approach, which the computer uses to train on a dataset of user activities. This is one example of classification of user activity as malicious or benign (TensorFlow, n.d.-d). This is discussed in detail in Section 3.3.1. Other common methods of supervised learning include Naïve Bayes (NB), Linear Regression (LR), Logic Regression (LOR), Random Forest (RF), and Support Vector Machine (SVM) (IBM, 2021).

An unsupervised machine learning algorithm analyzes unlabeled datasets and can discover patterns in the data without the need for human intervention. Some common unsupervised learning algorithms include the implementation of the process of dimensionality reduction which essentially streamlines the number of inputs while keeping the important information required to make decisions (The IoT Academy, 2025). This can be achieved using various techniques such as Principal Component Analysis

(PCA), Singular Value Decomposition (SVD), Linear Discriminant Analysis (LDA), t-Distribution Stochastic Neighbor Embedding (t-SNE) and Autoencoders (The IoT Academy, 2025). Autoencoders are neural networks that compress the data and help to detect new patterns (The IoT Academy, 2025). The application of autoencoder to achieve dimensionality is the technique used in this praxis to convert categorical data to numerical data. This praxis will use a dummy-encoder, a variant of One-hot-encoder. Other algorithms include the use of K-means clustering, and probabilistic clustering methods (scikit-learn, 2025b).

Semi-supervised learning combines both supervised and unsupervised learning by using a smaller labeled data sample during training to guide the classification and feature selection from a larger unlabeled dataset. This approach is particularly appropriate when there are limitations on obtaining a complete labeled dataset (IBM, 2021).

Reinforcement machine learning uses the concept of supervised learning; however, it is not dependent on training data to make decisions. The model learns on live data as it goes along and uses the sequence of successful outcomes to develop its recommendations. Due to time constraints, this praxis will not focus on the concept of reinforcement learning, which requires months of training time, instead focusing on supervised learning (IBM, 2021).

2.3.1 Neural Networks

Given that this praxis focuses on NN, a review of current research on NNs follows below. Google documentation states that NNs are ML models containing at least one hidden layer. NNs are designed to mimic the working of a human brain using three

layers: input, hidden layers and output. The elements described below are core components of the NN architecture, illustrated in Figure 1.

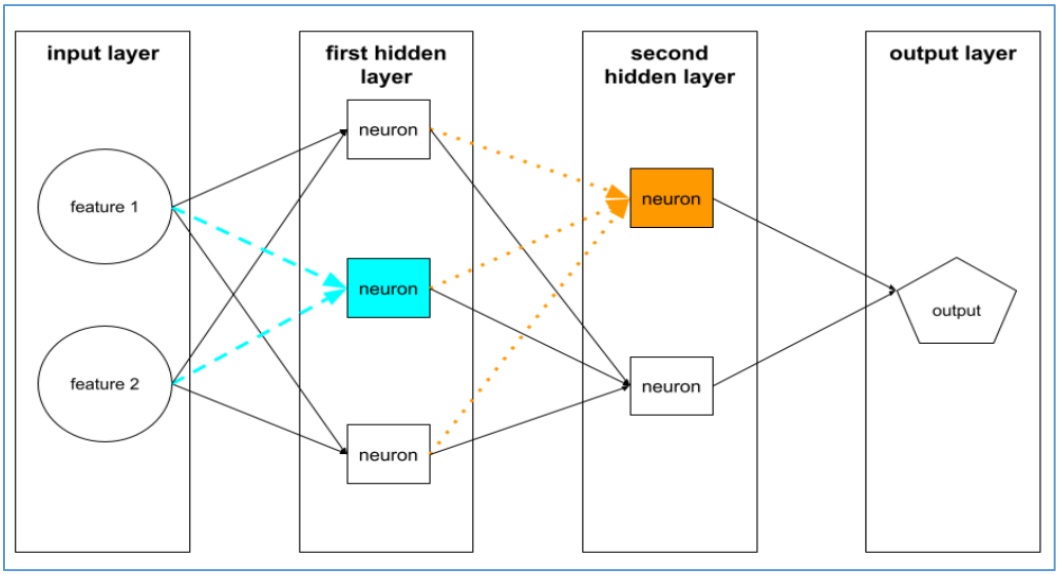

Figure 1: Neural Network Architecture
(Google Developers, n.d.; GeeksforGeeks, n.d.)

- **Neuron**: The unit within a hidden layer of a neural network which calculates the weighted sum (the sum of all relevant input values multiplied by their corresponding weights) of the input which then passes the information to an activation function (Google Developers, n.d.)
- **Activation function**: Activation Functions allow the NN to learn complex relationships among data. Some popular activation functions include the Rectified Linear Unit (ReLU) which produces an output of zero when a zero or a negative input is received. If a positive input is received the output will be the same as the input. Sigmoid Activation Function, in contrast, takes any input value and converts it to a range of 0 to 1, represented by the following equation (one) and Figure 2 below (Google Developers, n.d.). Both ReLU and Sigmoid functions

were used in the learners assessed in this praxis and are reviewed in detail in Section 3.3.1.

Equation 1: Sigmoid Equation 1

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \quad (1)$$

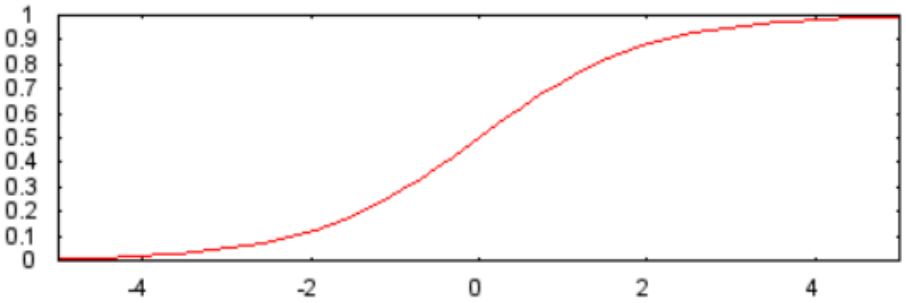

Figure 2: Sigmoid Activation Function Plot
(Google Developers, n.d.)

Figure 2 shows a summary of the Sigmoid Activation Function Plot which represents the transformation of the input values of a function, expressed in values ranging from 0 to 1.

2.4 Insider Threat Datasets

While this praxis will be centered around the CERT r4.2 dataset, the work of Carnegie Mellon University's Software Engineering Institute's Community Emergency Response Team is largely based on a sample of very specific Insider Threat cases. It therefore poses a challenge for researchers to gain a true understanding of the insider threat problem by attempting to generalize these instances as a true representation of all organizations (Cappelli et al., 2012). Other developed datasets have been made available to researchers and are described below.

2.4.1 The Wolf Of STUD (TWOS) Dataset

TWOS dataset was developed to obtain realistic instances of Insider Threats by capturing interactions emulating scenarios of users in a gaming competition (Homoliak et al., 2017). The competition consisted of 24 students who competed for five days during which their activities were captured. The captured activity included mouse clicks, keystrokes, processing transactions, filesystem monitor network traffic, emails, and login/logout. In summary, 320 hours of activity were captured, including 18 hours of masquerader data and two instances of traitor data (Homoliak et al., 2017). The email logs were analyzed using supervised machine learning. Results were fed into predictive models using linguistic analysis, which was then used to determine employee risk level by focusing on emails (Selvam & Selvy, 2022). Janjua et al. (2020) used the same dataset, the results of which showed a 98.3% AUC for AdaBoost (Janjua et al., 2020).

2.4.2 ENRON Dataset

The ENRON dataset consists of a collection of 500,000 messages from about 150 users, mostly senior management (CALO Project, 2015). This information was originally made public by the Federal Energy Regulatory Commission during its investigation. The email dataset was later purchased by Leslie Kaelbling at MIT (CALO Project, 2015). This dataset only consisted of email interaction; most of the other critical user activities were not made available to the public due to privacy concerns (CALO Project, 2015). This underscores one of the limitations associated with acquiring an appropriate dataset needed for Insider Threat research. Hence the ERON dataset was deemed inappropriate for use in this praxis.

2.4.3 APEX ‘07’ Dataset

The APEX ‘07’ dataset was collected by the National Institute of Standards and Technology in 2007. The dataset consisted of eight benign users and five malicious users. The focus of this dataset is inconsistent user behavior determined by a baseline profile (Santos et al., 2012). One of the limitations of this data was the lack of variety in Insider Threat scenarios; not having a dataset with sufficient information creates a challenge in understanding how different types of Insider activities may impact detection performance (Santos et al., 2012).

2.4.4 DARP 1998 Dataset

The DARP 1998 dataset was created by MIT’s Lincoln Laboratory as an intrusion detection test bed that simulated traffic against a large government website (MIT Lincoln Laboratory, 1999). The simulation consisted of more than 300 instances of 38 different automated attacks launched against a UNIX host over a seven-week period for training, and two weeks of testing data (Lippmann et al., 2000). As much as this dataset produced a significant amount of data, the nature of the data focused on intrusion detection rather than Insider Threats. Hence, this dataset was less desirable for the targeted research questions in the praxis.

2.4.5 NSL-KDD Dataset

The NSL-KDD was created from the KDD-99 dataset, which is a subset of the DARPA-98 cup dataset from 1998 (MIT Lincoln Laboratory, 1998). It was derived from TCP dump data simulated from network traffic captured at the Lincoln Labs (Ravipati & Abualkibash, 2019). The NSL-KDD dataset was proposed by Tavallaee in a 2009 article introducing a strategy to solve issues associated with the KDD-99 dataset. The dataset

consists of 125,972 total events across five classes (Tavallaee et al., 2009). The NSL-KDD was created based on the DARP 1998 dataset and therefore presented unanswered challenges with regard to reliability and representation of real network scenarios (McHugh, 2000). Therefore, this dataset was deemed as an undesirable choice to address the research questions in this praxis.

2.4.6 CERT Dataset

The CERT dataset is a publicly available dataset, developed between 2000 to 2011, and is widely used in Insider Threat detection studies due the comprehensive background data and synthetic malicious actor scenarios (Cappelli et al., 2012). There are several versions of the CERT dataset, with the most-used versions being r4.2 and r6.2 (Cappelli et al., 2012).

The CERT r4.2 dataset consists of 32,770,222 events from 1000 users. The dataset consists of 70 known Insiders, which is significantly greater than the number of known insiders from other commonly used dataset versions, such as r6.2, which consists of 2500 total users with 135,117,169 activities, but only 5 known insiders. Given that the focus of this praxis is malicious Insider events, a modified version of CERT r4.2 dataset was selected as detailed below (Bin Sarhan & Altwaijry, 2023). Unlike other datasets, CERT r4.2 dataset contains a significant number of known Insider activities (70) which will become a helpful factor for achieving randomness, which is critical for hypothesis testing in this praxis.

2.4.7 Dr. Noever's Dataset

While there are no clear dependent variables in the CERT r4.2 dataset, Dr. Noever created a modified dataset analyzed in his work *Classifier Suites for Insider Threat*

Detection (Noever, 2019). Noever's modified dataset offers an opportunity to gain further insight into open questions regarding additional Neural Networks models, and whether rescaling can improve performance (Manoharan et al., 2023). This dataset is discussed in further detail in Section 3.2, and is the dataset used in this praxis.

2.5 Types of Insiders

Insiders can be classified into several categories with the primary factor as disgruntlement, which turns an employee into an Insider Threat (Nurse et al., 2014). Depending on the factors that contributed to the disgruntlement, the individual may turn to theft, sabotage or fraud (Cappelli et al., 2012). The following categories are discussed below as malicious insiders, compromised insiders, careless insiders, contractors, inadvertent insiders, and disgruntled employees.

Malicious insiders are individuals who maliciously and intentionally abuse legitimate credentials, typically to steal information for financial or personal gain. In this instance, a disgruntled employee may sell their employer's secret information to a competitor (Imperva, n.d.).

Compromised Insiders leverage legitimate user account credentials which, when stolen, may enable an attacker to gain access to sensitive company systems and assets (Cyberseer, n.d). Careless Insiders are among the most common types of Insiders and include anyone who unknowingly exposes the system to outside threats (Imperva, n.d.). Contractors are Insiders with privileged access to sensitive information that can be traded (Manoharan et al., 2023). Inadvertent Insiders are Insiders who make common mistakes and do not adhere to the organization's security policies (Manoharan et al., 2023).

Disgruntled employees within an organization often feel they are not treated with sufficient respect and appreciation (Manoharan et al., 2023).

2.5.1 Insider Threat Classification

Cappelli et al. (2012) defines Malicious Insider Threats "as a current or former employee, contractor or business associate with access to an organization's assets. These individuals may misuse their access in a manner that negatively impacts the organization's confidentiality, integrity and availability or information systems" (Cappelli et al., 2012, p.354). The CERT dataset used in this praxis has identified three profiles associated with Insider Threat crimes, as follows:

1. **Information Technology (IT) Sabotage**: IT Sabotage describes malicious, technically privileged users who have the access and ability to launch an attack on an organization or an individual (Cappelli et al., 2012).
2. **Intellectual Property (IP) Theft:** IP Theft describes the theft of sensitive information that must be protected such as strategic or business plans, engineering or scientific information source code and other critical information (Cappelli et al., 2012).
3. **Fraud**: Fraud involves use or manipulation of any data without proper permission. Crimes related to fraud are usually committed by lower-level staff such as clerks and customer service. These crimes are motivated by financial need (Cappelli et al., 2012).

Other professional bodies may include National Security Espionage, which is described as the act of spying on an entity to gain national defense information with intent to allow a foreign nation to gain an advantage over the United States. There are

many similarities between National Security Espionage and all three types of profiles previously discussed. This research did not explore espionage since the CERT Insider Threat Center categorizes espionage separately from Insider Threats (Cappelli et al., 2012).

2.6 Research Related to Insider Threat Detection and Model Performance.

Table 1 shows a comprehensive list of research related to datasets that have been used to demonstrate the performance of various machine learning algorithms. Most of the research conducted around insider threat detection has used the CMU-CERT dataset, which will be the case for this praxis. Various researchers have demonstrated the possibility of leveraging models to achieve better performance. These approaches span from 2018 to 2024 and can be referenced in Table 1 as follows. According to the literature review, some researchers explored Neural Network however, there was a lack of research completed on the use of TensorFlow Neural Network.

Table 1: Related Research on Insider Threats Model Performance

Year of Publication	Author in Bibliography	Dataset	Models	Best Performing Algorithms
2024	Al-Shehari et al.	CERT r4.2	XGBoost, RF, DT, KNN, DBLOF	DBLOF achieved the highest F-score of 0.99 and accuracy of 0.98, this is slightly lower than the baseline, most likely skewed due to class imbalance.
2024	Farouk, Sakr, & Hikal	NSL-KDD	AdaBoost, DT, LR, GB, Linear SVM, RF, ET	AdaBoost at 0.992 AUC.
2024	Wei, Rauf, & Mohsen	CERT r4.2	LOF, Information Gain with Shannon Entropy, RF	RF = with accuracy of 95% LOF and Information Gain with Shannon Entropy accuracy = 98.48
2023	S. Besnaci, M. Hafidi, & M. Lamia	CERT r4.2	S-LSTM, SMOTE, AD-DNN	S-LSTM with AUC =99%
2023	U. Rauf, Z. Wei, & F. Mohsen	CERT r4.2	LOF, NN	NN had the best performance
2023	AlSlaiman, Salman, Saleh, & Wang	CERT r4.2	CM, LSTM, GE	GE exhibited superior performance in detecting insider threat AUC 0.97. GE was successful in maintaining temporal features on the dataset.
2023	Anivta Ajay Mahajan	CERT r4.2	LOF, IF	The hybrid approach performed better than the individual model. The IF training

Year of Publication	Author in Bibliography	Dataset	Models	Best Performing Algorithms
				accuracy of 97.09 and testing accuracy of 89.95. On the other hand, the LOF training accuracy was 99.91 and testing accuracy of 99.88. The results for the approach achieved training accuracy of 99.99 and testing accuracy of 100.
2022	Al-Shehari & Alsowail	CERT r4.2	Extreme GB, RF, DT, KNN, IF	DT performed best in the baseline with accuracy of 99% and detection rate of 64% and f-score of 66%. The IF algorithm with contamination parameter of 0.02 achieved high accuracy and indicated its ability to classify both normal and anomalous instances correctly.
2023	Bin Sarhan & Altwaijry	CERT r4.6 & CERT r6.2	OCSVM, IF, NN, SVM, AdaBoost, RF	SVM achieved the best performance successfully identifying all threat users.
2023	Al-Shehari et al.	CERT r4.2	XGB, RF, DT, and KNN	DT 94% ROC-AUC for under sampling and KNN 87% ROC-AUC when oversampling and hybrid sampling methods are utilized.
2022	S. K. Mamidanna, C. R. K. Reddy, & A. Gujju	CERT r4.2	XGBoost	Hyperparameter tuning on XGBoost
2022	Bartoszewski, F. W.	CERT r4.2 & CERT r6.2	Unsupervised models: SVM, LOF, Ensemble, Isolation Forest, Clustering, HMM	AUC 0.651 for IF and 0.629 for HMM ACC= 0.917 LOF ACC=
2021	D. C. Le & N. Zincir-Heywood	CERT r4.2 & CERT r6.2	AE, IF	Autoencoder using percentile representation is a better combination for anomaly detection. The isolation forest demonstrated robustness under extremely adversarial conditions. Isolation Forest has a clear advantage in detection performance.
2022	Selvam, A., & Selvy, T.	TWOS dataset	AdaBoost, KNN, SVM, Logic R, NB, Linear Regression	AdaBoost at 98.3% AUC
2021	Al-Shehari & Alsowail	CERT r4.2	LR, DT, RF, NB, K-NN, KSVM	DT outperforms all other ML techniques. SMOTE performed better at detection results.
2021	Nasser Al-Mhiqani, Ahmed, Zainal Abidin, & Isnin	CERT r4.2	AD-DNN, ADASYN, DNN, LSTM, OCSVM, SVM	AD-DNN
2021	Mohammed Dosh	CERT r4.1 and CERT r4.2	RF, Naïve Bayes, Nearest Neighbor	Nearest Neighbor had the best performance
2021	E. Pantelidis, G. Bendiab, S. Shiaeles, & N. Kolokotronis	CERT r4.2	Variational Autoencoder, Autoencoder	VAE accuracy 96% & F1-Score 94% illustrates the ability to correctly classify most of the samples.
2021	Nepal & Joshi	CERT r4.2	GRU	GRU obtained an AUC = 0.8732

Year of Publication	Author in Bibliography	Dataset	Models	Best Performing Algorithms
2020	Ren, X., & Wang, L.	CERT r4.2	Ed-Score and DMNAED, PCA	DMNAED outperforms LSTM, SOM & Graph
2020	Janjua, Masood, Abbas, & Rashid	TWOS dataset	AdaBoost, KNN, SVM, LR, NB	AdaBoost at 98.3% AUC
2020	Sharma, Pokharel, & Joshi	CERT r4.2	LSTM Autoencoder, OCSVM based on DBN	Accuracy of 90.17% for LSTM
2019	David A. Noever	CERT r4.2	RF, SVM, DT	Random Forest
2018	Yuan et al.	CERT r4.2	LSTM, CNN	LSTM-CNN produced AUC = 0.9449

Abbreviations can be found in the List of Acronyms

Studies listed in Table 1 deployed a combination of ML and some NNs, with the wide majority focusing on ML as the primary driving factor and motivation for their research. Selected studies in Table 1 are mentioned in further detail below as appropriate.

It is important to define some concepts introduced in the table to gain a better understanding of machine learning classification and performance. The concept of Neighborhoods Density is used in ML in association with algorithms such as K-Nearest Neighbors (KNN) to represent the proximity of data points used to predict the outcomes of an event (Future Machine Learning, n.d.). Outliers are often defined as noise or miscalculated data points which differ significantly from the general population of events (GeeksforGeeks, 2024).

Local Outliers (LO), which represent anomalies in relation to neighborhood densities, are calculated for each data instance using the K-nearest neighbors (Al-Shehari et al., 2024). The Density-Based Local Outlier Factor (DBLOF) is an advanced method of outlier detection and is most effective when applied in datasets with varying densities (scikit-learn developers, 2025a). A highly skewed dataset displays an imbalance between

normal/benign user activity and instances of malicious/abnormal insider activity (He & Garcia, 2009).

The Local Outlier Factor (LOF) is an unsupervised anomaly detection method measuring the local density deviation of the data point in relation to its neighbors to make determination on outliers or anomalies (scikit-learn developers, 2025a). Wei et al., (2024) describes LOF as an algorithm assessing the distance between data samples and their neighboring point to identify samples that present significantly lower density compared to their neighbors (Wei et al., 2024).

The CERT r4.2 dataset is characterized by a skewed distribution, in favor of benign activity. Al-Shehari et al. (2024) used the same dataset in a study separating malicious activity from benign activity. The researchers capitalized on the algorithm's ability to focus on the local density deviation of the data point, enabling the identification of outliers indicating potential Insider Threats. The DBLOF achieved an F1-score of 99% with accuracy of 98% (Al-Shehari et al., 2024).

Similar research performed by Wei et al. (2024) presents a hybrid feature engineering approach to Insider Threat detection by measuring Information Gain Entropy. Information Gain Entropy is defined as the change in uncertainty after the dataset is split and essentially measures the mixture of the classes in a dataset when making decisions (scikit-learn developers, 2025b).

It should be noted that this hybrid approach combines statistical methods with ML-based classification to determine if behavior is malicious. In this approach, Information Gain evaluates how much information is contained in a sample, while the LOF machine learning method assesses the distance between the samples and

neighboring points. The results demonstrate the ability of LOF and Information Gain to produce an Accuracy of 98.48%, exceeding Random Forest (RF), which produced 95% accuracy (Wei et al., 2024).

The CERT r4.2 dataset consists of 7% malicious to normal/benign events. This characteristic qualifies the dataset as biased or skewed, thus the random under-sampling technique would be very effective (Noever, 2019). The Random Under-sampling technique reduces the majority class (benign) training dataset by randomly selecting samples of insider observations. This results in the creation of a new version of the dataset which is fitted for the classification process of the machine learning algorithm (Al-Shehari et al., 2023).

Conversely, the random over-sampling technique increases the observations of the minority class (malicious). In this process, the minority class would be duplicated to create a new version of the training set. Over-sampling has the potential to improve the performance of machine learning algorithms influenced by skewed distributions (Al-Shehari et al., 2023).

The Hybrid resampling solution introduces the concept of combining a modest amount of the minority class (oversampling) and a modest amount of the majority class (undersampling) to create a new version of the dataset. Hybrid resampling mitigates limitations associated with losing important information due to under-sampling and overfitting. The combination of both techniques can lead to an improved overall performance compared to applying these techniques individually (Al-Shehari et al., 2023).

Al-Shehari et al. (2023) revisited the Insider Threat detection model that addressed the class imbalance using anomaly-based detection with the Isolation Forest (IR) algorithm. Their proposed model detected malicious insider behavior by treating these activities as outliers relative to normal user activity, which comprised a majority of the observations in the dataset. During this research. Decision Tree (DT) performed the best in the baseline with an accuracy of 99% (Al-Shehari et al., 2023).

Al-Shehari et al. (2023) also identified some promising results in research which proposed an Insider Threat data leakage detection model which leveraged random sampling techniques such as Random under-sampling, Random over-sampling and Hybrid resampling. The ML algorithms (XGB, RF, DT and KNN) were all applied with their default parameters. A relevant set of features was extracted from the CERT r4.2 dataset as a representation of a data leakage scenario producing results in favor of a ROC-AUC of 94% for Decision Tree (DT) under-sampling and ROC-AUC of 87% for K-Nearest Neighbors (KNN) when testing with over-sampling and hybrid sampling (Al-Shehari et al., 2023).

2.7 Research Related to TensorFlow

In the book *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* by Géron, TensorFlow is described as a powerful library for computation of large-scale machine learning (Géron, 2023). TensorFlow, developed by Google, was offered as open-source in November 2015. TensorFlow is now one of the most widely used deep learning libraries for machine learning tasks such as classification, which will be applied in this praxis.

In 2022, research was published on an unsupervised ML-based anomaly detection approach used for Insider Threat detection (Le & Zincir-Heywood, 2021). The study focuses on the use of Autoencoder (AE) and Isolation Forest (IR) with temporal information, which is a representation of the times or dates associated with discrete insider activities. The data processing was performed using Python 3 and Autoencoder was implemented using Keras with a TensorFlow backend (Le & Zincir-Heywood, 2021).

Similar research was conducted to investigate the effectiveness and usefulness of using Autoencoder and Variational Autoencoder Deep Learning algorithms to defend against Insider Threats without human intervention (Pantelidis et al., 2021). This was implemented using TensorFlow, and results demonstrated superior overall performance of Variational Autoencoder Neural Network. The results of the performance metrics achieved Accuracy of 96% and an F1-Score of 94%, which demonstrates the ability to correctly classify most samples (Pantelidis et al., 2021).

Chapter 3: Methodology

3.1 Introduction

This praxis examined the potential for categorizing and detecting Insider Threats using TensorFlow NN to optimize procedures for security professionals. The researcher will test 15 different NNs against a prepared dataset to identify which NN is most effective at identifying and classifying Insider threats. The goal is to demonstrate that the use of the CERT r4.2 dataset combined with the application of TensorFlow NN and leveraging four different rescaling techniques will prove to offer faster, better identification and classification of Insider Threats with fewer risk labels.

