NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLICATION OF ALGORITHM LEARNING TO IDENTIFY AND MITIGATE CONCEPT DRIFT

by

Nicholas T. Balk

March 2021

Thesis Advisor:	William Williamson
Second Reader:	James W. Scrofani

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE APPLICATION OF ALGORITHM LEARNING TO IDENTIFY AND MITIGATE CONCEPT DRIFT	**5. FUNDING NUMBERS**
6. AUTHOR(S) Nicholas T. Balk	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Data streams are becoming more numerous and complex, driven by an increased number of capable sensors. The complex, highly dimensional datasets created by these sensors contain information critical to our understanding of the battlefield situation. A significant change in the adversary's tactics, techniques, and procedures (TTPs) leads to a shift in the collected sensor data. The shift in the distribution of features in the data stream is known as concept drift, and this drift can be detected through predictive machine learning. We propose a novel drift detection method, named Reduced Dimensionality Drift Detection (RD3), based on dimensionality reduction to decrease feature space through supervised learning and unsupervised detection. Additionally, we show that concept drift can be mitigated after detection via an automated algorithm. We validate the performance of our novel method through comparison to a proven detection method, in similar trials and conditions, and show that RD3 performs comparably in concept drift detection and mitigation in all datasets evaluated.

14. SUBJECT TERMS concept drift; drift detection; ensemble learning; adaptive learning; classification; supervised learning; unsupervised learning; tactics, techniques, and procedures; TTPs; Reduced Dimensionality Drift Detection; RD3	**15. NUMBER OF PAGES** 111
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

APPLICATION OF ALGORITHM LEARNING TO IDENTIFY AND MITIGATE CONCEPT DRIFT

Nicholas T. Balk
Major, United States Marine Corps
BSCE, University of Nebraska at Lincoln, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
October 2021

Approved by: William Williamson
Advisor

James W. Scrofani
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Data streams are becoming more numerous and complex, driven by an increased number of capable sensors. The complex, highly dimensional datasets created by these sensors contain information critical to our understanding of the battlefield situation. A significant change in the adversary's tactics, techniques, and procedures (TTPs) leads to a shift in the collected sensor data. The shift in the distribution of features in the data stream is known as concept drift, and this drift can be detected through predictive machine learning. We propose a novel drift detection method, named Reduced Dimensionality Drift Detection (RD3), based on dimensionality reduction to decrease feature space through supervised learning and unsupervised detection. Additionally, we show that concept drift can be mitigated after detection via an automated algorithm. We validate the performance of our novel method through comparison to a proven detection method, in similar trials and conditions, and show that RD3 performs comparably in concept drift detection and mitigation in all datasets evaluated.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. OBJECTIVE....1
B. RELATED WORK....2
C. ORGANIZATION....3

II. BACKGROUND....5
A. DESCRIPTION OF CONCEPT DRIFT....5
B. TYPES OF CONCEPT DRIFT – EARLY DEFINITIONS....7
C. CONCEPT DRIFT DETECTION....7
D. DATA CLASSIFICATION AND CONCEPT DRIFT DETECTION....9
1. Training....9
2. Testing....10
E. ENSEMBLE OF CLASSIFIERS....10
F. DATASETS....10
1. Knowledge Discovery and Data Mining Cup '99 Dataset....11
2. Gradual Drift Profile....13
3. Individual Data Point Types and Classes....13
G. DATASET PREPROCESSING....15
1. Grouping Data Types....16
2. Data Label Extraction....16
3. One-Hot Encoding....16
4. Feature Scaling....17
5. Data Chunking....17
H. DRIFT MITIGATION DETAILS....18
1. Cued Retraining and Retraining Set Selection....18
2. Reduced Testing Window Size....19
I. BASELINE DRIFT DETECTION METHOD....19

III. CONCEPT DRIFT DETECTION AND MITIGATION ALGORITHMS....21
A. NOVEL METHOD: REDUCED DIMENSIONALITY DRIFT DETECTION....21
B. BASELINE METHOD: GAMA'S DRIFT DETECTION METHOD WITH MODIFIED THRESHOLDS....25

IV. IMPLEMENTATION....27

A. RD3: SUPERVISED TRAINING, UNSUPERVISED DETECTION....27
B. BASELINE METHOD: SUPERVISED TRAINING AND DETECTION....30

V. RESULTS....33
A. SELECTED METRICS....33
1. Accuracy Metrics....33
2. Precision Metric....34
B. TRIALS....34
C. DATA GATHERED TO SUPPORT METRICS ANALYSIS....36
1. Return to Accuracy....37
2. Test Set Drift Location....37
3. Initial Retrain Cycle....38
4. False Alarms....39
D. METRIC ANALYSIS....39
1. Accuracy Metrics Analysis....41
2. Precision Metric Analysis....58
E. COMPARATIVE ANALYSIS: PARAMETERS....60
F. COMPARATIVE ANALYSIS: METHODS....60

VI. CONCLUSIONS....63
A. CONCLUSION....63
B. FUTURE WORK....63

APPENDIX A. TYPES OF CONCEPT DRIFT—CURRENT DEFINITIONS....65
A. DRIFT MAGNITUDE....65
B. DRIFT DURATION....66
C. PATH LENGTH....66
D. DRIFT RATE....66
E. DRIFT SUBJECT....67
F. DRIFT FREQUENCY....67
G. DRIFT TRANSITION....67
H. DRIFT REOCCURRENCE....68