This chapter will describe methods used to conduct the praxis including pre-processing of the selected dataset as well as the selection of features, model development and rescaling. The chapter concludes with an overview of statistical analysis and experiment design.

It is necessary to explore the features of the dataset and feature selection before proceeding with the development of the models. The dataset of this praxis relies on publicly available synthetic versions of CERT r4.2 from the Carnegie Mellon University's CERT Division (Cappelli et al., 2012).

According to Al-Shehari and Aldsowail (2022), this dataset could be characterized as mostly imbalanced; however, compared to the datasets summarized in Chapter 2, Section 2.4, CERT r4.2 offers a better opportunity to support the research initiative for this praxis, and gain a better understanding of NN model behaviors due to the large number of observations it contains (Al-Shehari & Alsowail, 2022). Class

imbalance is a common issue in ML, which can create bias in the result. However, the total number of observations in the dataset provided a reasonable opportunity for the models to learn and produce results to support the research objectives.

Prior to the test, preprocessing was needed as a critical step to foster accuracy, efficiency and effectiveness for model learning. This step ensures all missing values are handled appropriately, categorical data is converted into a numeric format, outliers are removed to create a more robust model, and relevant features are selected to ensure optimal model performance. Finally, the data is split into a test and training set to evaluate the model's performance on unseen data. By addressing all these activities, the dataset becomes suitable for building the machine learning models (GeekforGeeks, 2025b; scikit-learn, 2025c).

3.2 Dataset Preprocessing and Constructions in Dr. Noevers' Research

Dr. Noever used the CERT r4.2 dataset to build a classifier suite for Insider Threat detection (Noever, 2019). Noever's method included the extraction of risk indicators, also known as features, which are summarized in Table 2, and which were determined using real network behaviors with threat descriptions. An example of a risk indicator in Table 2 is hr_leak, which represents unauthorized work hour activity for an employee. Such an employee was deemed by Noever as suspicious; therefore, such behaviors are included as a feature or risk indicator (Noever, 2019). Research conducted by Noever (2019) demonstrates that tracking 17 features or risk indicators using the RF algorithm as a classifier produced the highest detection accuracy at 98%.

Random Forest is a machine learning method that can be used for classification. It is used in application of feature importance by combining various decision trees trained

on a random subset, and the average of the outcome is used to predict accuracy and control over-/under-fitting (Mesiar & Sheikhi, 2021; scikit-learn, 2025c). Noever (2019) initially tracked 28 features but concluded that only the 17 features listed in Table 2 were essential indicators (Noever, 2019). This current praxis utilized only neural network approaches to create a modified version of the CERT r4.2 dataset provided by David Noever in order to ascertain if a further reduced number of features could achieve similar accuracy to Noever's indicator (Noever, 2019). Section 3.2.1 describes the process by which the 17 features were reduced to 10.

Table 2: Risk Indicators Extracted from CERT Dataset Features

Risk Indicator	Description of Features
risk_leak	Count of events from email or web logs linked to file sharing domains.
dow_leak	Weekend days identified and counted as leak risk
hr_leak	Non-authorized work hours as defined by HR which are counted as leak risk (Outside of 8AM – 5PM).
risk_thief	Count of events from email or web logs linked to job searches or competitors' domains.
dow_thief	Weekend activity count for thief risks.
hr_thief	Non-authorized work hours as defined by HR which are counted as thief risk (Outside of 8AM – 5PM).
risk_sabotage	Count of events from email or web logs linked to malware or keylogger domains.
dow_sabotage	Weekend activity count for Sabotage risks.
hr_sabotage	Non-authorized work hours as defined by HR which are counted as sabotage risk (Outside of 8AM – 5PM).
device_freq	Monthly USB device frequency usage on device.
file_freq	Monthly file access frequency, i.e. (word and PDF)
email_sentiment	Monthly sentiment polarity in email text (Positive or negative).
email_compete	Count of monthly emails sent to competitors.
web_sentiment	Monthly sentiment polarity in visited website text (Positive or negative).
executables	Count of monthly executable files accessed.
Unauthorized_log	Unauthorized usage of employee device following a departure.
gender	The employee gender based on first name.

As cited by Noever, 2019.

3.2.1 Feature Selection Using Random Forest

To ensure efficient processing time, not all Noever's dataset features are needed (Labory et al., 2024). The top performing model in this praxis ran almost a minute quicker than the top performer of Dr. Noever. Of the 17 features offered in Dr. Noever's dataset, this praxis selected 10 features. Random Forest algorithm, when used for feature selection, rather than simply a detection algorithm, has demonstrated the ability to accurately identify salient features (Labory et al., 2024; Mesiar & Sheikhi, 2021).

This approach was chosen after consideration was given to variance and correlation. A feature variance uses sample data to infer if there is difference in how the feature is spread in comparison to the average value of the feature (Frost, 2020). When performing feature variance analysis, features with the lowest variance or no variance should be dropped as they are an indication of an undesirable predictor of a label (Frost, 2020). In contrast, correlation refers to the extent to which two events are related to each other (Wheelan, 2013; Frost, 2020). For example, there is a correlation between ice-cream sales and summer temperature. When correlation is used to compare two features against a target label, the result can be positive or negative. Positively correlated features competing for the same feature space may suggest that weaker correlation could be dropped, as features with a higher correlation to the target label are more desirable selections (Wheelan, 2013; Frost, 2020).

Relying on variance and correlation for feature selection assumes that features are normally distributed. Out of concern that features were not normally distributed, this praxis rejected the variance and correlation approach. A test was conducted, and the results are in Appendix A, which clearly demonstrates the Random Forest approach as

shown in Figure 3 and Figure 4 is superior. Use of Random Forest for feature selection is described below.

Mesiar and Sheikhi (2021), point out that RF can be used as an alternate method for determining the most important features when a dataset is non-linear. The following are assumptions for this technique, which this praxis met (Balakrishnan, 2021).

1. **Independence of Trees**: The decision trees generated should be unique to every feature in the dataset.
2. **Sufficient Data**: The number of observations and dataset activity should be sufficient to generate the trees.
3. **Balanced Trees**: The algorithm assumes that there will be an equally random opportunity to explore the dataset to discover notable patterns.
4. **Noisy Data Handling**: The algorithm assumes that noise is randomly distributed as the data is explored.

With the assumptions met, RF was executed, and all features were ranked by importance, according to the feature importance function built into SKLearn (Balakrishnan, 2021). Figure 3 and Figure 4 below summarize these results, which indicate that hr_leak is the most important of all the 10 features followed by risk_leak, with dow_sabotage ranked as the least important of the 10 selected features. Figure 3 represents a histogram of the scores ranked in Figure 4. The features shown in these figures are represented as the risk indicators summarized in Table 2.

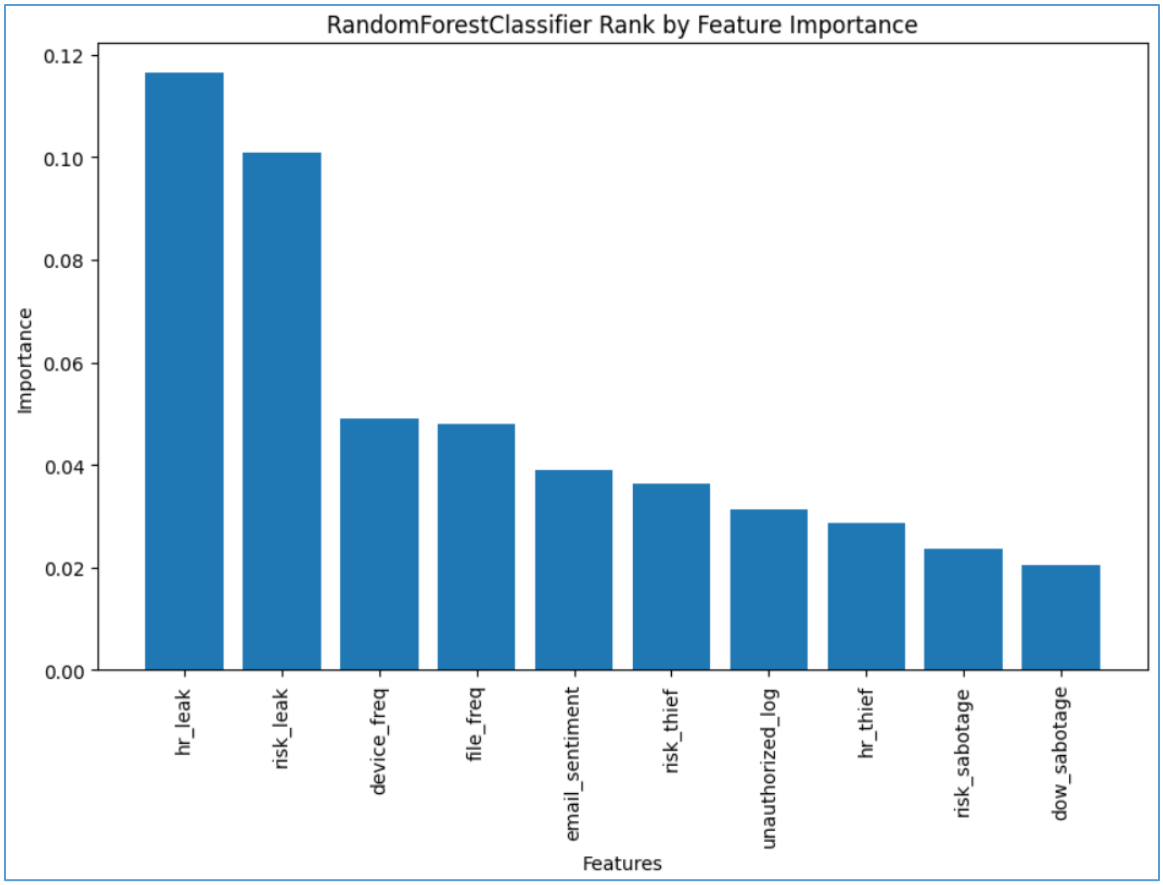

Figure 3: Random Forest Feature Selection Rank Order of Importance

Rank order	Feature Importance Rank	Score
1	hr_leak	0.116335505
2	risk_leak	0.100989934
3	device_freq	0.04902593
4	file_freq	0.048026153
5	email_sentiment	0.038962178
6	risk_thief	0.036244569
7	unauthorized_log	0.031277723
8	hr_thief	0.028693531
9	risk_sabotage	0.023605878
10	dow_sabotage	0.020360939

Figure 4: Top 10 Features by Rank Order of Importance

3.2.2 Use of SMOTE During Feature Selection

During feature selection, given that the CERT r4.2 dataset was unbalanced (7% malicious with 93% benign), the Synthetic Minority Oversampling Technique (SMOTE) was applied to the dataset, however, RF could better identify which features were desirable for selection. SMOTE balances the data by creating new synthetic data for the minority class, (in this case the malicious samples), by executing a linear interpolation with the minority class and their k-nearest neighbors (Elreedy et al., 2024). SMOTE ensured a 50% malicious to 50% benign sampling during the feature selection process; however, thereafter, the synthetic samples were not used in this praxis. The original, non-SMOTE adjusted dataset was analyzed in this praxis (Elreedy et al., 2024).

3.3 Model Generation, TensorFlow, Rescaling, and Performance Evaluation

The ten selected features were split into samples for training (70%) and testing (30%), referred to henceforth as a 70-30 split. The 70-30 split was performed using random sampling. Discussed in this section are the TensorFlow algorithms used in this praxis, the rescaling techniques evaluated, and how performance was evaluated.

3.3.1 TensorFlow NN Learners

Neural Network learners form an important part of the ML ecosystem and have evolved in their contributions to automating decision making and identifying complex patterns in dynamic environments (GeeksforGeeks, n.d.). Neural Networks are classified as Feedforward Networks which allow data to move in one direction from the input to the output, and Multilayer Perceptron (MLP) which is also a type of feedforward NNs which consists of three or more layers. A Convolutional Neural Network (CNN) focuses on image processing and classification. The final category is Recurrent Neural Network

(RNN) which is intended for sequential data processing and is useful for time series predictions and natural language processing. Long Short-Term Memory (LSTM) is a type of RNN which is focused on using memory cells and gates to make decisions on reading, writing and erasing information. The fifteen learners used in this praxis were selected due to their strengths in solving classification problems and fall in the categorizes of either feedforward or MLP (GeeksforGeeks, n.d.).

3.3.1.1 Radial Basis Function (RBF)

RBF is a kernel-based learner that excels at tasks such as interpolation, regression, and function approximation (Broomhead & Lowe, 1988). The function evaluates the distance between input points and a center to determine their influence. RBF is particularly effective in scenarios where relationships between variables are non-linear and complex (Broomhead & Lowe, 1988). In machine learning, the RBF kernel is commonly used in models such as support vector machines (SVM) and Gaussian processes (Amirian & Schwenker, 2020). It maps data into a higher-dimensional space, enabling linear separability of classes that are not linearly separable in the input space (Amirian & Schwenker, 2020).

In TensorFlow, RBF networks can be implemented through custom layers that define the RBF activation function either explicitly or by leveraging kernel methods available in the TensorFlow Probability library (Chollet, 2023; Stack Overflow, 2025). These implementations allow flexibility in model design for interpolation, density estimation, and non-linear regression tasks. For example, TensorFlow Probability provides tools to define Gaussian kernel density estimates, which are closely related to RBF networks, and facilitate probabilistic learning frameworks. This makes the RBF

approach versatile and powerful in addressing various computational challenges in machine learning and function approximation library (Chollet, 2023; Stack Overflow, 2025).

3.3.1.2 Radial Basis Function with Dynamic Decay Adjustment (RBF-DDA)

RBF-DDA modifies traditional RBF networks by introducing a dynamic decay mechanism to adjust the learning rate based on the model's performance (Hryniowski & Wong, 2019). It specifically focuses on the error rate during training. This approach aims to improve training efficiency and convergence speed (Neuroinformatics Group, n.d.; Hryniowski & Wong, 2019). In RBF-DDA the learning rate is not fixed, rather, it adjusts dynamically during the training process (Neuroinformatics Group, n.d.). The learning rate is modified based on the error rate, typically decreasing over time as the model converges. This helps the model to fine-tune its weight as training progresses. The classification accuracy is similar to MLP, but the training is significantly faster; this model usually requires only five Epochs (Neuroinformatics Group, n.d.).

As training proceeds and the error rate reduces, the learning rate is gradually decayed, allowing the model to make smaller updates to its weights, which reduces overfitting and improves generalization (Hryniowski & Wong, 2019). By adjusting the learning rate dynamically, RBF-DDA ensures that the model can focus on harder-to-learn patterns in the initial stages of training and refine its weights with smaller steps as it becomes more accurate, resulting in faster and more stable convergence (Hryniowski & Wong, 2019).

3.3.1.3 Radial Basis Function Network (RBFN)

RBFN is a type of NN where the hidden layer neurons use RBFs, typically with Gaussian functions as their activation functions (Broomhead & Lowe, 1988; Howlett & Jain, 2001). These networks are primarily used for tasks such as classification, regression, and function approximation due to their ability to model non-linear. The key strength of RBFNs is its ability to approximate non-linear functions. RBFs maps data into a space where linear models can better fit the data, making RBFNs powerful for tasks where the relationships between input and output are non-linear. Training the output layer using linear methods, such as least squares regression to find the best fit and predict the behavior of dependent variables, will essentially find the weights that map the hidden layer's outputs to the final predictions (Broomhead & Lowe, 1988; Howlett & Jain, 2001).

3.3.1.4 Determinantal Point Process (DPP)

DPPs are probabilistic models designed for subset selection tasks (Kulesza & Taskar, 2012). They are particularly useful in applications such as document summarization, recommendation systems, and feature selection, where diverse and representative subsets are preferred. DPPs assign higher probabilities to subsets with elements that are dissimilar to each other, thereby promoting diversity while ensuring coverage (Kulesza & Taskar, 2012).

In TensorFlow, DPPs can be implemented using the TensorFlow Probability library, which provides functions for sampling and modelling probabilistic distributions (TensorFlow, 2023). For example, the library includes tools for defining similarity kernels and sampling diverse subsets based on those kernels (TensorFlow, 2023). This

capability makes TensorFlow an effective framework for integrating DPPs into machine learning pipelines, particularly when leveraging Graphic Processing Unit (GPU) acceleration for large-scale operations (Dillon et al., 2017). The application of DPPs with TensorFlow's extensive ecosystem allows developers to build efficient and scalable solutions for tasks requiring diverse subset selection, enhancing the quality of machine learning models and systems (TensorFlow, 2023.).

3.3.1.5 Extreme Learning Machine (ELM)

ELM is a type of single-layer feedforward NN that differs from traditional NNs in its approach to training (Huang et al., 2006; Huang, 2015). Unlike conventional NNs, which iteratively update weights through backpropagation, ELM uses randomly assigned weights for the connections between the input and hidden layers. The weights from the input to the hidden layer are fixed and not adjusted during training using gradient descent. This results in a faster training process (Huang et al., 2006; Huang, 2015).

3.3.1.6 Extreme Learning Machine with Gaussian Kernel (ELM-klm)

ELM-klm is a variant of the Extreme Learning Machine (ELM) architecture which incorporates a Gaussian kernel transformation for mapping input data into a higher-dimensional space (Huang, 2015; Chollet, 2023; Struniawski, 2024). ELM-klm is characterized by its single hidden layer, where the weights are randomly assigned and not updated during training, enabling extremely fast training compared to traditional neural networks. The Gaussian kernel transformation in ELM-klm captures non-linear relationships within the data by computing similarity metrics between input instances and a set of random hidden layer nodes. This combination makes ELM-klm particularly

suited for regression, classification, and function approximation tasks requiring high efficiency (Huang, 2015; Chollet, 2023; Struniawski, 2024).

TensorFlow Probability can be used to enhance kernel computation efficiency, especially for large datasets, providing a scalable solution for deploying ELM-klm models in diverse applications. (Huang, 2015; Chollet, 2023; Struniawski, 2024). However, TensorFlow does not natively support ELM-klm as a built-in feature, although ELM-klm can be approximated. This approximate implementation can be achieved by designing custom layers and leveraging TensorFlow's functional API, using the following elements listed below. The required parameters were followed in the implementation of ELM-klm within this praxis. (Huang, 2015; Chollet, 2023; Struniawski, 2024).

- **Random Weight Initialization**: Create a custom layer to initialize random weights for the hidden layer. These weights remained fixed during training.
- **Gaussian Kernel Transformation**: Define a custom kernel layer to compute the Gaussian similarity between input features and the weights in the hidden layer.
- **Output Layer**: Use a standard dense layer for mapping the kernel-transformed inputs to the output, where the output layer weights are learned during training (Huang, 2015; Chollet, 2023; Struniawski, 2024).

3.3.1.7 Neural Network (NN)

TensorFlow’s NN algorithm is a DL architecture composed of layers of neurons, where each neuron in one layer is connected to every neuron in the next layer. (TensorFlow, n.d.-a; GeeksforGeeks, n.d.). It typically includes an input layer, hidden layers, and an output layer. Neural Networks learn to map inputs to outputs by adjusting the weights of connections during training through processes like backpropagation. This

process calculates the error between predicted output and the actual output which later transmits the results through the network to make adjustment to the weights of each neuron. (TensorFlow, n.d.-a).

In TensorFlow, NNs can be built using the Keras API, which offers two primary ways to define models: Sequential or Functional (TensorFlow, 2024b; TensorFlow, n.d.-b). The Sequential API can be used for models where layers are stacked in a linear sequence, with each layer having one input and one output (TensorFlow, 2024b; TensorFlow, n.d.-b). The Sequential API is comparatively simpler and easier to configure, suitable for straightforward architectures. In contrast, the Functional API offers better flexibility, allowing the creation of models with complex structures, such as multiple inputs or outputs and non-linear connections between layers (TensorFlow, n. d.-b).

The Functional API is ideal for more advanced Neural Network architectures (LeCun et al., 2015). Both APIs utilize TensorFlow's efficient computational power, including GPU acceleration, making it easier to train and deploy deep learning models. In this praxis, the Functional API was used (TensorFlow, 2024b; LeCun et al., 2015).

3.3.1.8 Averaged Neural Network (ANN)

The ANN is a form of ensemble learning where multiple NNs are trained independently, and their predictions are combined to produce a final output (Krogh & Vedelsby, 1995; TensorFlow, 2024d). This averaging can be done in different ways, such as Prediction Averaging, Weight Averaging and Snapshot Ensembles. The Prediction Averaging technique bases predictions on averaged multiple independently-trained NNs to produce a final output, improving the model's ability to perform well on new data

generalization and robustness. The Weight Averaging technique applies weights to multiple trained NNs that are then averaged to form a single model, ensuring smoother parameter optimization and reducing overfitting. Snapshot Ensembles is an approach where a single model is trained, and weights are saved at different stages (snapshots). The predictions from these snapshots are averaged, emulating an ensemble to make classification decisions without training multiple models (Krogh & Vedelsby, 1995; TensorFlow, 2024d).

3.3.1.9 Principal Component Neural Network PCNN

PCNN is a model that reduces the dimensionality of input data by projecting it into principal components (TensorFlow, 2024c; Jolliffe & Cadima, 2016). This technique is based on Principal Component Analysis (PCA), which identifies the most important directions (principal components) in the data, capturing the maximum variance while discarding less informative components. By reducing the dimensionality, PCNN aims to improve model performance to reduce overfitting, and accelerate training by working with a more compact representation of the data components (TensorFlow, 2024c; Jolliffe & Cadima, 2016).

In TensorFlow, PCA can be implemented and utilized through the TensorFlow Addons library, which provides a collection of additional functionalities that are not part of the core TensorFlow library components (TensorFlow, 2024c; Jolliffe & Cadima, 2016). TensorFlow Addons includes implementations for dimensionality reduction, including PCA, which can be used as part of the preprocessing pipeline before feeding data into NNs. Since PCA is sensitive to the scaling of features, the data should first be standardized (zero-mean) before applying PCA; however, this praxis ran PCNN with

each scaling format for sake of consistency components (TensorFlow, 2024c; Jolliffe & Cadima, 2016).

TensorFlow Addons provides PCA implementations to align the data into its principal components, reducing the dimensionality components (TensorFlow, 2024c; Jolliffe & Cadima, 2016). Once the data has been projected into principal components, it can be fed into a Neural Network model, which now operates on a lower-dimensional input. TensorFlow Addons makes it easy to integrate PCA into the workflow, allowing users to combine dimensionality reduction with other TensorFlow-based model training and evaluation task components (TensorFlow, 2024c; Jolliffe & Cadima, 2016).

3.3.1.10 Multi-Layer Perceptron (MLP)

MLP is a type of feedforward artificial NN consisting of multiple fully connected layers function (TensorFlow, n.d.-c; GeeksforGeeks, 2025a). It can be used for both regression and classification tasks. MLPs are structured with an input layer, one or more hidden layers, and an output layer. The key characteristic of MLP is that each neuron in a layer is connected to every neuron in the next layer, making it a fully connected network. MLPs are trained using backpropagation, where the loss (error) is propagated back through the network, and the weights are adjusted to minimize the error using an optimization algorithm such as Gradient Descent. MLPs can use an Activation Function to help the model analyze non-linear data, which in the case of this praxis was the Sigmoid function (TensorFlow, n.d.-c; LeCun et al., 2015).

TensorFlow's Keras API features an intuitive interface which is said to simplify training and building MLPs function (TensorFlow, n.d.-c). The linearly stacked layers in Sequential API in Keras make it particularly suitable for MLPs. The Keras Dense Layer

allows for the creation of fully connected layers in the network. The model in this praxis was compiled and optimized using the Adaptive Moment (Adam) function (TensorFlow, n.d.-c).

3.3.1.11 Multi-layer Perceptron with Weight Decay (MLP-wd)

For MLP, Weight Decay is a technique used to prevent overfitting in ML (Päpper, 2023; GeeksforGeeks, 2025a). It ensures that the model does not become too complex and sensitive to noise in the training data. Weight decay is a common technique in deep learning which improves the model's ability to generalize, making it essential for training robust NNs. MLP-wd works by adding a penalty to MLP's loss function, discouraging the model from assigning excessively large weights to the connections between neurons (Päpper, 2023; TensorFlow, n.d.-c).

MLP-wd is also known as *L2 regularization* (Päpper, 2023; GeeksforGeeks, 2025a). TensorFlow implements weight decay through the regularizer feature in Keras. L2 regularization can be applied to layers in NNs by specifying the kernel_regularizer argument in the layer definition as applied in this praxis (Päpper, 2023; GeeksforGeeks, 2025a).

3.3.1.12 Direct Kernel Perceptron (DKP)

DKP is a variant of the perceptron learning algorithm utilizing kernel functions to implicitly map data to a higher-dimensional space (Fernández-Delgado et al., 2014; GeeksforGeeks, 2025a). This allows the model to learn non-linear decision boundaries without explicitly computing the higher-dimensional representation. In this approach, the kernel functions calculate the similarity between data points in the input space, allowing the perception to make more complex decisions (Fernández-Delgado et al., 2014;

GeeksforGeeks, 2025a). While TensorFlow does not provide a direct, built-in implementation of DKP, it does support kernelized models through TensorFlow Probability (TensorFlow, n.d.-e). DKP uses kernel functions, much the same way as MLP does, which in the case of this praxis again was the Sigmoid function (Fernández-Delgado et al., 2014; GeeksforGeeks, 2025a).

3.3.1.13 Learning Vector Quantization (LVQ)

LVQ is a prototype-based supervised learning method that classifies data based on prototype vectors representing different classes (TensorFlow, n.d.-f; Kohonen, 1990). Each prototype corresponds to a class in the feature space, and the model assigns new data points to the class of the nearest prototype. These prototypes are adjusted during training to better reflect the characteristics of the classes they represent for classification. LVQ compares input data with prototypes and assigns the data to the class of the nearest prototype, typically using a distance metric such as the Euclidean distance or shortest distance. The idea is that similar data points should be closer to the same prototype, and in this regard, one can say that LVQ prototypes compete in order to best represent a given input (TensorFlow, n.d.-f; Kohonen, 1990).

While TensorFlow does not have a direct, built-in LVQ implementation, it does support LVQ-style learning through its embedding and similarity learning modules (TensorFlow, n.d.-f; Kohonen, 1990). These modules can be used to build models that learn feature representations (prototypes) for classification tasks. TensorFlow allows the creation of custom layers and loss functions, enabling the development of LVQ-based models where prototypes are learned and updated based on data points (TensorFlow, n.d.-f; Kohonen, 1990).

3.3.1.14 Cascade Correlation (CASCOR)

CASCOR is a self-organizing NN that incrementally adds hidden units (neurons) to the network as needed during training based on reducing error (Gimelfarb, 2025; Shultz, 2000; Lehtokangas, 2000). This method allows the model to grow dynamically, with new layers being added when the current network fails to sufficiently reduce error. CASCOR builds the network in a step-by-step process. At each step, the algorithm adds one or more neurons to the network, adjusting the weights of the existing neurons to further reduce error. The network's structure grows in response to error feedback, only becoming more complex when necessary, contributing to efficiency and avoidance of overfitting (Gimelfarb, 2025; Shultz, 2000; Lehtokangas, 2000).

3.3.1.15 Bi-directional Kohonen (BDK)

BDK is an extension of the Kohonen Self-Organizing Map which introduces bi-directional mapping between input and output layers (Melssen et al., 2006; TensorFlow, 2024-e). While the traditional Kohonen network only maps input data to output neurons, BDK extends this by allowing data to be mapped in both directions from input to output and from output to input (Melssen et al., 2006). Unlike the traditional Kohonen network, which only performs a one-way mapping, or forward feeding, (from input to output), the BDK network establishes a bi-directional relationship. This means that the model not only maps input data to a specific output neuron but also allows the output neurons to influence the input space, contributing to better data representation and clustering (Melssen et al., 2006; TensorFlow, 2024-e).

While TensorFlow does not have a built-in BDK implementation, BDK can be approximately implemented using embedding layers and custom training loops

(TensorFlow, 2024-e). The embedding layers allow the network to learn a compact representation of the input space. Custom training loops enable the dynamic adjustment of weights during training, simulating the bi-directional learning process (TensorFlow, 2024-e).

3.3.2 Data Scaling Techniques

It is important to prepare the dataset prior to the execution of any machine learning task, given that each algorithm is designed with an assumption of how data is or is not scaled (scikit-learn, 2025c). Each model may demonstrate better or worse performance depending on the scaling technique applied. In this praxis each model will be tested with four different scaling techniques to identify which combination achieves the best possible performance. Described below are four scaling approaches used in this praxis.

- **Normalization**: The process of normalization will scale the data in a range of zero (0) to one (1). Normalized scaling techniques are a significant area of discussion in this praxis as they will assist in determining which learners have a significant difference once that data is examined (Ali et al., 2014).
- **Standardization**: Standardized scaling is an alternative preprocessing technique which rescales the dataset features with a mean value of zero (0) and a standard deviation of one (1) (Ali et al., 2014). In this praxis the Standardization was configured in TensorFlow with Sk-Learn. This technique is typically applied to datasets with a normal distribution (Ali et al., 2014). It will be discussed in more detail when determining the statistical approach used in this praxis.

- **Non-Rescaled**: In this approach, no scaling techniques are applied to the dataset and the original distribution is preserved (scikit-learn, 2025e). Non-Rescaled datasets will likely contain a mixture of features with different scales, with some features having a normalized scale, some features having a standardized scale, or even something else. Testing Non-Rescaled data in TensorFlow with Sk-Learn can be achieved by not applying Standardization, Normalization or Robust.
- **Robust**: The Robust Scaling technique rescales each feature according to the interquartile range (IQR) of each feature (scikit-learn, 2025d). The IQR is the range between the 1st quartile (25th quartile) and the 3rd quartile (75th quartile). To this end, Robust works similarly to Standardization, but instead of rescaling features to a scale of negative to positive one, features are rescaled to fall within the feature's original 25th to 75th quartile. Comparatively, the Robust Scaler technique is more resistant to the presence of Outliers and is more like Standardization than Normalization (scikit-learn, 2025d).

3.3.3 Performance Model Metrics

In this praxis, the performance of learners was evaluated using the metrics described below.

- **Confusion Matrix (CM)**: The confusion matrix is a performance-based technique used to evaluate models by establishing a relationship between predicted values versus actual values (scikit-learn, 2025a). Confusion Matrices lists out scores for True Positives, False Negatives, False Positives, and True Negatives, as defined below.
- **True Positive (TP):** The count of malicious instances correctly detected.

- **False Negative (FN):** The count of malicious instances incorrectly detected or falsely detected as benign.
- **False Positive (FP)**: The count of benign instances incorrectly detected or falsely detected as malicious.
- **True Negative (TN)**: The count of benign instances correctly detected.
- **Accuracy**: Accuracy represents the number of correct predictions over the total number of predictions. The following equation (2) below represents how accuracy is determined based on the confusion matrix.

Equation 2: Accuracy Equation 2

$$Accuracy = \frac{\text{TP+TN}}{\text{TN+TP+FP+FN}} \quad (2)$$

- **Precision**: Precision is the amount of TP over the TP and FP. The following equation (3) below represents the Precision based on the confusion matrix components.

Equation 3: Precision Equation 3

$$Precision = \frac{\text{TP}}{\text{TP+FP}} \quad (3)$$

- **Recall**: Recall is represented by the number of TP over the TP and FN. The following equation (4) below represents Recall.

Equation 4: Recall Equation 4

$$Recall = \frac{\text{TP}}{\text{TP+FN}} \quad (4)$$

- **f1-Score**: F1-Score is a representation of the harmonic mean of the precision and recall. The following equation (5) below summarizes the f1-Score equation.

Equation 5: f1 Score Equation 5

$$f1\text{-}Score = 2\left(\frac{recall \text{ x } precision}{recall + precision}\right) \quad (5)$$

3.4 One-Way Analysis of Variance (ANOVA)

One-Way Analysis of Variance (ANOVA), commonly known as single-factor ANOVA, is a statistical approach used to evaluate if there is a difference between two or more categorical groups (Assaad et al., 2014). ANOVA is performed to determine if a statistically significant difference exists by comparing the mean and distribution of each category (Bevans, 2020). The interpretation of ANOVA is based at p-value, and ANOVA's null hypothesis of equal means.

ANOVA's null hypothesis (H_0) of equal means states that there is no significant difference in the means of all categories. ANOVA's alternate hypothesis (H_a) states that the means of at least one category is significantly different from the rest. A p-value is the chance of an event being true or false, or in this case the chance of if the null hypothesis can be rejected, or not, using a level of significance.

The level of significance in this praxis for all tests is 0.05. If the p-value is less than 0.05, ANOVA's null hypothesis of equal means can be rejected. If the ANOVA's p-value is more than 0.05, then the null hypothesis cannot be rejected.

When ANOVA detects the presence of a difference, a post hoc test is used to determine which categories are different (Frost, 2020). There are numerous kinds of post hoc tests; however, they share a common goal aimed to rank order group the discovered differences between group means, which ANOVA does not do (Frost, n.d.). Likewise, there are also variations of ANOVA.

To choose the correct variant of ANOVA, and the correct post hoc test, the underlying assumptions of the statistical test must be true. As will be discussed next, after

reviewing assumptions for ANOVA, the equivalent test to ANOVA used for this praxis was the Kruskal-Wallis Test. The correct post hoc test for Kruskal-Wallis, also used herein, was the Dunn's Test (Frost, 2020; Dinno, 2015).

3.4.1 Assumptions for One-Way Analysis of Variance

There are six assumptions that must be met for an ANOVA test to be valid (Laerd Statistics, n.d.).

1. **Categorical and Randomness**: The categorical aspect of this assumption for ANOVA requires three or more groups of approximately equal size. For categories to be roughly equal size, the count of samples of the largest category, when divided by the count of samples of the smallest category, should not exceed 1.5 (Frost, 2020). The assumptions of randomness and categorical separation were met. To achieve randomness, the dataset of this praxis represents a random sampling of the total population of Insider Threats seen in real-life, or at least those that were caught by CERT. The dataset was also segmented into training and testing sets based on a random 70-30 split, ensuring there was no sampling bias.
2. **Dependent**: The ANOVA assumption for dependent variable must be represented by a quantitative sample at a continuous scale level. An example of dependent variables used in this praxis included Accuracy and Time. These variables are all represented by interval measures that meet the continuous scale level requirement for ANOVA. The following assumption held true for the dataset used in this praxis (Frost, 2020).

3. **Independence of Observations**: The independence assumption requires that there is no relationship between the observations (i.e. Accuracy) in each category (i.e. Rescaled-Model-Epoch Triples) (Frost, 2020). Given that the accuracies, precision, recalls, etc. for any given learner are not conflated between groups, this assumption is met. This is to say, for example, the accuracies for the category of *Robust MLP with 35 Epochs* are also not dually recorded as the accuracies for *Standardized MLP with 35 Epochs* The following assumption was met in the dataset used in this praxis.
4. **Outliers and Proportional Samples**: Outliers are the values in a dataset that are notably different from other values. The proportional samples are the trials that are selected from a population to be used for testing for each category (Frost, 2020). To meet this assumption, outliers among performance results were removed. ANOVA permits the removal of outliers as long as the samples size requirement holds true in the random samples established in assumption one. In the case of this praxis, each category had a total of 21 random samples to be evaluated. The outliers were identified by checking the interquartile range fences (Vargas et al., 2024). Any category with fewer than 13 samples after removing outliers was dropped to ensure the roughly equal size requirement in assumption one held true.
5. **Homogeneity of Variance**: The homogeneity of variance assumption indicates whether different groups or categories have equal or a similar size of variance. The Levene's test can be used to confirm if there is homogeneity of variance among groups; however, the groups which were tested in this praxis did not have

homogeneity of variance, which was tested for using Minitab (SixSigma.us, 2024). As shown in Figure 5, below, no homogeneity of variance was concluded for learner Accuracies, given the p-value of the Levene's test was less than 0.05. As will be discussed at the end of the next assumption, an alternative to ANOVA was needed (Minitab, 2024).

6.

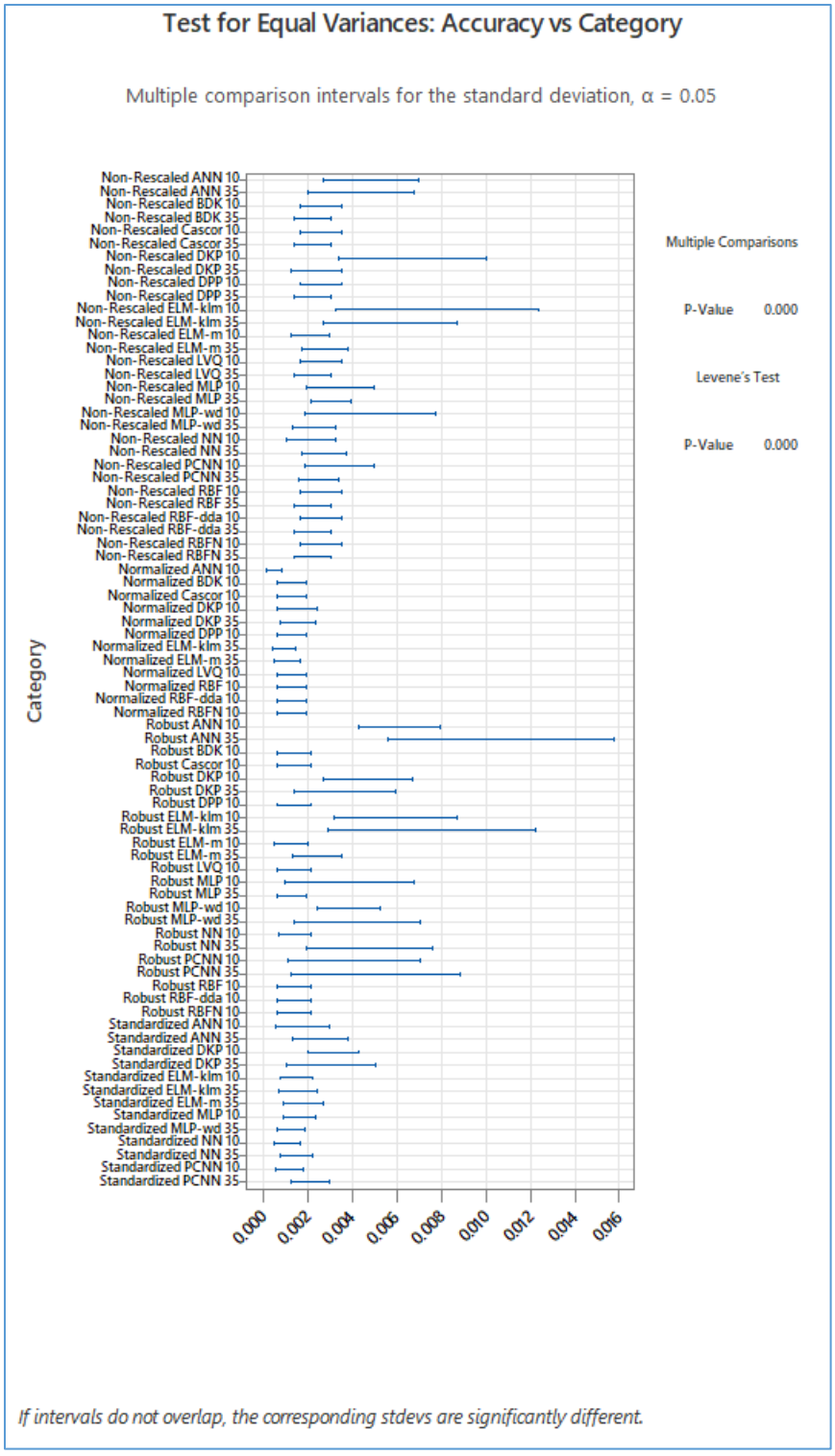

Figure 5: Levene's Test Demonstrating Lack of Homogeneity of Variance (Minitab, 2024).

7. **Normality**: The Normality assumption requires that the dependent variables (i.e. Accuracies) should have approximately normal distributions. The Jarque-Bera (JB) test was used to check if the scaler-model pair groupings were normally distributed or not, and an overwhelming number of model groupings did not have normally distributed results (Glinskiy et al., 2024). Results without normal distributions do not qualify for analysis with ANOVA, and Table 3 shares which categories failed this assumption, given their p-values were less than .05. Table 14 can also be seen in Appendix A, sharing the p-values of categories with normality, but given the lack of homogeneity of variance and given the number of categories without normality, an alternative to ANOVA was needed, and is discussed below.

Table 3: JB P-Value Test Results- Learner values less than 0.05

Model	Scaler Type	Epochs	P-Value
ANN	Normalized	10	0.015048079
ANN	Normalized	35	0.015048079
BDK	Normalized	35	0.01765698
Cascor	Normalized	35	0.01765698
DKP	Normalized	10	0.020416309
DPP	Normalized	35	0.01765698
ELM-klm	Normalized	10	0.028355544
LVQ	Normalized	35	0.01765698
MLP	Normalized	10	0.038594584
MLP	Normalized	35	0.015048079
MLP-wd	Normalized	10	0.015048079
MLP-wd	Normalized	35	0.015048079
NN	Normalized	10	0.028355544
NN	Normalized	35	0.028355544
PCNN	Normalized	10	0.038594584
RBF	Normalized	35	0.01765698
RBF-dda	Normalized	35	0.01765698
RBFN	Normalized	35	0.01765698
MLP	Robust	10	0.004263966
PCNN	Robust	10	0.018177462

Model	Scaler Type	Epochs	P-Value
PCNN	Robust	35	0.021700183
ANN	Standardized	10	0.006927456
BDK	Standardized	10	0.033113938
Cascor	Standardized	10	0.033113938
Cascor	Standardized	35	0.020700267
DPP	Standardized	10	0.033113938
DPP	Standardized	35	0.020700267
LVQ	Standardized	10	0.033113938
LVQ	Standardized	35	0.020700267
MLP	Standardized	35	0.028355544
RBF	Standardized	10	0.033113938
RBF	Standardized	35	0.020700267
RBF-dda	Standardized	10	0.033113938
RBF-dda	Standardized	35	0.020700267
RBFN	Standardized	10	0.033113938
RBFN	Standardized	35	0.020700267

Table 3 summarizes the results obtained on the JB test which shows that the p-values are less than 0.05. These are considered as failed results for the normality test and will provide a justification for the Kruskal-Wallis test discussed in Section 3.5.

3.5 Kruskal-Wallis and Dunns Test

The Kruskal-Wallis (KW) Test is a non-parametric alternative to ANOVA which is used to determine if there are significant differences between the medians of three or more independent groups (Statology, 2023). KW and ANOVA share four of the same assumptions, however KW does not require data to be normally distributed, nor for there to be homogeneity of variance (Laerd Statistics, n.d.; Dinno, 2015; Frost, 2020). The null and alternative hypotheses for Kruskal-Wallis are akin to that of ANOVA, testing for the presence of significant difference in least one category (Statology, 2023). If the KW test determines the presence of a difference, a post-hoc test can be performed to determine which groups are different (Dinno, 2015). The post-hoc test for KW selected for use in

this praxis was Dunn's Test. The Dunn's post hoc test was then used to determine which groups were different, given that Dunn's test has the exact same assumptions as the KW test (Dinno, 2015).

3.6 Experiment Design

All the models were executed in TensorFlow using a Google Collab Pro subscription, a cloud-based Jupyter environment used to write and execute Python 3 code for programing (TensorFlow, 2022). TPU (Tensor Processing Unit), a Google computing backend processing engine, was manually set to v2-8 TPU, which allows Google to dynamically assign the speeds and resources needed to run. These settings remained the same throughout all experiences (Abadi et al., 2015; TensorFlow, 2024a).

As alluded to in Section 3.4.1, regarding the generation of results that were analyzed by the KW and Dunn's tests, the categories were comprised of (a) a rescaling technique, (b) an NN model, and (c) an Epoch count. In effect, the categories of the KW and Dunn's test were triplets, such as *Non-Rescaled ANN 35* to denote a Non-Rescaled ANN with 35 epochs. Each of these categories was tested a total of 21 times, for a total of 2,520 tests. Outliers within each category triplet set of 21 were removed, and any category triplet with fewer than 13 remaining samples was removed from further analysis. The remaining category triplets had their results analyzed, with the subsequent findings shared in Chapter 4.

Chapter 4: Results

4.1 Introduction

This praxis examined the potential for categorizing and detecting Insider Threats using TensorFlow NN to optimize procedures to assist security professionals in selecting appropriate algorithm for making Insider Threat decisions. Over 2,000 tests were run using data from 10 risk identifiers against a preprocessed publicly available CERT dataset to isolate the most effective NN learners.

The methodology described in Chapter 3 introduced 15 Neural Network (NN) learners evaluated using TensorFlow. Performance metrics were collected for analysis. Accuracy was selected as the key component for hypothesis testing, and the Kruskal-Wallis (KW) test was used for testing for the presence of difference in performance (Statology, 2023; Frost, 2020). The KW test did confirm the presence of significant differences among learners, so Dunn's post hoc test was used to determine which learners were different (Dinno, 2015). These findings, among others, are presented in this chapter. A comparison of the various results will reveal the speed of learners in Section 4.2, overall learner performance in Section 4.3, KW and Dunn's test results in Section 4.4, followed by evaluation of hypotheses in Section 4.5.

4.2 Model Speed by Rescaling Technique

Model performance speeds measured in seconds are illustrated in four tables. Table 4 shares the speeds of Standardized learners, Table 5 shares the speeds of Normalized learners, Table 6 shares the speeds of Non-Rescaled learners, and Table 7 shares the speeds of learners with Robust Rescaling. Each of the four tables is ranked by

their speed, with the fastest learners at the top of each table. To be consistent with the KW and Dunn's test results shared in Section 4.4, only the learners that were candidates for analysis have been shared in Table 4 4 through 7.

Table 4: Standardized Speed Rank Order by Epoch 10 and 35

Scaler	Category	Rank	Epochs	Model	Speed (s)
Standardized	Standardized ELM-klm 10	1	10	ELM-klm	146.83
Standardized	Standardized DPP 10	2	10	DPP	149.54
Standardized	Standardized LVQ 10	3	10	LVQ	149.74
Standardized	Standardized Cascor 10	4	10	Cascor	149.75
Standardized	Standardized RBF-dda 10	5	10	RBF-dda	151.81
Standardized	Standardized BDK 10	6	10	BDK	152.28
Standardized	Standardized RBFN 10	7	10	RBFN	155.13
Standardized	Standardized NN 10	8	10	NN	175.56
Standardized	Standardized MLP 10	9	10	MLP	189.43
Standardized	Standardized PCNN 10	10	10	PCNN	201.46
Standardized	Standardized ANN 10	11	10	ANN	251.60
Standardized	Standardized DKP 10	12	10	DKP	340.67
Standardized	Standardized DPP 35	13	35	DPP	356.77
Standardized	Standardized RBF 35	14	35	RBF	363.93
Standardized	Standardized RBF-dda 35	15	35	RBF-dda	367.62
Standardized	Standardized ELM-klm 35	16	35	ELM-klm	368.80
Standardized	Standardized Cascor 35	17	35	Cascor	375.47
Standardized	Standardized RBFN 35	18	35	RBFN	398.64
Standardized	Standardized LVQ 35	19	35	LVQ	402.98
Standardized	Standardized MLP-wd 35	20	35	MLP-wd	423.30
Standardized	Standardized MLP 35	21	35	MLP	447.65
Standardized	Standardized NN 35	22	35	NN	454.88
Standardized	Standardized ELM-m 35	23	35	ELM-m	460.30
Standardized	Standardized PCNN 35	24	35	PCNN	469.65
Standardized	Standardized ANN 35	25	35	ANN	657.71
Standardized	Standardized DKP 35	26	35	DKP	701.29

Table 5: Normalized Speed Rank Order by Epoch 10 and 35

Scaler	Category	Rank	Epochs	Model	Speed (s)
Normalized	Normalized RBFN 10	1	10	RBFN	145.96
Normalized	Normalized RBF-dda 10	2	10	RBF-dda	148.60
Normalized	Normalized BDK 10	3	10	BDK	148.88
Normalized	Normalized LVQ 10	4	10	LVQ	149.09
Normalized	Normalized Cascor 10	5	10	Cascor	151.24
Normalized	Normalized DPP 10	6	10	DPP	151.85
Normalized	Normalized ELM-klm 10	7	10	ELM-klm	153.47
Normalized	Normalized RBF 10	8	10	RBF	157.79
Normalized	Normalized NN 10	9	10	NN	172.92
Normalized	Normalized MLP-wd 10	10	10	MLP-wd	180.25
Normalized	Normalized MLP 10	11	10	MLP	183.55
Normalized	Normalized PCNN 10	12	10	PCNN	187.49

Scaler	Category	Rank	Epochs	Model	Speed (s)
Normalized	Normalized ANN 10	13	10	ANN	251.41
Normalized	Normalized LVQ 35	14	35	LVQ	363.90
Normalized	Normalized RBF 35	15	35	RBF	368.17
Normalized	Normalized BDK 35	16	35	BDK	368.55
Normalized	Normalized DPP 35	17	35	DPP	368.81
Normalized	Normalized Cascor 35	18	35	Cascor	370.79
Normalized	Normalized RBF-dda 35	19	35	RBF-dda	372.38
Normalized	Normalized RBFN 35	20	35	RBFN	376.46
Normalized	Normalized ELM-klm 35	21	35	ELM-klm	376.47
Normalized	Normalized DKP 10	22	10	DKP	388.28
Normalized	Normalized NN 35	23	35	NN	414.60
Normalized	Normalized MLP-wd 35	24	35	MLP-wd	448.70
Normalized	Normalized ELM-m 35	25	35	ELM-m	468.41
Normalized	Normalized MLP 35	26	35	MLP	492.87
Normalized	Normalized ANN 35	27	35	ANN	645.71
Normalized	Normalized DKP 35	28	35	DKP	733.52

Table 6: Non-Rescaled Speed Rank Order by Epoch 10 and 35

Scaler	Category	Rank	Epochs	Model	Speed (s)
Non-Rescaled	Non-Rescaled RBFN 10	1	10	RBFN	148.43
Non-Rescaled	Non-Rescaled RBF 10	2	10	RBF	149.98
Non-Rescaled	Non-Rescaled LVQ 10	3	10	LVQ	150.15
Non-Rescaled	Non-Rescaled Cascor 10	4	10	Cascor	154.38
Non-Rescaled	Non-Rescaled BDK 10	5	10	BDK	154.63
Non-Rescaled	Non-Rescaled ELM-klm 10	6	10	ELM-klm	154.70
Non-Rescaled	Non-Rescaled RBF-dda 10	7	10	RBF-dda	155.08
Non-Rescaled	Non-Rescaled NN 10	8	10	NN	178.45
Non-Rescaled	Non-Rescaled MLP-wd 10	9	10	MLP-wd	181.79
Non-Rescaled	Non-Rescaled PCNN 10	10	10	PCNN	182.79
Non-Rescaled	Non-Rescaled MLP 10	11	10	MLP	187.03
Non-Rescaled	Non-Rescaled ELM-m 10	12	10	ELM-m	195.91
Non-Rescaled	Non-Rescaled DKP 10	13	10	DKP	342.26
Non-Rescaled	Non-Rescaled LVQ 35	14	35	LVQ	365.92
Non-Rescaled	Non-Rescaled RBF 35	15	35	RBF	366.60
Non-Rescaled	Non-Rescaled RBFN 35	16	35	RBFN	368.43
Non-Rescaled	Non-Rescaled RBF-dda 35	17	35	RBF-dda	370.35
Non-Rescaled	Non-Rescaled ELM-klm 35	18	35	ELM-klm	371.35
Non-Rescaled	Non-Rescaled DPP 35	19	35	DPP	374.34
Non-Rescaled	Non-Rescaled BDK 35	20	35	BDK	375.59
Non-Rescaled	Non-Rescaled Cascor 35	21	35	Cascor	380.06
Non-Rescaled	Non-Rescaled NN 35	22	35	NN	412.83
Non-Rescaled	Non-Rescaled MLP-wd 35	23	35	MLP-wd	418.11
Non-Rescaled	Non-Rescaled PCNN 35	24	35	PCNN	442.60
Non-Rescaled	Non-Rescaled MLP 35	25	35	MLP	446.90
Non-Rescaled	Non-Rescaled ELM-m 35	26	35	ELM-m	467.49
Non-Rescaled	Non-Rescaled DPP 10	27	10	DPP	552.94
Non-Rescaled	Non-Rescaled ANN 35	28	35	ANN	661.94
Non-Rescaled	Non-Rescaled DKP 35	29	35	DKP	674.12

Table 7: Robust Speed Rank Order by Epoch 10 and 35

Scaler	Category	Rank	Epochs	Model	Speed (s)
Robust	Robust RBF-dda 10	1	10	RBF-dda	149.47
Robust	Robust RBFN 10	2	10	RBFN	149.96
Robust	Robust ELM-klm 10	3	10	ELM-klm	152.19
Robust	Robust BDK 10	4	10	BDK	152.37
Robust	Robust DPP 10	5	10	DPP	153.42
Robust	Robust Cascor 10	6	10	Cascor	154.45
Robust	Robust RBF 10	7	10	RBF	154.60
Robust	Robust RBF 10	8	10	RBF	160.04
Robust	Robust LVQ 10	9	10	LVQ	161.01
Robust	Robust NN 10	10	10	NN	181.01
Robust	Robust MLP-wd 10	11	10	MLP-wd	185.09
Robust	Robust MLP 10	12	10	MLP	207.31
Robust	Robust ELM-m 10	13	10	ELM-m	211.08
Robust	Robust PCNN 10	14	10	PCNN	234.14
Robust	Robust ANN 10	15	10	ANN	250.67
Robust	Robust DKP 10	16	10	DKP	353.69
Robust	Robust ELM-klm 35	17	35	ELM-klm	378.12
Robust	Robust NN 35	18	35	NN	419.70
Robust	Robust MLP 35	19	35	MLP	427.76
Robust	Robust PCNN 35	20	35	PCNN	446.68
Robust	Robust MLP-wd 35	21	35	MLP-wd	469.04
Robust	Robust ELM-m 35	22	35	ELM-m	495.34
Robust	Robust ANN 35	23	35	ANN	675.39
Robust	Robust DKP 35	24	35	DKP	763.95

The results for Table 4 - Table 7 were presented for all the learners which were evaluated for KW and Dunn's test. These learners were evaluated for 10 and 35 Epoch. The category column represents the summary of the combination of (scaler, model and Epoch triplets). The shortest runtimes by rescaling technique were ranked as follows, Standardized ELM-klm 10, Normalized RBFN 10, Non-Rescaled RBFN 10 and Robust RBF-dda 10.

4.3 Model Accuracy by Rescaling Technique

To be consistent with the KW and Dunn's test results shared in Section 4.4, only the learners that were candidates for analysis have been shared in Table 8 - Table 11, which share the rank ordered accuracy for learners that were Standardized, Normalized, Non-Rescaled, or given Robust Rescaling respectively. As mentioned previously the

Category column in Table 8- Table 11 represents the (Scaler, Model, Epochs Triplets). In addition to Tables sharing the Accuracies, bar plots for each corresponding table have been included.

Table 8: Standardized Accuracy Rank Order

Scaler	Category	Rank	Epochs	Model	Mean Accuracy
Standardized	Standardized DKP 35	1	35	DKP	0.978992
Standardized	Standardized NN 35	2	35	NN	0.977591
Standardized	Standardized BDK 10	3	10	BDK	0.976190
Standardized	Standardized Cascor 10	4	10	Cascor	0.976190
Standardized	Standardized Cascor 35	5	35	Cascor	0.976190
Standardized	Standardized DKP 10	6	10	DKP	0.976190
Standardized	Standardized DPP 10	7	10	DPP	0.976190
Standardized	Standardized DPP 35	8	35	DPP	0.976190
Standardized	Standardized LVQ 10	9	10	LVQ	0.976190
Standardized	Standardized LVQ 35	10	35	LVQ	0.976190
Standardized	Standardized MLP 10	11	10	MLP	0.976190
Standardized	Standardized MLP 35	12	35	MLP	0.976190
Standardized	Standardized NN 10	13	10	NN	0.976190
Standardized	Standardized PCNN 10	14	10	PCNN	0.976190
Standardized	Standardized RBF 35	15	35	RBF	0.976190
Standardized	Standardized RBF-dda 10	16	10	RBF-dda	0.976190
Standardized	Standardized RBF-dda 35	17	35	RBF-dda	0.976190
Standardized	Standardized RBFN 10	18	10	RBFN	0.976190
Standardized	Standardized RBFN 35	19	35	RBFN	0.976190
Standardized	Standardized ANN 10	20	10	ANN	0.974790
Standardized	Standardized ANN 35	21	35	ANN	0.974790
Standardized	Standardized ELM-klm 10	22	10	ELM-klm	0.974790
Standardized	Standardized ELM-m 35	23	35	ELM-m	0.974790
Standardized	Standardized MLP-wd 35	24	35	MLP-wd	0.974790
Standardized	Standardized PCNN 35	25	35	PCNN	0.974790
Standardized	Standardized ELM-klm 35	26	35	ELM-klm	0.973389

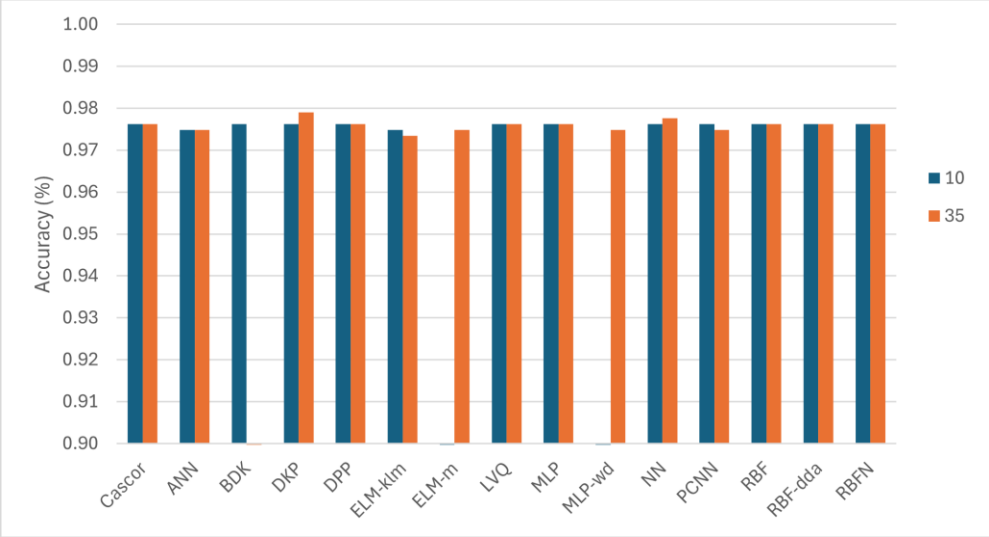

Figure 6: Standardized Accuracy Bar Plot

The summary, Table 8, and the corresponding Bar Plot in Figure 6, illustrated the results for all learners. The results demonstrate Standardized DPK 35 ranked number one with an accuracy of 0.978992.

Table 9: Normalized Accuracy Rank Order

Scaler	Category	Rank	Epochs	Model	Mean Accuracy
Normalized	Normalized NN 10	1	10	NN	0.976190
Normalized	Normalized MLP 10	2	10	MLP	0.976190
Normalized	Normalized PCNN 10	3	10	PCNN	0.976190
Normalized	Normalized NN 35	4	35	NN	0.976190
Normalized	Normalized MLP 35	5	35	MLP	0.976190
Normalized	Normalized DKP 35	6	35	DKP	0.976190
Normalized	Normalized RBFN 10	7	10	RBFN	0.974790
Normalized	Normalized RBF-dda 10	8	10	RBF-dda	0.974790
Normalized	Normalized BDK 10	9	10	BDK	0.974790
Normalized	Normalized LVQ 10	10	10	LVQ	0.974790
Normalized	Normalized Cascor 10	11	10	Cascor	0.974790
Normalized	Normalized DPP 10	12	10	DPP	0.974790
Normalized	Normalized RBF 10	13	10	RBF	0.974790
Normalized	Normalized LVQ 35	14	35	LVQ	0.974790
Normalized	Normalized RBF 35	15	35	RBF	0.974790
Normalized	Normalized BDK 35	16	35	BDK	0.974790
Normalized	Normalized DPP 35	17	35	DPP	0.974790
Normalized	Normalized Cascor 35	18	35	Cascor	0.974790
Normalized	Normalized RBF-dda 35	19	35	RBF-dda	0.974790

Scaler	Category	Rank	Epochs	Model	Mean Accuracy
Normalized	Normalized RBFN 35	20	35	RBFN	0.974790
Normalized	Normalized ELM-klm 35	21	35	ELM-klm	0.974790
Normalized	Normalized ELM-m 35	22	35	ELM-m	0.974790
Normalized	Normalized ELM-klm 10	23	10	ELM-klm	0.973389
Normalized	Normalized DKP 10	24	10	DKP	0.970588
Normalized	Normalized MLP-wd 10	25	10	MLP-wd	0.966387
Normalized	Normalized ANN 10	26	10	ANN	0.966387
Normalized	Normalized MLP-wd 35	27	35	MLP-wd	0.966387
Normalized	Normalized ANN 35	28	35	ANN	0.966387

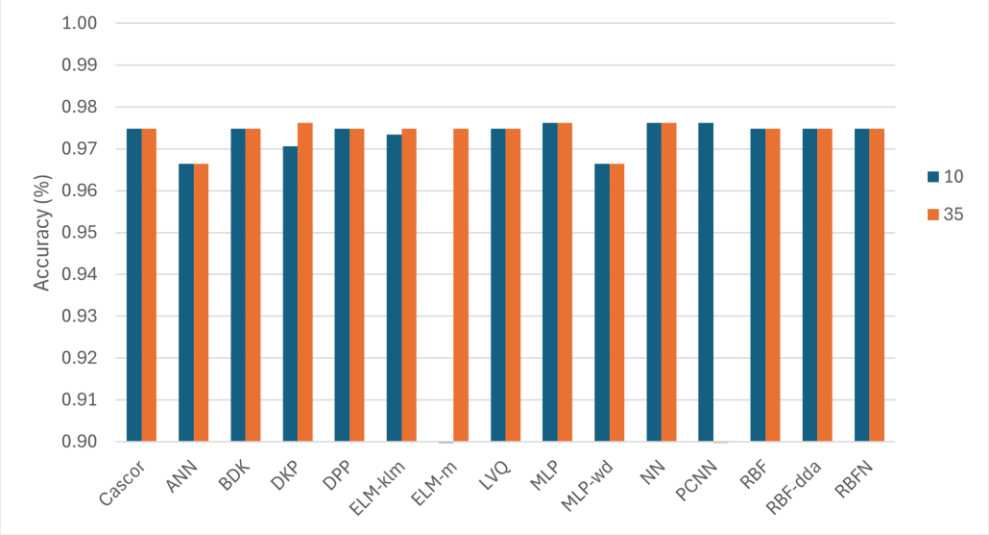

Figure 7: Normalized Accuracy Bar Plot

The summary, Table 9, and the corresponding Bar Plot in Figure 7, illustrate the results for all learners. The results demonstrate Normalized NN 10 ranked number one with a mean Accuracy of 0.976190.

Table 10: Non-Rescaled Accuracy Rank Order

Scaler	Category	Rank	Epochs	Model	Mean Accuracy
Non-Rescaled	Non-Rescaled ELM-klm 10	1	10	ELM-klm	0.980392
Non-Rescaled	Non-Rescaled MLP 35	2	35	MLP	0.977591
Non-Rescaled	Non-Rescaled MLP 10	3	10	MLP	0.976190
Non-Rescaled	Non-Rescaled PCNN 10	4	10	PCNN	0.974790
Non-Rescaled	Non-Rescaled DKP 10	5	10	DKP	0.974790
Non-Rescaled	Non-Rescaled RBFN 10	6	10	RBFN	0.973389
Non-Rescaled	Non-Rescaled RBF 10	7	10	RBF	0.973389
Non-Rescaled	Non-Rescaled LVQ 10	8	10	LVQ	0.973389

Scaler	Category	Rank	Epochs	Model	Mean Accuracy
Non-Rescaled	Non-Rescaled Cascor 10	9	10	Cascor	0.973389
Non-Rescaled	Non-Rescaled BDK 10	10	10	BDK	0.973389
Non-Rescaled	Non-Rescaled RBF-dda 10	11	10	RBF-dda	0.973389
Non-Rescaled	Non-Rescaled MLP-wd 10	12	10	MLP-wd	0.973389
Non-Rescaled	Non-Rescaled NN 35	13	35	NN	0.973389
Non-Rescaled	Non-Rescaled DPP 10	14	10	DPP	0.973389
Non-Rescaled	Non-Rescaled NN 10	15	10	NN	0.971989
Non-Rescaled	Non-Rescaled ELM-m 10	16	10	ELM-m	0.971989
Non-Rescaled	Non-Rescaled LVQ 35	17	35	LVQ	0.971989
Non-Rescaled	Non-Rescaled RBF 35	18	35	RBF	0.971989
Non-Rescaled	Non-Rescaled RBFN 35	19	35	RBFN	0.971989
Non-Rescaled	Non-Rescaled RBF-dda 35	20	35	RBF-dda	0.971989
Non-Rescaled	Non-Rescaled DPP 35	21	35	DPP	0.971989
Non-Rescaled	Non-Rescaled BDK 35	22	35	BDK	0.971989
Non-Rescaled	Non-Rescaled Cascor 35	23	35	Cascor	0.971989
Non-Rescaled	Non-Rescaled MLP-wd 35	24	35	MLP-wd	0.971989
Non-Rescaled	Non-Rescaled ELM-klm 35	25	35	ELM-klm	0.970588
Non-Rescaled	Non-Rescaled PCNN 35	26	35	PCNN	0.969188
Non-Rescaled	Non-Rescaled ELM-m 35	27	35	ELM-m	0.969188
Non-Rescaled	Non-Rescaled DKP 35	28	35	DKP	0.969188
Non-Rescaled	Non-Rescaled ANN 10	29	10	ANN	0.966387
Non-Rescaled	Non-Rescaled ANN 35	30	35	ANN	0.963585

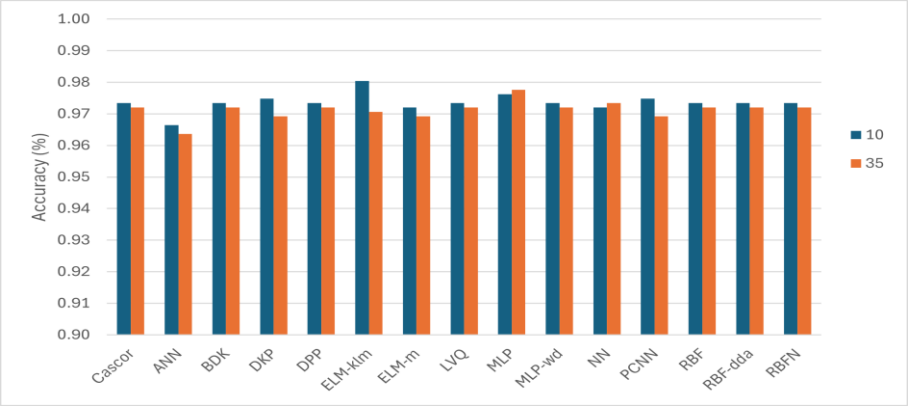

Figure 8: Non-Rescaled Accuracy Bar Plot

The summary, Table 10, and the corresponding Bar Plot in Figure 8, illustrate the results for all learners. The results demonstrate Non-Rescaled ELM-klm 10 ranked number one with a mean Accuracy of 0.980392.

Table 11: Robust Accuracy Rank Order

Scaler	Category	Rank	Epochs	Model	Mean Accuracy
Robust	Robust RBF-dda 10	1	10	RBF-dda	0.978992
Robust	Robust RBFN 10	2	10	RBFN	0.978992
Robust	Robust ELM-klm 10	3	10	ELM-klm	0.978992
Robust	Robust BDK 10	4	10	BDK	0.978992
Robust	Robust DPP 10	5	10	DPP	0.978992
Robust	Robust Cascor 10	6	10	Cascor	0.978992
Robust	Robust RBF 10	7	10	RBF	0.978992
Robust	Robust LVQ 10	8	10	LVQ	0.978992
Robust	Robust MLP 10	9	10	MLP	0.978992
Robust	Robust NN 10	10	10	NN	0.977591
Robust	Robust MLP-wd 10	11	10	MLP-wd	0.977591
Robust	Robust ELM-m 10	12	10	ELM-m	0.977591
Robust	Robust PCNN 10	13	10	PCNN	0.977591
Robust	Robust NN 35	14	35	NN	0.977591
Robust	Robust MLP 35	15	35	MLP	0.977591
Robust	Robust PCNN 35	16	35	PCNN	0.977591
Robust	Robust ELM-m 35	17	35	ELM-m	0.977591
Robust	Robust RBF 10	18	10	RBF	0.976190
Robust	Robust ELM-klm 35	19	35	ELM-klm	0.970588
Robust	Robust DKP 35	20	35	DKP	0.970588
Robust	Robust MLP-wd 35	21	35	MLP-wd	0.969188
Robust	Robust ANN 10	22	10	ANN	0.966387
Robust	Robust DKP 10	23	10	DKP	0.964986
Robust	Robust ANN 35	24	35	ANN	0.953782

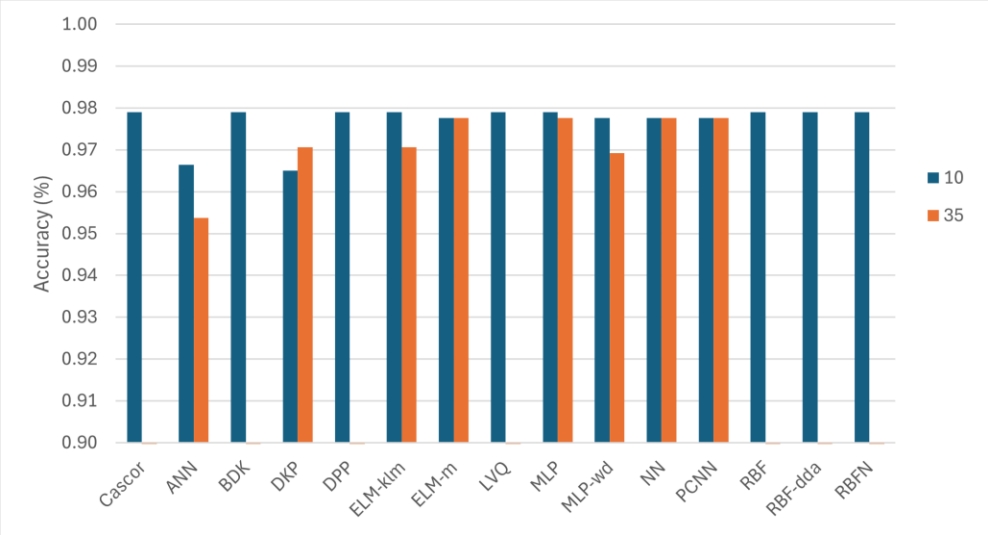

Figure 9: Robust Accuracy Comparison by Model and Epoch

The summary, Table 11, and the corresponding Bar Plot in Figure 9, illustrate the results for all learners. The results demonstrate Robust RBF-dda 10 ranked number one with a mean Accuracy of 0.978992.

4.4 Kruskal-Wallis Analysis

With the assumptions of KW test met, Minitab, a statistical hypothesis testing software, was used to test for the presence of significant difference between groups. (Laerd Statistics, n.d.). As shown in Figure 10, given the p-value for the KW test was less than 0.05, there was a difference present between groups, so the Dunn's post hoc test was warranted (Dinno, 2015). Looking at Table 12, the rank ordered medians are shared, according to best performance, as used in the KW test.

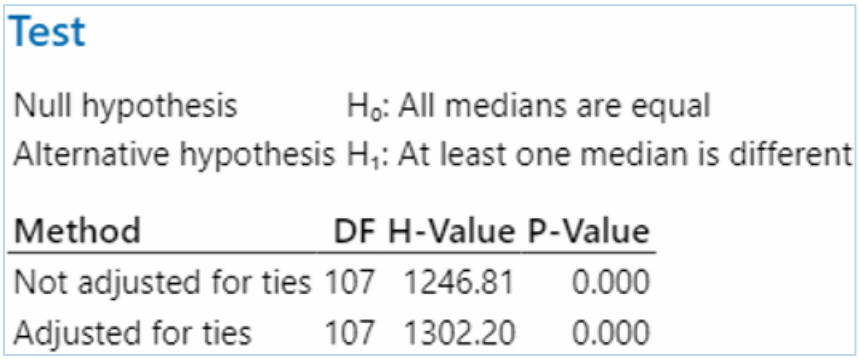

Figure 10: Kruskal-Wallis Significance

Table 12: Kruskal-Wallis Median Rank Order

Category	N	Median	Median Rank Order	Mean Rank	Z-Value
Robust MLP 35	18	0.978992	1	1995.7	6.69
Non-Rescaled ELM-klm 35	20	0.977591	2	1325	2.07
Robust BDK 10	17	0.977591	2	1683	4.36
Robust Cascor 10	17	0.977591	2	1683	4.36
Robust DPP 10	17	0.977591	2	1683	4.36
Robust ELM-m 10	16	0.977591	2	1946.6	5.98
Robust LVQ 10	17	0.977591	2	1683	4.36
Robust MLP 10	19	0.977591	2	1580.8	3.87
Robust NN 10	19	0.977591	2	1743.4	5.05

Category	N	Median	Median Rank Order	Mean Rank	Z-Value
Robust PCNN 10	19	0.977591	2	1635.8	4.27
Robust RBF 10	17	0.977591	2	1683	4.36
Robust RBF-dda 10	17	0.977591	2	1683	4.36
Robust RBFN 10	17	0.977591	2	1683	4.36
Non-Rescaled BDK 35	21	0.97619	14	1192.4	1.11
Non-Rescaled Cascor 35	21	0.97619	14	1192.4	1.11
Non-Rescaled DPP 35	21	0.97619	14	1192.4	1.11
Non-Rescaled ELM-klm 10	21	0.97619	14	1216.4	1.29
Non-Rescaled LVQ 35	21	0.97619	14	1192.4	1.11
Non-Rescaled RBF 35	21	0.97619	14	1192.4	1.11
Non-Rescaled RBF-dda 35	21	0.97619	14	1192.4	1.11
Non-Rescaled RBFN 35	21	0.97619	14	1192.4	1.11
Normalized DKP 35	16	0.97619	14	1244	1.31
Normalized ELM-m 35	21	0.97619	14	1233.7	1.42
Normalized MLP 10	15	0.97619	14	1448.5	2.58
Normalized MLP 35	21	0.97619	14	1448.5	3.06
Normalized NN 10	17	0.97619	14	1448.5	2.75
Normalized NN 35	17	0.97619	14	1448.5	2.75
Normalized PCNN 10	15	0.97619	14	1448.5	2.58
Robust ELM-m 35	19	0.97619	14	1368.8	2.33
Robust MLP-wd 35	20	0.97619	14	1299.8	1.88
Robust PCNN 35	19	0.97619	14	1342.1	2.14
Standardized ANN 35	21	0.97619	14	1322.1	2.1
Standardized BDK 10	16	0.97619	14	1448.5	2.67
Standardized Cascor 10	16	0.97619	14	1448.5	2.67
Standardized Cascor 35	19	0.97619	14	1448.5	2.91
Standardized DKP 35	20	0.97619	14	1483.4	3.25
Standardized DPP 10	16	0.97619	14	1448.5	2.67
Standardized DPP 35	19	0.97619	14	1448.5	2.91
Standardized LVQ 10	16	0.97619	14	1448.5	2.67
Standardized LVQ 35	19	0.97619	14	1448.5	2.91
Standardized MLP 10	21	0.97619	14	1571	3.99
Standardized MLP 35	17	0.97619	14	1448.5	2.75
Standardized NN 10	20	0.97619	14	1277.9	1.72
Standardized NN 35	19	0.97619	14	1270.4	1.62
Standardized PCNN 10	19	0.97619	14	1321	1.99
Standardized RBF 10	16	0.97619	14	1448.5	2.67
Standardized RBF 35	19	0.97619	14	1448.5	2.91
Standardized RBF-dda 10	16	0.97619	14	1448.5	2.67
Standardized RBF-dda 35	19	0.97619	14	1448.5	2.91
Standardized RBFN 10	16	0.97619	14	1448.5	2.67
Standardized RBFN 35	19	0.97619	14	1448.5	2.91
Normalized BDK 10	20	0.97549	52	1217.3	1.27
Normalized Cascor 10	20	0.97549	52	1217.3	1.27
Normalized DPP 10	20	0.97549	52	1217.3	1.27
Normalized LVQ 10	20	0.97549	52	1217.3	1.27
Normalized RBF 10	20	0.97549	52	1217.3	1.27
Normalized RBF-dda 10	20	0.97549	52	1217.3	1.27
Normalized RBFN 10	20	0.97549	52	1217.3	1.27
Non-Rescaled MLP 35	21	0.97479	59	1086.5	0.3
Normalized BDK 35	20	0.97479	59	898.5	-1.1
Normalized Cascor 35	20	0.97479	59	898.5	-1.1

Category	N	Median	Median Rank Order	Mean Rank	Z-Value
Normalized DPP 35	20	0.97479	59	898.5	-1.1
Normalized ELM-klm 35	21	0.97479	59	1081.8	0.27
Normalized LVQ 35	20	0.97479	59	898.5	-1.1
Normalized RBF 35	20	0.97479	59	898.5	-1.1
Normalized RBF-dda 35	20	0.97479	59	898.5	-1.1
Normalized RBFN 35	20	0.97479	59	898.5	-1.1
Robust ELM-klm 10	21	0.97479	59	1164.9	0.9
Robust MLP-wd 10	18	0.97479	59	1026.7	-0.14
Robust NN 35	21	0.97479	59	1145.4	0.75
Standardized ANN 10	20	0.97479	59	1142.4	0.71
Standardized DKP 10	21	0.97479	59	993.7	-0.41
Standardized ELM-klm 10	20	0.97479	59	1196.8	1.11
Standardized ELM-m 35	19	0.97479	59	1049.6	0.02
Standardized MLP-wd 35	21	0.97479	59	1167.1	0.92
Standardized PCNN 35	21	0.97479	59	1043.2	-0.03
Non-Rescaled BDK 10	21	0.973389	77	767.8	-2.13
Non-Rescaled Cascor 10	21	0.973389	77	767.8	-2.13
Non-Rescaled DPP 10	21	0.973389	77	767.8	-2.13
Non-Rescaled LVQ 10	21	0.973389	77	767.8	-2.13
Non-Rescaled PCNN 35	21	0.973389	77	529.1	-3.95
Non-Rescaled RBF 10	21	0.973389	77	767.8	-2.13
Non-Rescaled RBF-dda 10	21	0.973389	77	767.8	-2.13
Non-Rescaled RBFN 10	21	0.973389	77	767.8	-2.13
Normalized ELM-klm 10	17	0.973389	77	600	-3.06
Robust ELM-klm 35	21	0.973389	77	1012	-0.27
Standardized ELM-klm 35	17	0.973389	77	530.6	-3.54
Non-Rescaled ELM-m 35	21	0.971989	88	517.6	-4.03
Non-Rescaled NN 10	19	0.971989	88	558.3	-3.54
Non-Rescaled NN 35	20	0.971989	88	580.4	-3.47
Non-Rescaled PCNN 10	21	0.971989	88	654.3	-2.99
Robust DKP 35	17	0.971989	88	491.3	-3.81
Non-Rescaled DKP 35	19	0.970588	93	376.1	-4.86
Non-Rescaled ELM-m 10	19	0.970588	93	375.7	-4.86
Non-Rescaled MLP 10	21	0.970588	93	428.3	-4.72
Non-Rescaled MLP-wd 35	19	0.970588	93	326.8	-5.22
Normalized DKP 10	19	0.970588	93	314.8	-5.3
Non-Rescaled MLP-wd 10	21	0.969188	98	347.3	-5.33
Non-Rescaled DKP 10	21	0.966387	99	196	-6.49
Normalized ANN 10	21	0.966387	99	133.6	-6.96
Normalized ANN 35	21	0.966387	99	130	-6.99
Normalized MLP-wd 10	21	0.966387	99	130	-6.99
Normalized MLP-wd 35	21	0.966387	99	130	-6.99
Robust ANN 10	21	0.966387	99	154.3	-6.8
Robust ANN 35	21	0.966387	99	388.5	-5.02
Non-Rescaled ANN 10	21	0.964986	106	103	-7.19
Non-Rescaled ANN 35	20	0.964986	106	144	-6.71
Robust DKP 10	21	0.964986	106	100.3	-7.21

4.4.1 Dunn's Test Lettered Groupings

Given that the KW test indicated significant difference, the Dunn's test was performed with Python and subsequently compiled into lettered groupings in Excel. In total, 26 lettered groups were produced. For any given lettered group, there was no determinable significant difference according to Dunn's test, with the best performing lettered group being group A, followed by B, then C, and so on. Figure 11, Figure 12, and Figure 13 share the total findings.

Rescaling Type	Learner Category	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	Y	X	Z
Robust	Robust MLP 35	A																								
Robust	Robust ELM-m 10	A	B																							
Robust	Robust NN 10	A	B	C																						
Robust	Robust PCNN 10	A	B	C	D																					
Robust	Robust BDK 10	A	B	C	D	E																				
Robust	Robust Cascor 10	A	B	C	D	E																				
Robust	Robust DPP 10	A	B	C	D	E																				
Robust	Robust LVQ 10	A	B	C	D	E																				
Robust	Robust RBF 10	A	B	C	D	E																				
Robust	Robust RBF-dda 10	A	B	C	D	E																				
Robust	Robust RBFN 10	A	B	C	D	E																				
Standardized	Standardized MLP 10	A	B	C	D	E	F																			
Standardized	Standardized DKP 35	A	B	C	D	E	F																			
Robust	Robust MLP 10	A	B	C	D	E	F																			
Standardized	Standardized LVQ 35	A	B	C	D	E	F	G	H																	
Standardized	Standardized RBF-dda 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized RBFN 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized RBF 10	A	B	C	D	E	F	G	H	I																
Normalized	Normalized MLP 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized LVQ 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized DPP 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized Cascor 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized BDK 10	A	B	C	D	E	F	G	H	I																
Standardized	Standardized Cascor 35	A	B	C	D	E	F	G	H	I																
Standardized	Standardized DPP 35	A	B	C	D	E	F	G	H	I																
Normalized	Normalized MLP 35	A	B	C	D	E	F	G	H	I																
Standardized	Standardized RBF 35	A	B	C	D	E	F	G	H	I																
Standardized	Standardized RBF-dda 35	A	B	C	D	E	F	G	H	I																
Standardized	Standardized RBFN 35	A	B	C	D	E	F	G	H	I																
Normalized	Normalized PCNN 10	A	B	C	D	E	F	G	H	I																
Normalized	Normalized NN 35	A	B	C	D	E	F	G	H	I	K															
Normalized	Normalized NN 10	A	B	C	D	E	F	G	H	I	K															
Standardized	Standardized MLP 35	A	B	C	D	E	F	G	H	I	K															
Standardized	Standardized ANN 35	A	B	C	D	E	F	G	H	I	K	L														
Robust	Robust ELM-m 35	A	B	C	D	E	F	G	H	I	K	L	M													
Standardized	Standardized PCNN 10	A	B	C	D	E	F	G	H	I	K	L	M	N												
Standardized	Standardized NN 10	A	B	C	D	E	F	G	H	I	K	L	M	N												
Standardized	Standardized NN 35	A	B	C	D	E	F	G	H	I	K	L	M	N												
Non-Rescaled	Non-Rescaled ELM-klm 35	A	B	C	D	E	F	G	H	I	K	L	M	N	O											
Normalized	Normalized DKP 35	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P										
Normalized	Normalized ELM-m 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Normalized	Normalized BDK 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									

Figure 11: Dunn's Test Lettered Groupings, One of Three

Rescaling Type	Learner Category	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	Y	X	Z
Normalized	Normalized Cascor 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Normalized	Normalized DPP 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Normalized	Normalized LVQ 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Normalized	Normalized RBF 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Normalized	Normalized RBF-dda 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Normalized	Normalized RBFN 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Robust	Robust PCNN 35	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Standardized	Standardized ELM-klm 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Standardized	Standardized MLP-wd 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Robust	Robust MLP-wd 35	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Non-Rescaled	Non-Rescaled LVQ 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Non-Rescaled	Non-Rescaled RBFN 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Non-Rescaled	Non-Rescaled RBF 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Non-Rescaled	Non-Rescaled RBF-dda 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Non-Rescaled	Non-Rescaled DPP 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q									
Non-Rescaled	Non-Rescaled Cascor 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R								
Non-Rescaled	Non-Rescaled BDK 35		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S							
Standardized	Standardized ANN 10			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T						
Normalized	Normalized ELM-klm 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T						
Standardized	Standardized ELM-m 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T						
Normalized	Normalized LVQ 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Normalized	Normalized RBFN 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Normalized	Normalized RBF-dda 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Normalized	Normalized RBF 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Normalized	Normalized DPP 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Normalized	Normalized Cascor 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Normalized	Normalized BDK 35				D		F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Standardized	Standardized PCNN 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Non-Rescaled	Non-Rescaled MLP 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U					
Robust	Robust ELM-klm 10			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V				
Robust	Robust NN 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V				
Non-Rescaled	Non-Rescaled ELM-klm 10		B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W			
Standardized	Standardized DKP 10				D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X		
Robust	Robust MLP-wd 10			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X		
Robust	Robust ELM-klm 35			C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	
Non-Rescaled	Non-Rescaled LVQ 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled RBFN 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled RBF-dda 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled RBF 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled DPP 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Figure 12: Dunn's Test Lettered Groupings, Two of Three

Rescaling Type	Learner Category	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	Y	X	Z
Non-Rescaled	Non-Rescaled Cascor 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled BDK 10							G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Normalized	Normalized ELM-klm 10									I		L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Standardized	Standardized ELM-klm 35											L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled NN 10											L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled PCNN 10									I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled NN 35											L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled PCNN 35											L		N		P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled ELM-m 35													N		P	Q	R	S	T	U	V	W	X	Y	Z
Robust	Robust DKP 35											L		N		P	Q	R	S	T	U	V	W	X	Y	Z
Non-Rescaled	Non-Rescaled ELM-m 10															P		R		T	U	V		X	Y	Z
Non-Rescaled	Non-Rescaled DKP 35																	R		T	U	V		X	Y	Z
Non-Rescaled	Non-Rescaled MLP 10															P	Q	R		T	U	V	W	X	Y	Z
Normalized	Normalized DKP 10																			T	U			X	Y	Z
Non-Rescaled	Non-Rescaled MLP-wd 35																			T	U			X	Y	Z
Non-Rescaled	Non-Rescaled MLP-wd 10																			T	U	V		X	Y	Z
Normalized	Normalized ANN 35																				U					Z
Normalized	Normalized MLP-wd 10																				U					Z
Normalized	Normalized MLP-wd 35																				U					Z
Non-Rescaled	Non-Rescaled ANN 35																				U			X		Z
Non-Rescaled	Non-Rescaled DKP 10																				U			X	Y	Z
Robust	Robust ANN 10																				U					Z
Robust	Robust ANN 35																	R		T	U	V		X	Y	Z
Robust	Robust DKP 10																				U					Z
Non-Rescaled	Non-Rescaled ANN 10																				U					Z

Figure 13: Dunn's Test Lettered Groupings, Three of Three

4.5 Addressing Research Questions and Hypotheses Outcomes

A series of four hypotheses was presented in Chapter 1 to test whether or not use of TensorFlow NN could produce better results when various are applied. The hypotheses are divided into sets highlighting performance differences, quantifying if the results are superior to the top seven performers identified in Dr. Noever's research (Noever, 2019) and if data rescaling techniques improved performance, and whether or not epoch counts had a significant bearing on results. Below the results are presented alongside their corresponding hypothesis.

4.5.1 Research Question and Hypothesis, Set 1

RQ_1: Use TensorFlow to detect if there are differences in performance between the NN models created to detect insider threats?

$H_{1\text{-}n}$: Of NN models created to detect insider threats, there is no significant difference between their performance.

$H_{1\text{-}a}$: Of NN models created to detect insider threats, there is at least one model with a significant difference between their performance as determined by a P-Value threshold.

The KW test as shown in Figure 10 indicates a p-value of less than 0.05, which provides evidence that there is at least one model with a significant difference in performance. As such, the null hypothesis of this set ($H_{1\text{-}n}$) was rejected, and the alternative hypothesis of this set ($H_{1\text{-}a}$) was accepted. If there was no significant difference in performance between models, then the p-value for KW would be greater than 0.05, which was not the case.

4.5.2 Research Question and Hypothesis, Set 2

RQ_2: Use TensorFlow NN to identify if the tested NN models are better at detecting Insider Threats than the top seven best performing models from *Classifier Suites for Insider Threat Detection* (Noever, 2019).

$H_{2\text{-}n}$: Of NN models created to detect insider threats, no models of this praxis performed better than the top seven models from "Classifier Suites for Insider Threat Detection" (Noever, 2019).

$H_{2\text{-}a}$ Of NN models created to detect insider threats, at least one model of this praxis performed better than the top seven models from *Classifier Suites for Insider Threat Detection* (Noever, 2019).

Table 13: Top 7 Over-sampled Result by Model Accuracy
(Noever, 2019)

Model	Rank	Mean Accuracy	Execution Time (s)
Random Forest	1	0.9835	210.56
Random Forest by Randomization	2	0.9831	200.16
K-Nearest Neighbors	3	0.9814	4.44
Localized Linear Discriminant Analysis	4	0.9806	114.64
Regularized Random Forest	5	0.975	475.19
Bagged CART	6	0.9727	18.66
Logistic Model Trees	7	0.9683	49.61

According to results shared in Table 13, Dr. Noever's top performer achieved an accuracy of 0.9835. In contrast, the results obtained during this praxis show that the top performing model was ELM-klm non-rescaled, which achieved an accuracy of 0.980392 (see Table 10). These results represent a relatively small difference in performance; however, it is evident that this value did not outperform best performer of the Top 7 models tested by Dr. Noever. As such, the null hypothesis ($H_{2\text{-}n}$) was not rejected, and the alternative hypothesis of ($H_{2\text{-}a}$) was rejected.

4.5.3 Research Question and Hypothesis, Set 3

RQ_3: Are there significant differences between data rescaling techniques used in NN models for Insider Threat detection?

$H_{3\text{-}n}$: Of NN models created to detect insider threats, there is no significant difference among different data rescaling techniques.

$H_{3\text{-}a}$: Of NN models created to detect insider threats, there is a significant difference among different data rescaling techniques.

If it were true that rescaling techniques do not make a difference to model performance, then the rescaling techniques of any given Learner-Epoch set would exist within the same Dunn's group. By way of example, Robust MLP 35 was in group A, Normalized MLP 35 was in group A, Standardized MLP 35 was in group A, whereas Non-Rescaled MLP 35 was in group C. If rescaling made no difference, each of these four would have been in the same Dunn's group. That was not the case.

Similar discrepancies can be found among other learners. The evidence is illustrated in Figure 11-Figure 13 of Dunn's Test Letter Groupings. For this reason, the null hypothesis ($H_{3\text{-}n}$) was rejected, and the alternative hypothesis ($H_{3\text{-}a}$) was accepted. According to Dunn's groupings, rescaling did make a difference in performance, however, there was no trend suggesting that a single rescaling technique consistently produced top performing models.

4.5.4 Research Question and Hypothesis, Set 4

RQ_4: Are there significant differences between different NN model's epoch counts used for Insider Threat detection?

$H_{4\text{-}n}$: Of NN models created to detect Insider Threats, the use of epochs makes no significant difference in learner performance.

$H_{4\text{-}a}$: Of NN models created to detect Insider Threats, the use of epochs makes a significant difference in at least one learner's epoch performance.

If it were true that the count of Epochs did not make a difference, then the Epochs of any Scaling-Learner triplets would be in the same Dunn's group. That was not the case. By way of example, group A included Robust MLP 10 & 35, Normalized MLP 10 & 35, and Standardized MLP 10 & 35, while in group P was Non-Rescaled MLP 10, with

Non-Rescaled MLP 35 in group C. If Epochs made no difference, both Epochs 10 and 35 would exist in the same Dunn's group, regardless of the scaling technique used. Similar discrepancies can be found among other learners and the evidence was illustrated in Figures 11-13 of the Dunn's Test Letter Grouping. For this reason, the null hypothesis ($H_{4\text{-}n}$) was rejected, and the alternative hypothesis ($H_{4\text{-}a}$) was accepted

Chapter 5: Discussion and Conclusions

This praxis examined the potential for categorizing and detecting Insider Threats using TensorFlow NN to optimize procedures for security professionals. Over 2,000 tests were run using 10 risk identifiers against a preprocessed publicly available CERT dataset to isolate the most effective NN learners.

5.1 Discussion

This chapter summarizes the discussion points and conclusions of the findings of this praxis, shares contributions to the body of knowledge, and highlights some recommended paths for future research.

5.2 Conclusions

This study focused on Insider Threat detection with TensorFlow NN algorithms. The results demonstrated the capability of various NN algorithms such as ELM-klm and MLP to accurately detect Insider Threat activity based on the analysis of user and network data. The pairwise comparison generated by Dunn's grouping test indicated that there is a significant difference among different rescaling techniques. Robust scaling, however, accounted for the top five performers, all within the top performing group, group A. This suggests that a significant difference exists in at least one model.

In addition, the Epoch count applied to the various models resulted in the discovery of significant differences across models. It was discovered that the top performing model MLP showed no significant difference when applied to Normalization, Standardization and Robust scaling. However, when the same model was executed with

Non-Rescaled, the results indicate that there was a significant difference. This relationship was evident from the results shown in Figure 11-Figure 13.

According to KW Dunn's post hoc test, the highest median accuracy was Robust MLP 35 at 0.978992 and average of .977591. However, the highest average accuracy was Non-Rescaled ELM-klm 10 at of 0.980392. The average accuracy produced by both models, as well as others, fell within the range of the top average results obtained by Dr. Noever in Table 13 (Noever, 2019).

Looking at speed, Non-Rescaled ELM-klm 10 ranked in the top seven fastest learners, ranked in sixth place at 154.46 seconds. Standardized ELM-klm 10 ranked first at 145.83 seconds, Normalized ELM-klm 10 ranked seventh at 153.47 seconds, and Robust ELM-klm 10 ranked third at 152.19 seconds. This is an indication that the ELM-klm consistently ranks as a top performer in terms of speed and may be an attractive alternative for security professionals seeking an algorithm where timely results are needed for Insider Threat detection applications. Finally, this praxis leveraged the user information from 10 risk indicators to achieve the performance outcomes.

5.3 Contributions to Body of Knowledge

This praxis leveraged TensorFlow NN models to demonstrate and identify best performing algorithms for Insider Threat detection, which had yet to be evaluated on Dr. Noever's dataset (Noever, 2019). By identifying some of the top performing models, this praxis shares insights useful to information security professionals designing ML models appropriate for Insider Threat detection. This praxis also proposed a supervised learning solution that adapts to evolving concepts using TensorFlow Neural Network algorithms.

Comparing this approach to traditional supervised learning approaches and demonstrates the effectiveness of using data which was created using real-world network scenarios.

The praxis also leverages the Random Forest Classifier as a feature selection technique, proving it was possible to use fewer features to achieve quicker execution times while achieving comparable accuracy. Table 18 (see Appendix A) shares a comparative summary for the accuracy and speed of the top performers identified in this praxis, compared to the top seven performers in Dr. Noever's research, for each of sampling techniques he utilized. This is beneficial in closing the gap regarding detection and containment times for Insider events across many industries. Finally, this praxis demonstrated that it is possible to use fewer risk indicators to achieve comparable performance to Dr. Noever's.

5.4 Recommendations for Future Research

The dataset used in this praxis was based on synthetic data based on real-world scenarios and, as such, may not be a true and accurate representation of performance and detection in a changing environment where users may alter their behaviors. One suggestion for future research could be the creation of a packaged tool utilizing the top performing models and testing in a live environment to compare the results to some of the available tools on the market targeting Insider Threat detection. Another idea for future research is the use of AutoML libraries like PyCaret to explore these NN models using the same dataset, or a similar dataset, to evaluate the capabilities of the model to continue to work more effectively when new and evolving threats occur in a changing environment (Abadi et al., 2015). A final suggestion is to use a different approach or

platform to run the models; possibly the use of quantum computing to achieve greater runtimes than Google Collab Pro TPUs.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... and Zheng, X. (2015, October). *TensorFlow: Large-Scale machine learning on heterogeneous systems* [White paper]. Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI '15), Savannah, Georgia, USA. https://research.google/pubs/tensorflow-large-scale-machine-learning-on-heterogeneous-distributed-systems/?form=MG0AV3

Ali, P. J. M., Faraj, R. H., Koya, E. (2014). Data normalization and standardization: a technical report. *Machine Learning Technical Reports, 1*(1), 1-6. https://www.researchgate.net/profile/Peshawa-Muhammad-Ali/publication/340579135_Data_Normalization_and_Standardization_A_Technical_Report/links/5e91b65d299bf130798fc1bd/Data-Normalization-and-Standardization-A-Technical-Report.pdf

Al-Mhiqani, M. N., Ahmed, R., Zainal Abidin, Z. A., and Isnin, S. N. (2021). An integrated imbalanced learning and deep neural network model for insider threat detection. *International Journal of Advanced Computer Science and Applications, 12*(1), 573–577. https://doi:10.14569/IJACSA.2021.0120166

Al-Shehari, T. A., Rosaci, D., Al-Razgan, M., Alfakih, T., Kadrie, M., Afzal, H., and Nawaz, R. (2024). Enhancing insider threat detection in imbalanced cybersecurity settings using the density-based local outlier factor algorithm. *IEEE Access, 12*, 34820–34834. https://doi.org/10.1109/ACCESS.2024.3373694

Al-Shehari, T., Al-Razgan, M., Alfakih, T., Alsowail, R. A., and Pandiaraj, S. (2023). Insider threat detection model using anomaly-based isolation forest algorithm. *IEEE Access, 11*, 118170–118185. https://doi:10.1109/ACCESS.2023.3326750

Al-Shehari, T. and Alsowail, R. A. (2021). An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques. *Entropy, 23*(10), 1258. https://doi:10.3390/e23101258

Al-Shehari, T. and Alsowail, R. A. (2022). Random resampling algorithms for addressing the imbalanced dataset classes in insider threat detection. *International Journal of Information Security, 22*, 611–629. https://doi.org/10.1007/s10207-022-00651-1

AlSlaiman, M., Salman, M. I., Saleh, M. M., and Wang, B. (2023). Enhancing false negative and positive rates for efficient insider threat detection. *Computers & Security, 126*, 103066. https://doi:10.1016/j.cose.2022.103066

Alzaabi, F. R., & Mehmood, A. (2024). A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods. *IEEE Access, 12*, 30907-30927.

Amirian, M. and Schwenker, F. (2020). Radial basis function networks for convolutional neural networks to learn similarity distance metric and improve interpretability. *IEEE Access, 8*, 9133368. https://doi.org/10.1109/ACCESS.2020.3007337

Assaad, H. I., Zhou, L., Carroll, R. J. and Wu, G. (2014). Rapid publication-ready MS-Word tables for one-way ANOVA. *SpringerPlus, 3,* 474. https://doi.org/10.1186/2193-1801-3-474

Balakrishnan, S. S. (2021). Understanding random forest. *Analytics Vidhya.* https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

Bartoszewski, F. W. (2022). *Machine learning and anomaly detection for insider threat detection* [Master's thesis, Heriot-Watt University]. Heriot-Watt University. https://www.ros.hw.ac.uk/bitstream/handle/10399/4593/BartoszewskiFW_0422_macsSS.pdf

Besnaci, S., Hafidi, M. and Lamia, M. (2023, March 6-7). *Dealing with extremly unbalanced data and detecting insider threats with deep neural networks* [Conference paper]. 2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS), Blida, Algeria. doi:10.1109/ICAECCS56710.2023.10105103

Bevans, R. (2020). One-way ANOVA: When and how to use it (with examples). *Scribbr*. https://www.scribbr.com/statistics/one-way-anova/ (Accessed: 22 February 2025)

Bin Sarhan, B. and Altwaijry, N. (2023). Insider threat detection using machine learning approach. *Applied Sciences, 13*(1), 259. https://doi.org/10.3390/app13010259

Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multivariable functional interpolation, and adaptive networks. *Complex Systems, 2*(3), 321-355.\ https://archive.org/details/DTIC_ADA196234?form=MG0AV3

CALO Project. (2015). Enron email dataset. https://www.cs.cmu.edu/~enron/ (Accessed: 14 March 2024)

Cappelli, D., Moore, A., and Trzeciak, R. (2012). *The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud).* Addison-Wesley Professional

Chollet, F. (2023). TensorFlow core: Making new layers and models via subclassing. *TensorFlow*. https://www.tensorflow.org/guide/keras/custom_layers_and_models (Accessed: 22 November 2024)

CrowdStrike. (2024). CrowdStrike : 2024 Threat hunting report. *CrowdStrike.* https://go.crowdstrike.com/rs/281-OBQ-266/images/CrowdStrike2024ThreatHuntingReport.pdf?version=0 (Accessed: 22 February 2025)

Cyberseer. (n.d.). Compromise insider. *Cyberseer.* https://www.cyberseer.net/company/compromise-insider/ (Accessed: 22 October 2024)

Defense CounterIntelligence and Security Agency. (2023). Insider threat indicators in user activity monitoring [Job Aid]. *Defense CounterIntelligence and Security Agenc*y. https://www.cdse.edu/Portals/124/Documents/jobaids/insider/Insider-Threat-Indicators-in-UAM.pdf?form=MG0AV3

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., ... and Saurous, R. A. (2017). Tensorflow distributions. *arXiv preprint arXiv:1711.10604 [cs.LG]*. https://doi.org/10.48550/arXiv.1711.10604

Dinno, A. (2015). Nonparametric pairwise multiple comparisons in independent groups using Dunn's test. *The Stata Journal, 15*(1), 292-300. https://doi.org/10.1177/1536867X1501500117

Dosh, M. H. (2021). Detecting insider threat within institutions using CERT dataset and different ML techniques. *Periodicals of Engineering and Natural Sciences (PEN), 9*(2), 873–884. http://dx.doi.org/10.21533/pen.v9i2.1911

Elreedy, D., Atiya, A.F. and Kamalov, F. (2024). A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning. *Machine Learning, 113,* 4903–4923. https://doi.org/10.1007/s10994-022-06296-4

Farouk, M., Sakr, R. H. and Hikal, N. (2024). Identifying the most accurate machine learning classification technique to detect network threats. *Neural Computing and Applications, (36),* 8977-8944. doi:10.1007/s00521-024-09562-9

Fernández-Delgado, M., Cernadas, E., Barro, S., Ribeiro, J. and Neves, J. (2014). Direct Kernel Perceptron (DKP): Ultra-fast kernel ELM-based classification with non-iterative closed-form weight calculation. *Neural Networks, 50,* 60-71. https://doi.org/10.1016/j.neunet.2013.11.002

Frost, J. (n.d.). Using post hoc tests for ANOVA. *Statistics by Jim.* https://statisticsbyjim.com/anova/post-hoc-tests-anova/ (Accessed: 19 January 2025)

Frost, J. (2020). Hypothesis testing: An intuitive guide for making data-driven decisions. *Statistics by Jim Publishing*.

Future Machine Learning. (n.d.). Understanding K-nearest neighbors: A detailed overview. *Future Machine Learning.* https://futuremachinelearning.org/understanding-k-nearest-neighbors-a-detailed-overview/?form=MG0AV3 (Accessed: Accessed: 2 February 2025)

GeeksforGeeks. (2024). What are outliers in data? *GeeksforGeeks.* https://www.geeksforgeeks.org/what-are-outliers-in-data/?ref=header_outind (28 December 2024)

GeeksforGeeks. (2025a). Multi-layer perceptron learning in Tensorflow. *GeeksforGeeks.* https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/ (Accessed: 5 December 2024)

GeekforGeeks. (2025b). ML: Data preprocessing in Python.*GeeksforGeeks*. https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/ (Accessed date: 15 December 2024)

GeeksforGeeks. (n.d.). What is a neural network? *GeeksforGeeks*. https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/

Géron, A. (2023). *Hands-on machine learning with scikit-learn, keras, and TensorFlow* (3rd ed.). O'Reilly Media.

Gimelfarb, M. (2025). Cascade-correlation neural networks. *GitHub, Inc.* https://github.com/mike-gimelfarb/cascade-correlation-neural-networks?form=MG0AV3 (Accessed: 12 February 2025)

Glinskiy, V., Ismayilova, Y., Khrushchev, S., Logachov, A., Logachova, O., Serga, L., Yambbartsev, A. and Zaykov, K. (2024). Modifications to the Jarque–Bera test. *Mathematics, 12*(16), 2523. https://doi.org/10.3390/math12162523

Google Developers. (n.d.). Neural networks. *Machine Learning*. https://developers.google.com/machine-learning/crash-course/neural-networks?form=MG0AV3

Gurucul. (2024). 2024 Insider threat report. *Cybersecurity Insiders*. https://go1.gurucul.com/2024-insider-threat-report?_gl=118w81m9_gcl_auMjQzNzI4NzA0LjE3NDAzNDg1MjA._gaOTA2MzUxMDM4LjE3NDAzNDg1MjE._ga_XK6L3BZR7J*MTc0MDQyODg2OC4yLjEuMTc0MDQyOTIzNS42MC4wLjA

He, H. and 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

Homoliak, I., Toffalini, F., Guarnizo, J. D., Elovici, Y. and Ochoa, M. (2017, October). *TWOS: A dataset of malicious insider threat behavior based on a gamified competition* [Conference paper]. 2017 International Workshop. https://www.researchgate.net/profile/Ivan-Homoliak/publication/320745933_TWOS_A_Dataset_of_Malicious_Insider_Threat_Behavior_Based_on_a_Gamified_Competition/links/59f9491b458515547c26b848/TWOS-A-Dataset-of-Malicious-Insider-Threat-Behavior-Based-on-a-Gamified-Competition.pdf?form=MG0AV3

Howlett, R. J. and Jain, L. C. (eds.) (2001). *Radial basis function networks 1: Recent developments in theory and applications*. Physica-Verlag. https://searchworks.stanford.edu/view/13998532?form=MG0AV3

Hryniowski, A. and Wong, A. (2019). DeepLABNet: End-to-end learning of deep radial basis networks with fully learnable basis functions. *arXiv preprint arXiv:1911.09257 [cs.NE]*. https://arxiv.org/pdf/1911.09257

Huang, G. B., Zhu, Q. Y. and Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing, 70*(1–3), 489-501. https://web.njit.edu/~usman/courses/cs675_fall20/ELM-NC-2006.pdf?form=MG0AV3

Huang, G.-B. (2015). What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle. *Cognitive Computation, 7*(3), 263-278. https://www.extreme-learning-machines.org/pdf/ELM-Rosenblatt-Neumann.pdf

IBM. (2021). Supervised vs. unsupervised learning: What's the difference? *IBM.*https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning

Imperva. (n.d.). Insider threat. *Imperva.* https://www.imperva.com/learn/application-security/insider-threats/

Janjua, F., Masood, A., Abbas, H. and Rashid, I. (2020). Handling insider threat through supervised machine learning techniques. *Procedia Computer Science, 177*, 64-71. https://www.researchgate.net/publication/346870471_Handling_Insider_Threat_Through_Supervised_Machine_Learning_Techniques/fulltext/5fd1c190a6fdcc697bf2b67b/Handling-Insider-Threat-Through-Supervised-Machine-Learning-Techniques.pdf.

Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A, 374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE, 78*(9), 1464-1480. https://doi.org/10.1109/5.58325

Krogh, A. and Vedelsby, J. (1994). Neural network ensembles, cross-validation, and active learning. *Advances in Neural Information Processing Systems, 7,* 231-238.

Kulesza, A. and Taskar, B. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning, 5*(2–3), 123–286. https://doi.org/10.1561/2200000044

Labory, J., Njomgue-Fotso, E. and Bottini, S. (2024). Benchmarking feature selection and feature extraction methods to improve the performances of machine-learning algorithms for patient classification using metabolomics biomedical data.*Computational and Structural Biotechnology Journal, 23*, 1274–1287. https://doi:10.1016/j.csbj.2024.03.016

Laerd Statistics. (n.d.). One-way ANOVA using Minitab. Laerd Statistics. *Laerd statistics*. https://statistics.laerd.com/minitab-tutorials/one-way-anova-using-minitab.php

Le, D. C. and Zincir-Heywood, N. (2021). Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management, 18*(2), 1152–1164. https://doi.org/10.1109/TNSM.2021.3071928

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444.

Lehtokangas, M. (2000). Modified cascade-correlation learning for classification. *IEEE Transactions on Neural Networks, 11*(3), 795–798. https://doi.org/10.1109/72.846749

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J. and Das, K. (2000, January 25-27). *Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation.* Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX), Los Alamitos, CA. https://www.scirp.org/reference/referencespapers?referenceid=1516859.

Luckey, D., Stebbins, D., Orrie, R., Rebhan, E., Bhatt, S. D. and Beaghley, S. (2019). *Continuous evaluation approaches for insider threats.* Defense Technical Information Center. Rand Corporation, Santa Monica, CA, USA. https://apps.dtic.mil/sti/pdfs/AD1078959.pdf?form=MG0AV3

Machine Learning. (n.d.). ML Concepts: Classification:ROC and AUC. *Machine Learning.* https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (Accessed: May 18, 2024)

Mahajan, A. A**.** (2023). Hybrid model using LOF and iForest algorithms for detection of insider threats. TechRxiv. https://www.techrxiv.org/doi/full/10.36227/techrxiv.24522679.v1

Mamidanna, S. K., Reddy, C. R. K. and Gujju, A. (2022, January 28-29). *Detecting an insider threat and analysis of XGBoost using hyperparameter tuning* [Conference paper]. 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India. doi:10.1109/ACCAI53970.2022.9752509

Manoharan, P., Yin, J., Wang, H., Zhang, Y. and Ye, W. (2023). Insider threat detection using supervised machine learning algorithms. *Telecommunication Systems, 87*(1), 899-915. https://doi.org/10.1007/s11235-023-01085-3

McHugh, J. (2000). Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security, 3*(4), 262-294. https://doi.org/10.1145/382912.382923

Melssen, W., Wehrens, R. and Buydens, L. (2006). Supervised Kohonen networks for classification problems. *Chemometrics and Intelligent Laboratory Systems, 83*(2), 99-113. http://dx.doi.org/10.1016/j.chemolab.2006.02.003

Mesiar, R. and Sheikhi, A. (2021). Nonlinear random forest classification, a copula-based approach. *Applied Sciences, 11*(15), 7140. https://doi.org/10.3390/app11157140

Minitab. (2024). Minitab support. *Minitab, LLC.* https://support.minitab.com/en-us/minitab/?SID=0

MIT Lincoln Laboratory. (1998). 1998 DARPA98 intrusion detection evaluation dataset. *MIT Lincoln Laboratory.* https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset?form=MG0AV314. (Accessed: 15 December 2024)

MIT Lincoln Laboratory. (1999). DARPA intrusion detection evaluation dataset. *MIT Lincoln Laboratory*. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

Nepal, S. and Joshi, B. (2021). User behavior analytics for insider threat detection using deep learning. *Peer Reviewed ISSN*, 2350–8914 .

Neuroinformatics Group. (n.d.). SNNS: Simulation environment. https://www.inf.ufsc.br/~aldo.vw/patrec/SNNS/UserManual/node193.html

Noever, D. (2019). Classifier suites for insider threat detection. *arXiv preprint arXiv:1901 [cs.LG]*.10948. https://arxiv.org/abs/1901.10948.

Nurse, J. R. C., Buckley, O., Legg, P. A., Goldsmith, M., Creese, S., Wright, G. R. T. and Whitty, M. (2014, May 17-18*). Understanding insider threat: A framework for characterising attacks* [Conference paper]. 2014 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA.. https://doi.org/10.1109/SPW.2014.38

Pantelidis, E., Bendiab, G., Shiaeles, S. and Kolokotronis, N. (2021, July 26-28). *Insider threat detection using deep autoencoder and variational autoencoder neural networks* [Conference paper]. 2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece. http://doi:10.1109/CSR51186.2021.9527925

Päpper, M. (2023). Understanding the difference between weight decay and L2 regularization. *Paepper.com.* https://www.paepper.com/blog/posts/understanding-the-difference-between-weight-decay-and-l2-regularization/ (Accessed 22 November 2024)

Ponemon Institute. (2023). Cost of insider risks global report 2023. *Ponemon Sullivan Privacy Report.* https://ponemonsullivanreport.com/2023/10/cost-of-insider-risks-global-report-2023/ (Accessed 22 September 2024)

Rahmani, A. M., Bayramov, S. and Kiani Kalejahi, B. (2022). Internet of things applications: Opportunities and threats. *Wireless Personal Communications, 122*(1), 451-476. https://doi.org/10.1007/s11277-021-08907-0

Rauf, U., Wei, Z. and Mohsen, F.. (2023, October 16-18). *Employee watcher: A machine learning-based hybrid insider threat detection framework* [Conference paper]. 2023 7th Cyber Security in Networking Conference (CSNet), Montreal, QC, Canada. https://doi.org/10.1109/CSNet59123.2023.10339777

Ravipati, R. and Abualkibash, M. (2019). Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets - A review paper. *International Journal of Computer Science & Information Technology (IJCSIT), 11*(3), 16-29. https://doi.org/10.2139/ssrn.3428211

Ren, X., & Wang, L. (2019, March 22). *DMNAED: A novel framework based on dynamic memory network for abnormal event detection in enterprise networks* [Conference paper]. In J. Li, X. Cao, & X. Guo (Eds.), Advances in knowledge discovery and data mining (pp. 574–586). Springer, Cham. https://doi.org/10.1007/978-3-030-16148-4_44

Santos, E., Nguyen, H., Yu, F., Kim, K. J., Li, D., Wilkinson, J. T., Olson, A., Russell, J. and Clark, B. (2012). Intelligence analyses and the insider threat. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 42*(2), 331-347. https://doi: 10.1109/TSMCA.2011.2162500

scikit-learn developers. (2025a). Outlier detection with Local Outlier Factor (LOF). *Scikit-learn*. https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html (Accessed: 22 February 2025)

scikit-learn developers. (2025b). DecisionTreeClassifier. *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (Accessed: 22 February 2025)

scikit-learn. (2025a). Model evaluation: Confusion matrix. Scikit-learn. https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics (Accessed: 22 February 2025)

scikit-learn. (2025b). OneHotEncoder. *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html?form=MG0AV3 (Accessed: 22 February 2025)

scikit-learn. (2025c). 6.3 Preprocessing data. *Scikit-learn*. https://scikit-learn.org/stable/modules/preprocessing.html?form=MG0AV3 https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifierf (Accessed: 22 February 2025)

scikit-learn. (2025d). RobustScaler. *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html (Accessed: 22 February 2025)

scikit-learn. (2025e). Importance of feature scaling.. *Scikit-learn*. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html?form=MG0AV3 (Accessed: 22 February 2025)

Securonix. (2024). 2024 Insider threat report: Trends, challenges, and solutions. *Cybersecurity Insiders*. https://www.securonix.com/wp-content/uploads/2024/01/2024-Insider-Threat-Report-Securonix-final.pdf?form=MG0AV3

Selvam, A. and Selvy, T. (2022). Improved classification model in handling insider threat through supervised machine learning techniques. *NeuroQuantology, 20(*10), 7787-7793. https://www.neuroquantology.com/media/article_pdfs/7787-7793.pdf

Sharma, B., Pokharel, P. and Joshi, B. (2020, July 1-3). *User behavior analytics for anomaly detection using LSTM autoencoder-insider threat detection* [Conference paper]. Proceedings of the 11th International Conference on Advances in Information Technology. Bangkok, Thailand

Shultz, T. R. (2000). A tutorial on cascade-correlation. *Thomas R. Shultz.* https://www.psych.mcgill.ca/perpg/fac/shultz/personal/Recent_Publications_files/cc_tutorial_files/v3_document.htm (Accessed: 22 January 2025)

SixSigma.us. (2024). Levene's test: Your guide to understanding, performing, and interpreting homogeneity of variance. *SixSigma.us*. https://www.6sigma.us/six-sigma-in-focus/levenes-test/ (Accessed: 20 January 2025)

Stack Overflow. (2025). How to implement RBF activation function in keras? *Stack Exchange Inc.* https://stackoverflow.com/questions/53855941/how-to-implement-rbf-activation-function-in-keras (Accessed: 12 November 2024)

Statology. (2023). Kruskal-Wallis test: Definition, formula, and examples. *Statology*. https://www.statology.org/kruskal-wallis-test/ (Accessed: 20 December 2024)

Stewart, J. (2017). What is machine learning? The Conversation. *The Conversation US, Inc*. https://theconversation.com/what-is-machine-learning-76759 (Accessed: 22 August 2024)

Struniawski, K. (2024). TfELM: TensorFlow extreme learning machines. *GitHub, Inc*. https://github.com/KStruniawski/TfELM (Accessed: : 22 December 2024)

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July 8-10). *A detailed analysis of the KDD CUP 99 data set* [Symposium]. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009). https://www.ecb.torontomu.ca/~bagheri/papers/cisda.pdf.

TensorFlow. (2022). Citing TensorFlow: Large-scale machine learning on heterogeneous systems. *TensorFlow*. https://www.tensorflow.org/about/bib (Accessed: 25 February 2025)

TensorFlow. (2023). tfp.distributions.DeterminantalPointProcess. *TensorFlow*. https://www.tensorflow.org/probability/api_docs/python/tfp/distributions/DeterminantalPointProcess (Accessed: 25 February 2025)

TensorFlow. (2024a). Classification on TPUs. *TensorFlow*. https://www.tensorflow.org/guide/tpu#classification_on_tpus (Accessed: 25 February 2025)

TensorFlow. (2024b). Keras functional API guide. *TensorFlow*. https://www.tensorflow.org/guide/keras/functional_api (Accessed: January 8, 2025) (Accessed: 25 February 2025)

TensorFlow. (2024c). Probabilistic PCA. *TensorFlow*. https://www.tensorflow.org/probability/examples/Probabilistic_PCA (Accessed: 25 February 2025)

TensorFlow. (2024d). tf.keras.layers.average. *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Average (Accessed: 25 February 2025)

TensorFlow. (2024e). tf.keras.layers.bidirectional. *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Bidirectional (Accessed: 25 February 2025)

TensorFlow. (n.d.-a). Composing decision forest and neural network models. https://www.tensorflow.org/decision_forests/tutorials/model_composition_colab?hl=en (Accessed: 20 February 2025)

TensorFlow. (n.d.-b). Keras sequential model. *TensorFlow*. .https://www.tensorflow.org/guide/keras/sequential_model (Accessed: 20 February 2025)

TensorFlow. (n.d.-c). MLP core guide. https://www.tensorflow.org/guide/core/mlp_core (Accessed: 20 February 2025)

TensorFlow. (n.d.-d). Multilayer perceptrons for digit recognition with core APIs. https://www.tensorflow.org/guide/core/mlp_core (Accessed: 20 February 2025)

TensorFlow. (n.d.-e). Overview: TensorFlow probability. https://www.tensorflow.org/probability/overview (Accessed: 20 February 2025)

TensorFlow. (n.d.-f). Training comprehensive guide: Quantization aware model. *TensorFlow.* https://www.tensorflow.org/model_optimization/guide/quantization/training_comprehensive_guide#define_quantization_aware_model (Accessed: 20 February 2025)

The IoT Academy. (2025). Dimensionality reduction. *The IoT Academy.* https://www.theiotacademy.co/blog/dimensionality-reduction/?form=MG0AV3 (Accessed: 22 November 2025)

Vargas, D. A., Bueno López, R., Casas, D. E., Osorio-Doblado, A. M., Rodríguez, K. M., Vargas, N., Gragg, S. E., Brashears, M. M., Miller, M. F. and Sanchez-Plata, M. X. (2024). Modernization data analysis and visualization for food safety research outcomes. *Applied Sciences, 14*(12), 5259.

Wei, Z., Rauf, U. and Mohsen, F. (2024). E-watcher: Insider threat monitoring and detection for enhanced security. *Annals of Telecommunications, (79),* 819-831. doi:10.1007/s12243-024-01023-7

Wheelan, C. (2013). Naked statistics: Stripping the dread from the data. W. W. Norton & Company.

Yuan, F., Cao, Y., Shang, Y., Liu, Y., Tan, J. and Fang, B. (2018, June 11-13). *Insider threat detection with deep neural network* [Conference paper]. 18th International Conference on Computational Science, ICCS 2018, J10860 LNCS,Wuxi, China. http://dx.doi.org/10.1007/978-3-31

Appendix A

Figure 14: Correlation Matrix and Variance for the Dataset Feature Section

Feature Labels	Variance	Label	risk_leak	hr_leak	risk_thief	hr_thief	risk_sabotage	dow_sabotage	device_freq	file_freq	email_sentiment	unauthorized_log
target	0.03	1.00	0.57	0.57	0.05	0.07	0.04	0.04	0.05	0.01	0.01	0.22
risk_leak	0.07	0.57	1.00	1.00	0.04	0.05	0.00	-0.01	-0.01	-0.01	0.00	0.09
hr_leak	0.07	0.57	1.00	1.00	0.04	0.05	0.00	-0.01	-0.01	-0.01	0.00	0.09
risk_thief	5263.31	0.05	0.04	0.04	1.00	0.72	0.18	0.19	0.16	0.20	0.49	-0.02
hr_thief	69.66	0.07	0.05	0.05	0.72	1.00	0.23	0.24	0.27	0.32	0.27	-0.01
risk_sabotage	2826.10	0.04	0.00	0.00	0.18	0.23	1.00	0.96	0.31	0.31	0.19	0.03
dow_sabotage	510.25	0.04	-0.01	-0.01	0.19	0.24	0.96	1.00	0.37	0.37	0.18	0.04
device_freq	7650.05	0.05	-0.01	-0.01	0.16	0.27	0.31	0.37	1.00	0.92	-0.06	0.06
file_freq	11604.96	0.01	-0.01	-0.01	0.20	0.32	0.31	0.37	0.92	1.00	-0.04	0.02
email_sentiment	13084.03	0.01	0.00	0.00	0.49	0.27	0.19	0.18	-0.06	-0.04	1.00	-0.04
unauthorized_log	302.07	0.22	0.09	0.09	-0.02	-0.01	0.03	0.04	0.06	0.02	-0.04	1.00

Table 14: JB P-Value Test Results- Learner Values More Than or Equal to 0.05

Model	Scaler	Epoch	P-Value
ANN	Non-Rescaled	10	0.521464317
ANN	Non-Rescaled	35	0.56812896
BDK	Non-Rescaled	10	0.523752395
BDK	Non-Rescaled	35	0.520725648
Cascor	Non-Rescaled	10	0.523752395
Cascor	Non-Rescaled	35	0.520725648
DKP	Non-Rescaled	10	0.851239521
DKP	Non-Rescaled	35	0.742776097
DPP	Non-Rescaled	10	0.523752395
DPP	Non-Rescaled	35	0.520725648
ELM-klm	Non-Rescaled	10	0.290317962
ELM-klm	Non-Rescaled	35	0.378173039
ELM-m	Non-Rescaled	10	0.742997293
ELM-m	Non-Rescaled	35	0.741014297
LVQ	Non-Rescaled	10	0.523752395
LVQ	Non-Rescaled	35	0.520725648
MLP	Non-Rescaled	10	0.836464214
MLP	Non-Rescaled	35	0.43969309
MLP-wd	Non-Rescaled	10	0.609862102
MLP-wd	Non-Rescaled	35	0.885058953
NN	Non-Rescaled	10	0.859665786
NN	Non-Rescaled	35	0.706723174
PCNN	Non-Rescaled	10	0.764032169
PCNN	Non-Rescaled	35	0.694409618
RBF	Non-Rescaled	10	0.523752395
RBF	Non-Rescaled	35	0.520725648
RBF-dda	Non-Rescaled	10	0.523752395
RBF-dda	Non-Rescaled	35	0.520725648
RBFN	Non-Rescaled	10	0.523752395
RBFN	Non-Rescaled	35	0.520725648
BDK	Normalized	10	0.394959826
Cascor	Normalized	10	0.394959826
DKP	Normalized	35	0.993393633

Model	Scaler	Epoch	P-Value
DPP	Normalized	10	0.394959826
ELM-klm	Normalized	35	0.121392551
ELM-m	Normalized	35	0.728064978
LVQ	Normalized	10	0.394959826
RBF	Normalized	10	0.394959826
RBF-dda	Normalized	10	0.394959826
RBFN	Normalized	10	0.394959826
ANN	Robust	10	0.329349311
ANN	Robust	35	0.628236988
BDK	Robust	10	0.80596217
Cascor	Robust	10	0.80596217
DKP	Robust	10	0.483327851
DKP	Robust	35	0.386788705
DPP	Robust	10	0.80596217
ELM-klm	Robust	10	0.412365522
ELM-klm	Robust	35	0.50941749
ELM-m	Robust	10	0.135365402
ELM-m	Robust	35	0.762362779
LVQ	Robust	10	0.80596217
MLP	Robust	35	0.72650089
MLP-wd	Robust	10	0.503274067
MLP-wd	Robust	35	0.074269586
NN	Robust	10	0.808549544
NN	Robust	35	0.30036662
RBF	Robust	10	0.80596217
RBF-dda	Robust	10	0.80596217
RBFN	Robust	10	0.80596217
ANN	Standardized	35	0.924320595
DKP	Standardized	10	0.666565706
DKP	Standardized	35	0.103459206
ELM-klm	Standardized	10	0.670379282
ELM-klm	Standardized	35	0.154246032
ELM-m	Standardized	35	0.423641045
MLP	Standardized	10	0.441589455
MLP-wd	Standardized	35	0.981924359
NN	Standardized	10	0.934717113
NN	Standardized	35	0.946419788
PCNN	Standardized	10	0.917955366
PCNN	Standardized	35	0.450224641

Sum of P-Value	Scaler			
Model	Non-Rescaled	Normalized	Robust	Standardized
Cascor	1.044478044	0.412616807	0.80596217	0.053814205
ANN	1.089593277	0.030096159	0.957586299	0.931248051
BDK	1.044478044	0.412616807	0.80596217	0.033113938
DKP	1.594015619	1.013809941	0.870116556	0.770024912
DPP	1.044478044	0.412616807	0.80596217	0.053814205
ELM-klm	0.668491001	0.149748094	0.921783012	0.824625314
ELM-m	1.48401159	0.728064978	0.897728181	0.423641045
LVQ	1.044478044	0.412616807	0.80596217	0.053814205
MLP	1.276157303	0.053642663	0.730764856	0.469944999
MLP-wd	1.494921055	0.030096159	0.577543653	0.981924359
NN	1.566388959	0.056711087	1.108916165	1.8811369
PCNN	1.458441787	0.038594584	0.039877644	1.368180007
RBF	1.044478044	0.412616807	0.80596217	0.053814205
RBF-dda	1.044478044	0.412616807	0.80596217	0.053814205
RBFN	1.044478044	0.412616807	0.80596217	0.053814205

Figure15: JB P-Value Test Results Summary by Learner and Scaler

Table 15: Top 7 Under-sampled Result by Model Accuracy
(Noever, 2019)

Model	Rank	Accuracy	Execution Time (s)
Boosted Logistic Regression	1	0.96	0.72
Random Forest by Randomization	2	0.94	1.62
Logistic Model Trees	3	0.92	1.28
Random Forest	4	0.9	0.69
Regularized Random Forest	5	0.9	0.83
Rule-Based Classifier (PART)	6	0.9	0.86
J48 (C4.5)-like Trees	7	0.9	1.78

Table 16: TensorFlow Neural Network Performance Metrics by Rescale Type

Scaler	Category	Epochs	Model	Time (min)	Accuracy	Precision	Recall	F1 Score	AUC
Normalized	Normalized ANN 10	10	ANN	4.19	0.97	0.00	0.00	0.00	0.86
Normalized	Normalized ANN 35	35	ANN	10.76	0.97	0.00	0.00	0.00	0.23
Normalized	Normalized BDK 10	10	BDK	2.48	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized BDK 35	35	BDK	6.14	0.97	0.65	0.54	0.59	0.86
Normalized	Normalized Cascor 10	10	Cascor	2.52	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized Cascor 35	35	Cascor	6.18	0.97	0.65	0.54	0.59	0.86
Normalized	Normalized DKP 10	10	DKP	6.47	0.97	0.80	0.17	0.28	0.87
Normalized	Normalized DKP 35	35	DKP	12.23	0.98	0.71	0.50	0.59	0.95
Normalized	Normalized DPP 10	10	DPP	2.53	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized DPP 35	35	DPP	6.15	0.97	0.65	0.54	0.59	0.86
Normalized	Normalized ELM-klm 10	10	ELM-klm	2.56	0.97	0.65	0.46	0.54	0.78

Scaler	Category	Epochs	Model	Time (min)	Accuracy	Precision	Recall	F1 Score	AUC
Normalized	Normalized ELM-klm 35	35	ELM-klm	6.27	0.97	0.65	0.54	0.59	0.88
Normalized	Normalized ELM-m 35	35	ELM-m	7.81	0.97	0.65	0.54	0.59	0.97
Normalized	Normalized LVQ 10	10	LVQ	2.48	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized LVQ 35	35	LVQ	6.07	0.97	0.65	0.54	0.59	0.86
Normalized	Normalized MLP 10	10	MLP	3.06	0.98	0.68	0.54	0.60	0.95
Normalized	Normalized MLP 35	35	MLP	8.21	0.98	0.68	0.54	0.60	0.97
Normalized	Normalized MLP-wd 10	10	MLP-wd	3.00	0.97	0.00	0.00	0.00	0.79
Normalized	Normalized MLP-wd 35	35	MLP-wd	7.48	0.97	0.00	0.00	0.00	0.87
Normalized	Normalized NN 10	10	NN	2.88	0.98	0.68	0.54	0.60	0.96
Normalized	Normalized NN 35	35	NN	6.91	0.98	0.68	0.54	0.60	0.97
Normalized	Normalized PCNN 10	10	PCNN	3.12	0.98	0.68	0.54	0.60	0.96
Normalized	Normalized RBF 10	10	RBF	2.63	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized RBF 35	35	RBF	6.14	0.97	0.65	0.54	0.59	0.86
Normalized	Normalized RBF-dda 10	10	RBF-dda	2.48	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized RBF-dda 35	35	RBF-dda	6.21	0.97	0.65	0.54	0.59	0.86
Normalized	Normalized RBFN 10	10	RBFN	2.43	0.97	0.75	0.38	0.50	0.68
Normalized	Normalized RBFN 35	35	RBFN	6.27	0.97	0.65	0.54	0.59	0.86
Standardized	Standardized ANN 10	10	ANN	4.19	0.97	0.65	0.54	0.59	0.91
Standardized	Standardized ANN 35	35	ANN	10.96	0.97	0.65	0.54	0.59	0.93
Standardized	Standardized BDK 10	10	BDK	2.54	0.98	0.68	0.54	0.60	0.94
Standardized	Standardized Cascor 10	10	Cascor	2.50	0.98	0.68	0.54	0.60	0.94
Standardized	Standardized Cascor 35	35	Cascor	6.26	0.98	0.68	0.54	0.60	0.96
Standardized	Standardized DKP 10	10	DKP	5.68	0.98	0.82	0.38	0.51	0.86
Standardized	Standardized DKP 35	35	DKP	11.69	0.98	0.76	0.54	0.63	0.95
Standardized	Standardized DPP 10	10	DPP	2.49	0.98	0.68	0.54	0.60	0.94
Standardized	Standardized DPP 35	35	DPP	5.95	0.98	0.68	0.54	0.60	0.96
Standardized	Standardized ELM-klm 10	10	ELM-klm	2.45	0.97	0.65	0.54	0.59	0.91
Standardized	Standardized ELM-klm 35	35	ELM-klm	6.15	0.97	0.62	0.54	0.58	0.91
Standardized	Standardized ELM-m 35	35	ELM-m	7.67	0.97	0.67	0.50	0.57	0.95
Standardized	Standardized LVQ 10	10	LVQ	2.50	0.98	0.68	0.54	0.60	0.94
Standardized	Standardized LVQ 35	35	LVQ	6.72	0.98	0.68	0.54	0.60	0.96
Standardized	Standardized MLP 10	10	MLP	3.16	0.98	0.68	0.54	0.60	0.97
Standardized	Standardized MLP 35	35	MLP	7.46	0.98	0.68	0.54	0.60	0.96
Standardized	Standardized MLP-wd 35	35	MLP-wd	7.06	0.97	0.65	0.54	0.59	0.92
Standardized	Standardized NN 10	10	NN	2.93	0.98	0.68	0.54	0.60	0.97
Standardized	Standardized NN 35	35	NN	7.58	0.98	0.72	0.54	0.62	0.96
Standardized	Standardized PCNN 10	10	PCNN	3.36	0.98	0.68	0.54	0.60	0.96
Standardized	Standardized PCNN 35	35	PCNN	7.83	0.97	0.65	0.54	0.59	0.95
Standardized	Standardized RBF 35	35	RBF	6.07	0.98	0.68	0.54	0.60	0.96

Scaler	Category	Epochs	Model	Time (min)	Accuracy	Precision	Recall	F1 Score	AUC
Standardized	Standardized RBF-dda 10	10	RBF-dda	2.53	0.98	0.68	0.54	0.60	0.94
Standardized	Standardized RBF-dda 35	35	RBF-dda	6.13	0.98	0.68	0.54	0.60	0.96
Standardized	Standardized RBFN 10	10	RBFN	2.59	0.98	0.68	0.54	0.60	0.94
Standardized	Standardized RBFN 35	35	RBFN	6.64	0.98	0.68	0.54	0.60	0.96
Robust	Robust RBF 10	10	RBF	2.67	0.98	0.68	0.54	0.60	0.94
Robust	Robust ANN 10	10	ANN	4.18	0.97	0.00	0.00	0.00	0.96
Robust	Robust ANN 35	35	ANN	11.26	0.95	0.35	0.46	0.40	0.95
Robust	Robust BDK 10	10	BDK	2.54	0.98	0.80	0.50	0.62	0.91
Robust	Robust Cascor 10	10	Cascor	2.57	0.98	0.80	0.50	0.62	0.91
Robust	Robust DKP 10	10	DKP	5.89	0.96	0.00	0.00	0.00	0.81
Robust	Robust DKP 35	35	DKP	12.73	0.97	0.58	0.46	0.51	0.97
Robust	Robust DPP 10	10	DPP	2.56	0.98	0.80	0.50	0.62	0.91
Robust	Robust ELM-klm 10	10	ELM-klm	2.54	0.98	0.85	0.46	0.59	0.81
Robust	Robust ELM-klm 35	35	ELM-klm	6.30	0.97	0.57	0.54	0.55	0.92
Robust	Robust ELM-m 10	10	ELM-m	3.52	0.98	0.83	0.42	0.56	0.93
Robust	Robust ELM-m 35	35	ELM-m	8.26	0.98	0.72	0.54	0.62	0.93
Robust	Robust LVQ 10	10	LVQ	2.68	0.98	0.80	0.50	0.62	0.91
Robust	Robust MLP 10	10	MLP	3.46	0.98	1.00	0.38	0.55	0.93
Robust	Robust MLP 35	35	MLP	7.13	0.98	0.90	0.38	0.53	0.96
Robust	Robust MLP-wd 10	10	MLP-wd	3.08	0.98	1.00	0.33	0.50	0.94
Robust	Robust MLP-wd 35	35	MLP-wd	7.82	0.97	0.55	0.46	0.50	0.96
Robust	Robust NN 10	10	NN	3.02	0.98	0.83	0.42	0.56	0.96
Robust	Robust NN 35	35	NN	6.99	0.98	0.75	0.50	0.60	0.94
Robust	Robust PCNN 10	10	PCNN	3.90	0.98	0.90	0.38	0.53	0.93
Robust	Robust PCNN 35	35	PCNN	7.44	0.98	0.75	0.50	0.60	0.94
Robust	Robust RBF 35	35	RBF	2.58	0.98	0.80	0.50	0.62	0.91
Robust	Robust RBF-dda 10	10	RBF-dda	2.49	0.98	0.80	0.50	0.62	0.91
Robust	Robust RBFN 10	10	RBFN	2.50	0.98	0.80	0.50	0.62	0.91
Non-Rescaled	Non-Rescaled ANN 10	10	ANN	4.28	0.97	0.00	0.00	0.00	0.86
Non-Rescaled	Non-Rescaled ANN 35	35	ANN	11.03	0.96	0.44	0.33	0.38	0.85
Non-Rescaled	Non-Rescaled BDK 10	10	BDK	2.58	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled BDK 35	35	BDK	6.26	0.97	0.59	0.54	0.57	0.96
Non-Rescaled	Non-Rescaled Cascor 10	10	Cascor	2.57	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled Cascor 35	35	Cascor	6.33	0.97	0.59	0.54	0.57	0.96
Non-Rescaled	Non-Rescaled DKP 10	10	DKP	5.70	0.97	0.80	0.33	0.47	0.66
Non-Rescaled	Non-Rescaled DKP 35	35	DKP	11.24	0.97	0.55	0.46	0.50	0.91
Non-Rescaled	Non-Rescaled DPP 10	10	DPP	9.22	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled DPP 35	35	DPP	6.24	0.97	0.59	0.54	0.57	0.96
Non-Rescaled	Non-Rescaled ELM-klm 10	10	ELM-klm	2.58	0.98	0.86	0.50	0.63	0.85
Non-Rescaled	Non-Rescaled ELM-klm 35	35	ELM-klm	6.19	0.97	0.57	0.54	0.55	0.89
Non-Rescaled	Non-Rescaled ELM-m 10	10	ELM-m	3.27	0.97	0.59	0.54	0.57	0.95
Non-Rescaled	Non-Rescaled ELM-m 35	35	ELM-m	7.79	0.97	0.54	0.54	0.54	0.92

Scaler	Category	Epochs	Model	Time (min)	Accuracy	Precision	Recall	F1 Score	AUC
Non-Rescaled	Non-Rescaled LVQ 10	10	LVQ	2.50	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled LVQ 35	35	LVQ	6.10	0.97	0.59	0.54	0.57	0.96
Non-Rescaled	Non-Rescaled MLP 10	10	MLP	3.12	0.98	0.82	0.38	0.51	0.93
Non-Rescaled	Non-Rescaled MLP 35	35	MLP	7.45	0.98	0.72	0.54	0.62	0.96
Non-Rescaled	Non-Rescaled MLP-wd 10	10	MLP-wd	3.03	0.97	0.67	0.42	0.51	0.88
Non-Rescaled	Non-Rescaled MLP-wd 35	35	MLP-wd	6.97	0.97	0.61	0.46	0.52	0.84
Non-Rescaled	Non-Rescaled NN 10	10	NN	2.97	0.97	0.61	0.46	0.52	0.93
Non-Rescaled	Non-Rescaled NN 35	35	NN	6.88	0.97	0.62	0.54	0.58	0.94
Non-Rescaled	Non-Rescaled PCNN 10	10	PCNN	3.05	0.97	0.69	0.46	0.55	0.93
Non-Rescaled	Non-Rescaled PCNN 35	35	PCNN	7.38	0.97	0.54	0.54	0.54	0.94
Non-Rescaled	Non-Rescaled RBF 10	10	RBF	2.50	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled RBF 35	35	RBF	6.11	0.97	0.59	0.54	0.57	0.96
Non-Rescaled	Non-Rescaled RBF-dda 10	10	RBF-dda	2.58	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled RBF-dda 35	35	RBF-dda	6.17	0.97	0.59	0.54	0.57	0.96
Non-Rescaled	Non-Rescaled RBFN 10	10	RBFN	2.47	0.97	0.62	0.54	0.58	0.92
Non-Rescaled	Non-Rescaled RBFN 35	35	RBFN	6.14	0.97	0.59	0.54	0.57	0.96

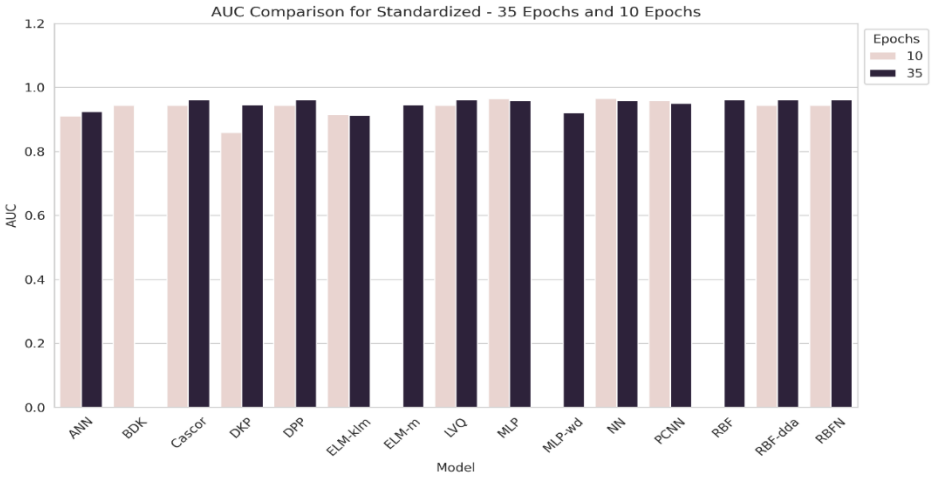

Figure 15: AUC Comparison for Standardized- 10 and 35 Epochs

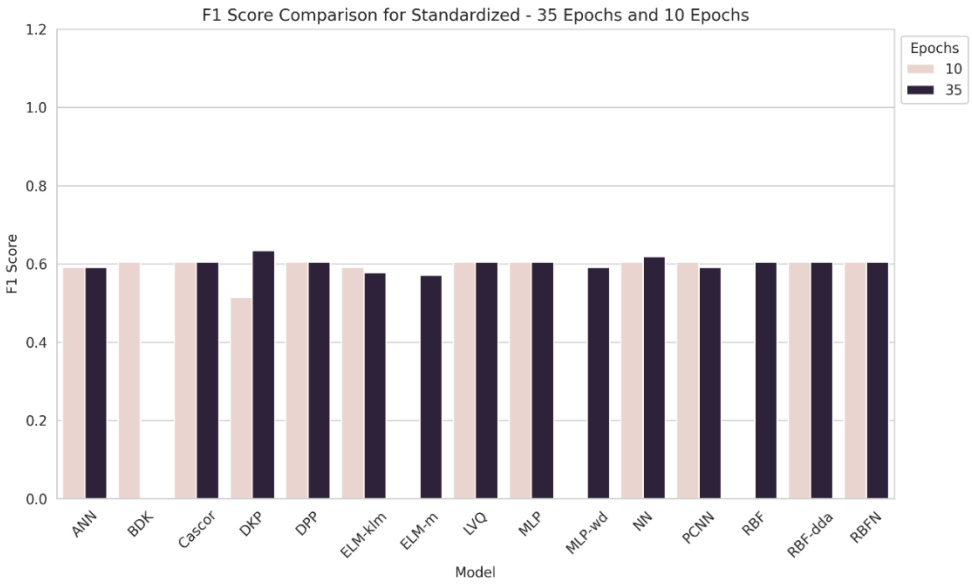

Figure 16: F1 Score Comparison for Standardized- 10 and 35 Epochs

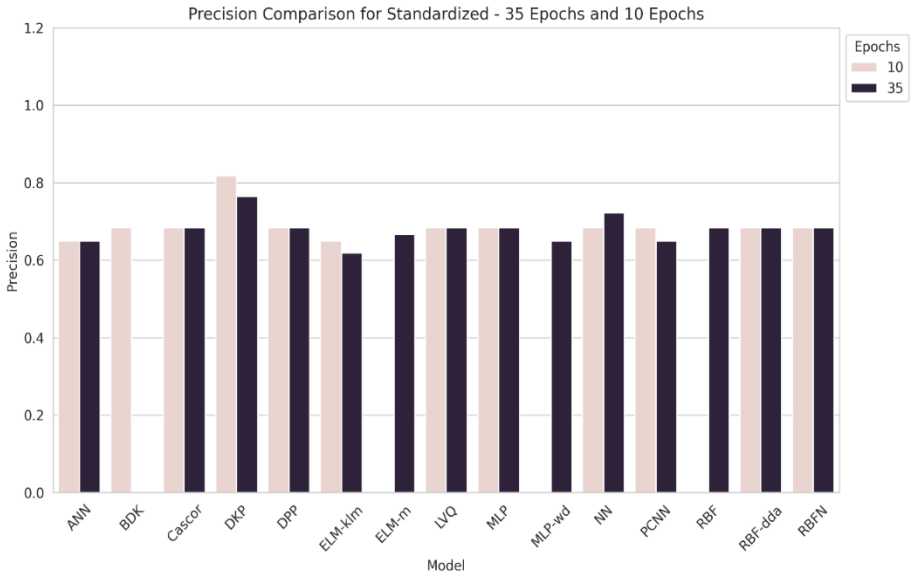

Figure 17: Precision Comparison for Standardized- 10 and 35 Epochs

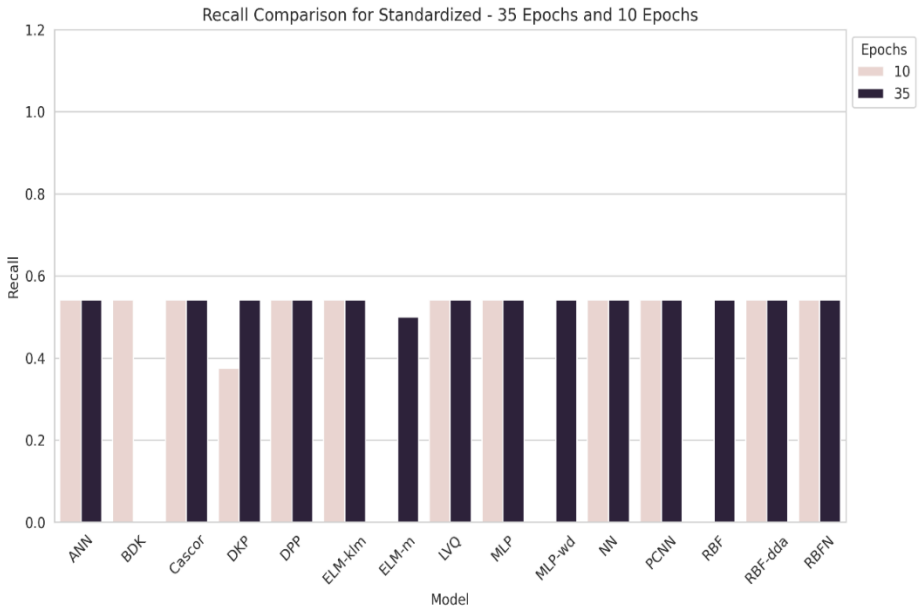

Figure 18: Recall Comparison for Standardized- 10 and 35 Epochs

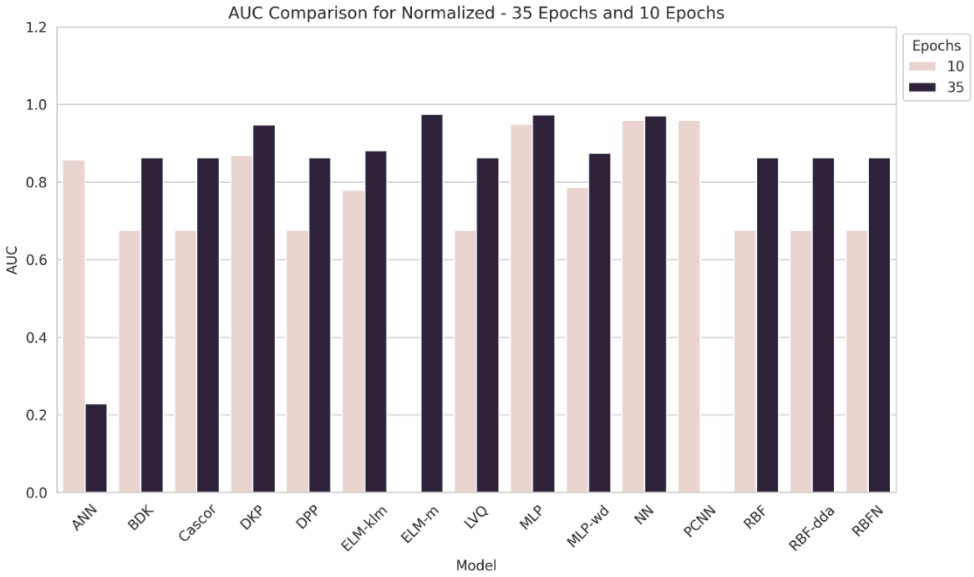

Figure 19: AUC Comparison for Normalized- 10 and 35 Epochs

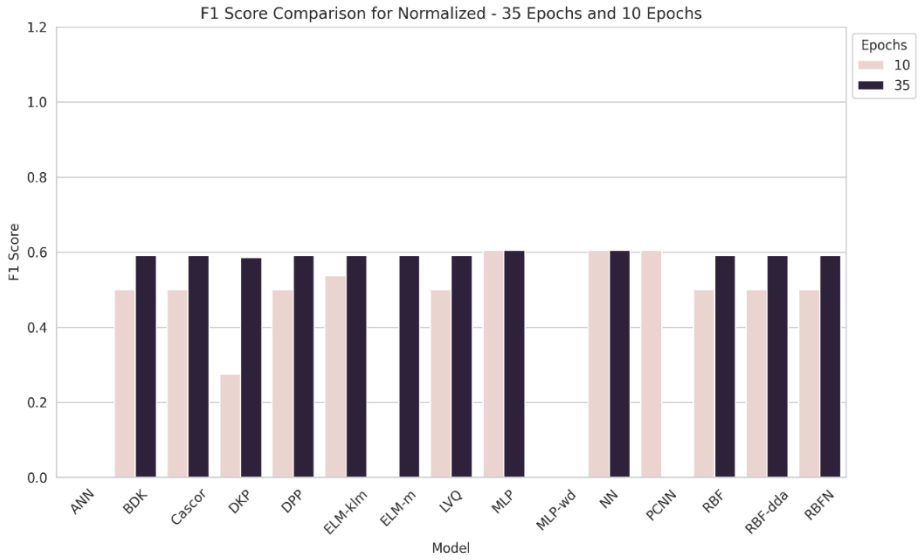

Figure 20: F1 Score Comparison for Normalized- 10 and 35 Epochs

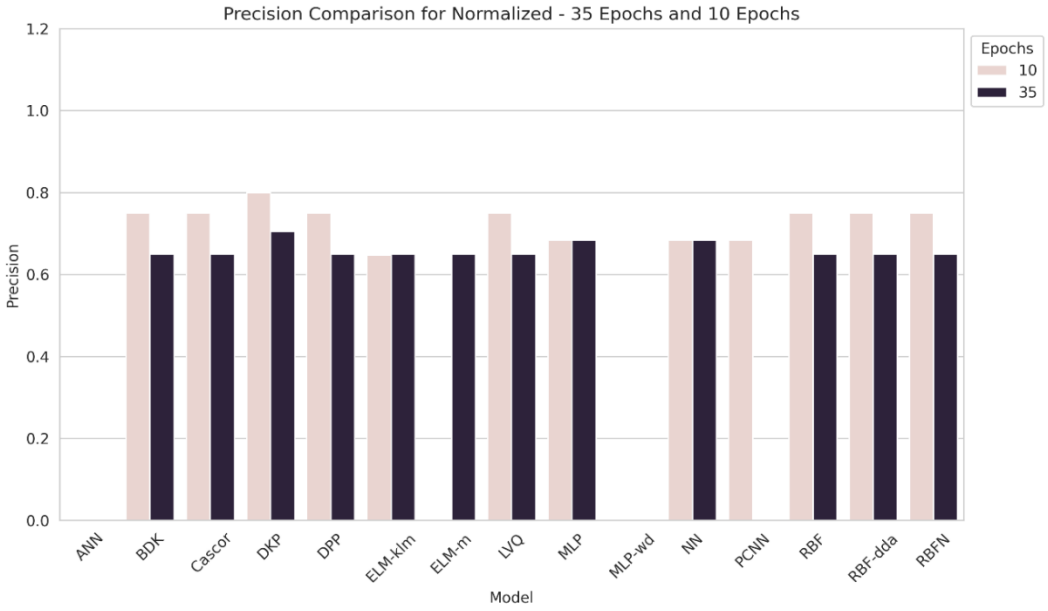

Figure 21: Precision Comparison for Normalized- 10 and 35 Epochs

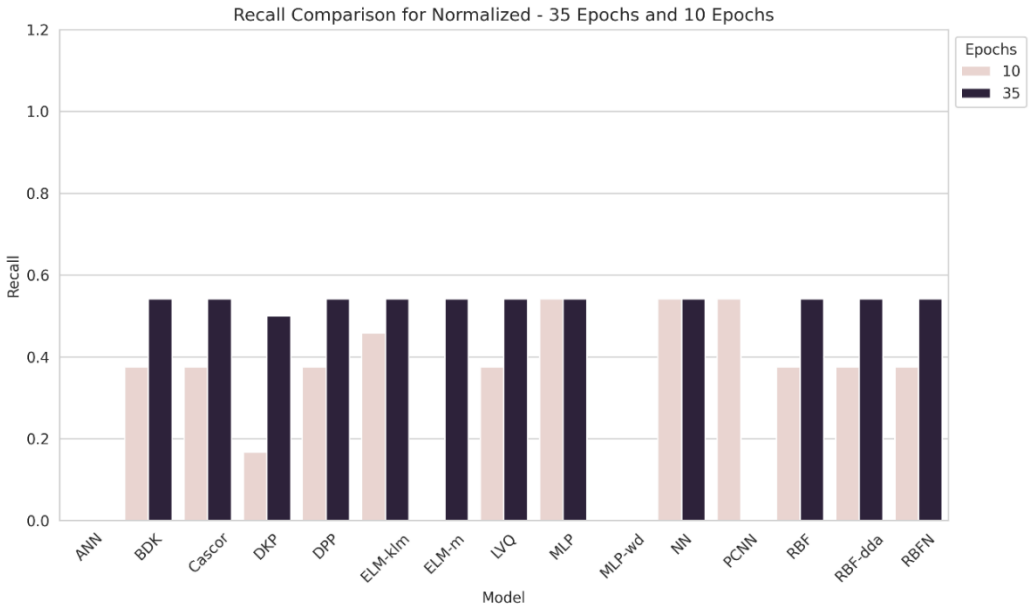

Figure 22: Recall Comparison for Normalized- 10 and 35 Epochs

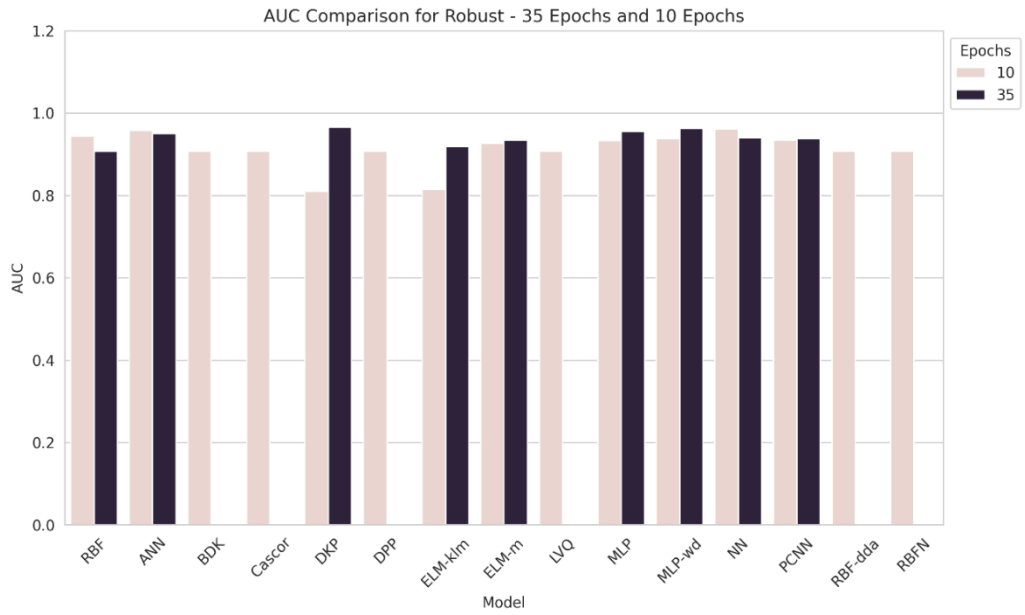

Figure 23: AUC Comparison for Robust- 10 and 35 Epochs

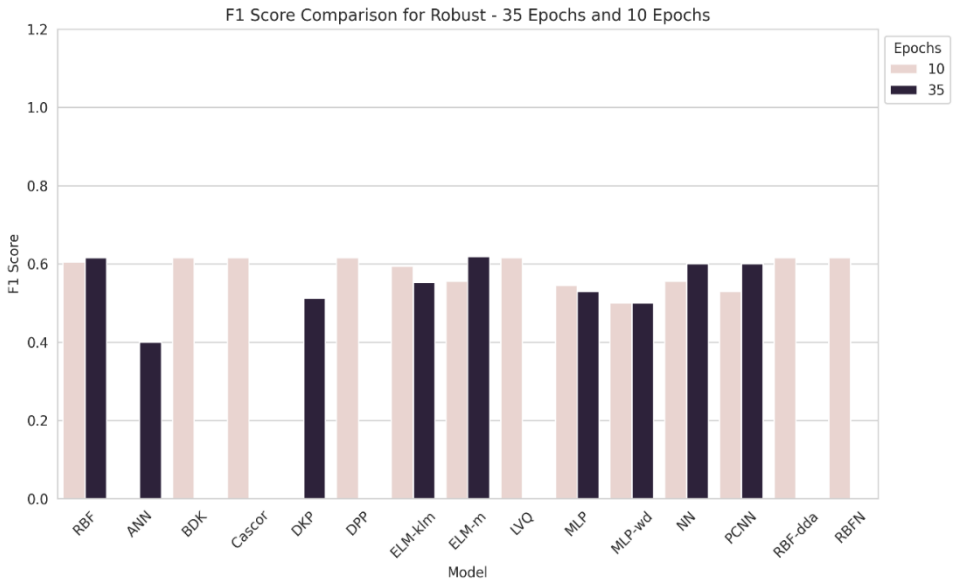

Figure 24: F1 Score Comparison for Robust- 10 and 35 Epochs

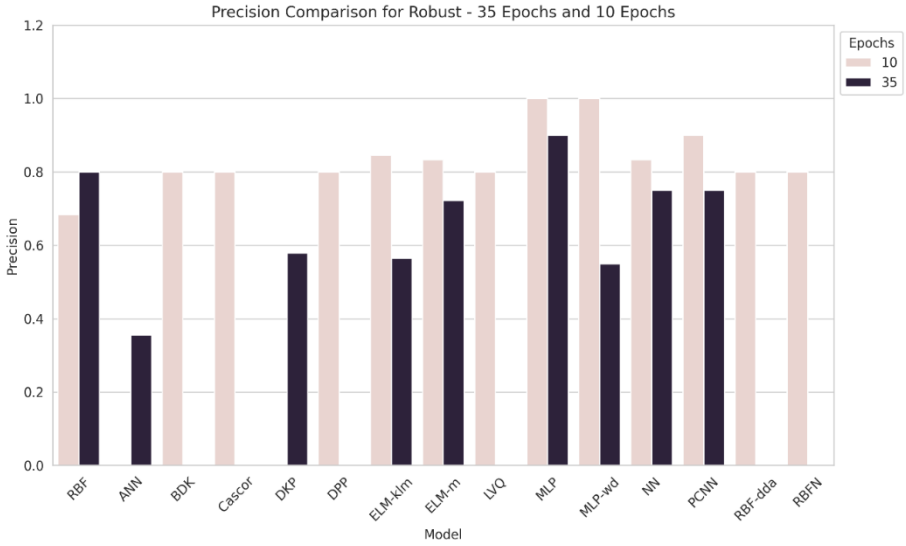

Figure 25: Precision Comparison for Robust- 10 and 35 Epochs

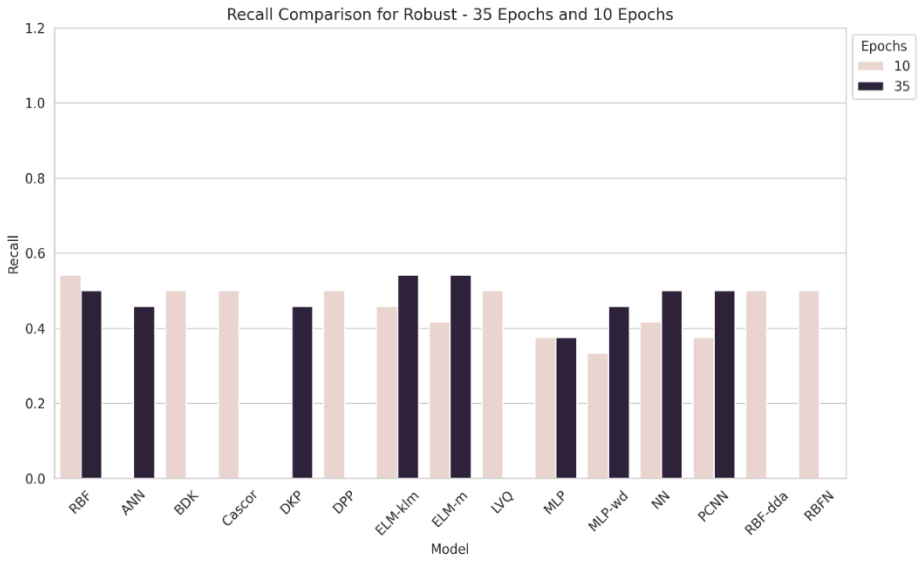

Figure 26: Recall Comparison for Robust- 10 and 35 Epochs

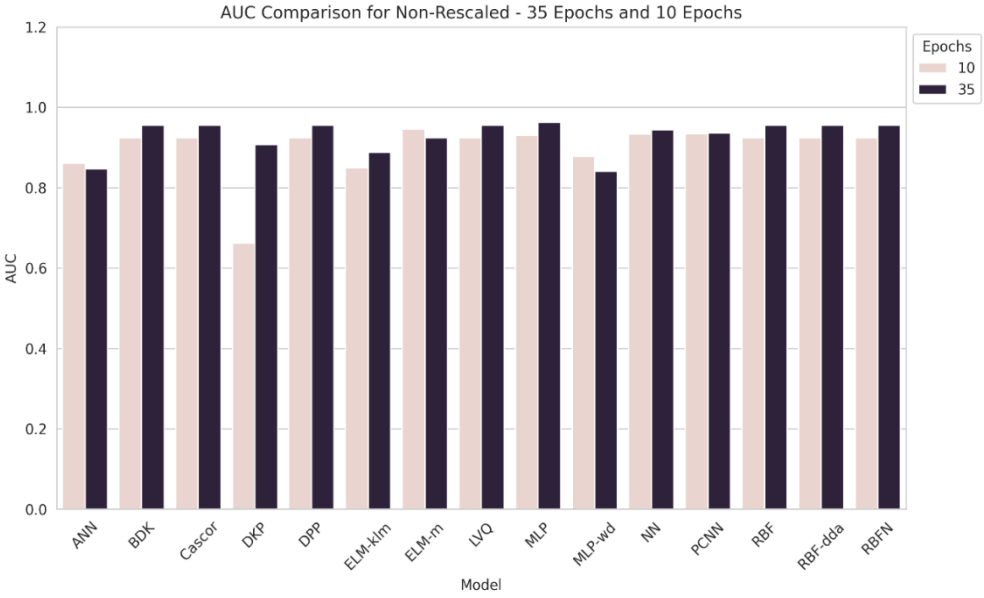

Figure 27: AUC Comparison for Non-Rescaled- 10 and 35 Epochs

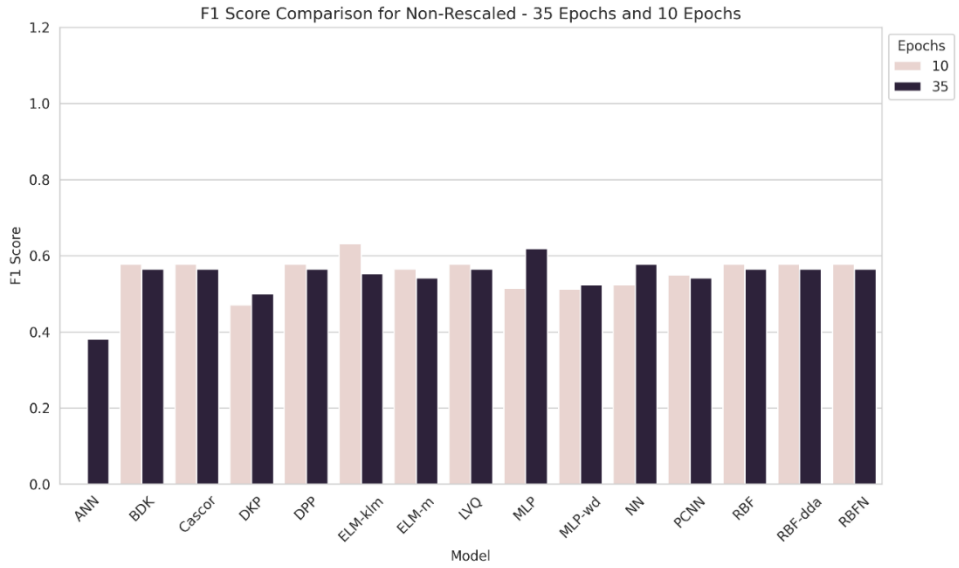

Figure 28: F1 Score Comparison for Non-Rescaled- 10 and 35 Epochs

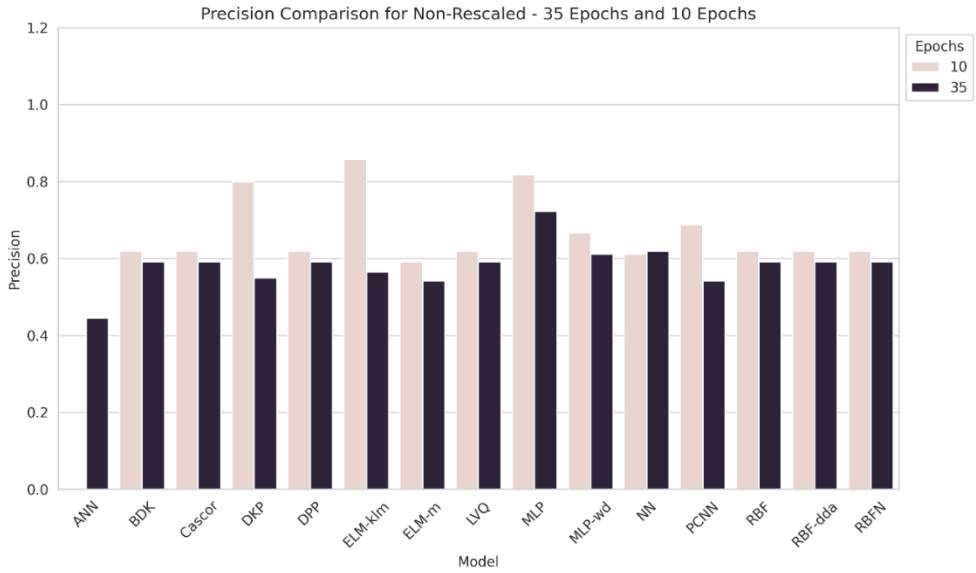

Figure 29: Precision Comparison for Non-Rescaled- 10 and 35 Epochs

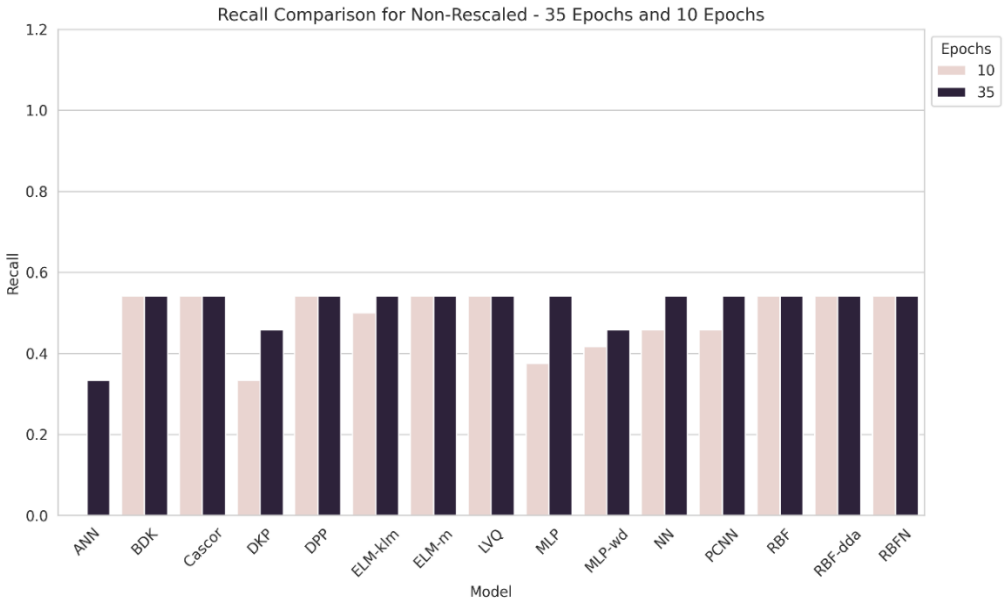

Figure 30: Recall Comparison for Non-Rescaled- 10 and 35 Epochs

Table 17: Average Speed Rank by Model

Model	Rank	Average Speed (s)
BDK	1	225.38
RBF-dda	2	245.04
RBF	3	245.87
Cascor	4	248.02
LVQ	5	248.97
RBFN	6	249.00
ELM-klm	7	262.74
DPP	8	301.10
NN	9	301.24
PCNN	10	309.26
MLP	11	322.81
MLP-wd	12	329.47
ELM-m	13	383.09
ANN	14	456.37
DKP	15	537.22

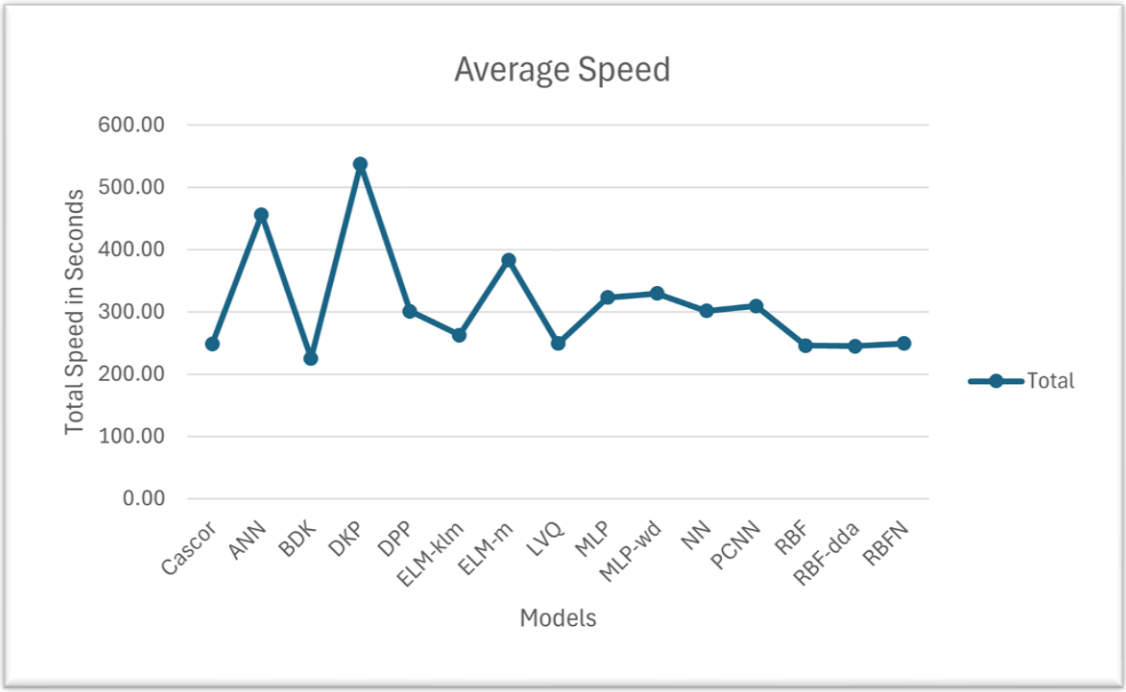

Figure 31: Average Speed by Model

Table 18: Praxis Top Performer comparison with Dr. Noever Top Seven Performers Rank Order by Accuracy (Noever, 2019)

Model	Data Treatment/ Test Category	Researcher	Rank Order	Mean Accuracy	Execution Time (seconds)
Random Forest	Over-Sampled	Noever	1	0.9835	210.56
Random Forest by Randomization	Over-Sampled	Noever	2	0.9831	200.16
K-Nearest Neighbors	Over-Sampled	Noever	3	0.9814	4.44
Localized Linear Discriminant Analysis	Over-Sampled	Noever	4	0.9806	114.64
Robust BDK 10	Dunn's Post hoc	Walter	5	0.9790	152.37
Robust Cascor 10	Dunn's Post hoc	Walter	6	0.9790	154.45
Robust DPP 10	Dunn's Post hoc	Walter	7	0.9790	153.42
Robust MLP 35	Dunn's Post hoc	Walter	8	0.9776	427.76
Robust ELM-m 10	Dunn's Post hoc	Walter	9	0.9776	211.08
Robust NN 10	Dunn's Post hoc	Walter	10	0.9776	181.01
Robust PCNN 10	Dunn's Post hoc	Walter	11	0.9776	234.14
Regularized Random Forest	Over-Sampled	Noever	12	0.9750	475.19
Bagged CART	Over-Sampled	Noever	13	0.9727	18.66
Logistic Model Trees	Over-Sampled	Noever	14	0.9683	49.61
Boosted Logistic Regression	Under-Sampled	Noever	15	0.9600	0.72
Random Forest by Randomization	Under-Sampled	Noever	16	0.9400	1.62
Logistic Model Trees	Under-Sampled	Noever	17	0.9200	1.28
Random Forest	Under-Sampled	Noever	18	0.9000	0.69
Regularized Random Forest	Under-Sampled	Noever	19	0.9000	0.83
Rule-Based Classifier (PART)	Under-Sampled	Noever	20	0.9000	0.86
J48 (C4.5)-like Trees	Under-Sampled	Noever	21	0.9000	1.78