APPENDIX B. DATA CLASSIFICATION AND CONCEPT DRIFT DETECTION....69
A. SUPERVISED....69
1. Decision Tree....69
2. Random Forest....70

3. Support Vector Machine 71
4. k-Nearest Neighbor 71
5. Artificial Neural Network 72
B. UNSUPERVISED 73
C. SEMI-SUPERVISED 73

APPENDIX C. ATTACK CLASSES 75
A. PROBE ATTACK CLASS 75
B. DENIAL OF SERVICE ATTACK CLASS 75
C. USER TO ROOT ATTACK CLASS 75
D. REMOTE TO LOCAL ATTACK CLASS 75

APPENDIX D. EXECUTION TIME METRIC ANALYSIS 77
A. EXECUTION TIME METRIC 77
B. EXECUTION TIMES 77
C. RETRAIN TOTALS 77
D. EXECUTION TIME METRIC ANALYSIS 78

APPENDIX E: MODEL FLOWCHARTS 85

LIST OF REFERENCES 89

INITIAL DISTRIBUTION LIST 93

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Early Concept Drift Descriptions. Source: [3]. 7

Figure 2. The RD3 Process 22

Figure 3. RD3 Flowchart 28

Figure 4. Baseline Method Flowchart 31

Figure 5. Trial 1-F Initial Classification Accuracy Benchmark 45

Figure 6. Trial 1-F Final Classification Accuracy Scores 45

Figure 7. Trial 1-I Initial Classification Accuracy Benchmark 46

Figure 8. Trial 1-I Final Classification Accuracy Scores 47

Figure 9. Trial 1-K Initial Classification Accuracy Benchmark 48

Figure 10. Trial 1-K Final Classification Accuracy Scores 49

Figure 11. Trial 2-B Initial Classification Accuracy Benchmark 50

Figure 12. Trial 2-B Initial Detection Scores 51

Figure 13. Trial 2-B Final Classification Accuracy Scores 51

Figure 14. Trial 2-B Full Detection and Retraining Profile 52

Figure 15. Trial 2-H Initial Classification Accuracy Benchmark 53

Figure 16. Trial 2-H Initial Detection Scores 54

Figure 17. Trial 2-H Final Classification Accuracy Scores 54

Figure 18. Trial 2-H Full Detection and Retraining Profile 55

Figure 19. Trial 2-N Initial Classification Accuracy Benchmark 56

Figure 20. Trial 2-N Initial Detection Scores 56

Figure 21. Trial 2-N Final Classification Accuracy Scores 57

Figure 22. Trial 2-N Full Detection and Retraining Profile 57

Figure 23. Concept Drift Taxonomy. Source: [5]. 65

Figure 24. Decision Tree Example. Source: [31]. 70
Figure 25. Support Vector Machine Hyperplane Example. Source: [33]. 71
Figure 26. K-Nearest Neighbor Cluster Example ($k = 4$). Source: [35]. 72
Figure 27. Feed-Forward Artificial Neural Network Example. Source: [36]. 73
Figure 28. Baseline Method Flowchart 86
Figure 29. RD3 Flowchart....87

LIST OF TABLES

Table 1. KDD Dataset Feature List. Source: [15]. 11

Table 2. KDD Dataset Attack Types and Associated Classes 14

Table 3. Incremental Dataset Attack Types and Associated Classes 15

Table 4. RD3 Concept Drift Detection Thresholds and Triggers 24

Table 5. Baseline Method Concept Drift Detection Thresholds and Triggers 25

Table 6. Trial List 36

Table 7. Datasets and Related Drift Starting Locations 38

Table 8. Trial List with Results 40

Table 9. Trials Containing False Alarms 42

Table 10. Trials with Accurate Detection 42

Table 11. Trials Used to Assess Classifier Return to Accuracy 43

Table 12. Trial 1-F Data 44

Table 13. Trial 1-I Data 46

Table 14. Trial 1-K Data 48

Table 15. Trial 2-B Data 50

Table 16. Trial 2-H Data 53

Table 17. Trial 2-N Data 55

Table 18. Detection Metric Measurements 58

Table 19. Comparison of Select Trials 61

Table 20. Execution Time Measurements 78

Table 21. Initial Training Cycle Time Measurements 80

Table 22. Initial Test Cycle Time Measurements 82

Table 23. Total Program Execution Time Measurements 84

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ANN	Artificial Neural Network
DDM	Drift Detection Method
DOS	Denial of Service
DT	Decision Tree
KDD	Knowledge Discovery and Data Mining Cup ‘99
KNN	k-Nearest Neighbor
PCA	Principal Component Analysis
R2L	Remote to Local
RD3	Reduced Dimensionality Drift Detection
RF	Random Forest
SVM	Support Vector Machine
U2R	User to Root

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Data exists in a virtual environment, but millions of daily real-world decisions are made from that data. In the intelligence field, much of that information is gathered from an overwhelming and ever-increasing number of sensors and other data sources. Machine learning is essential to analyzing and understanding this information.

The military, and world at large, is increasingly dependent on machine learning. Many sources of military intelligence provide constant data feeds, commonly referred to as data streams. The distribution of the data representing the signal of interest and the background noise in these data streams are not always static. The information contained in the stream may change, and the change may affect machine learning algorithms in ways that are not immediately discernable. Machine learning models only allow us to understand data streams while the information in the data stream represents the data on which the models are trained.

Sensors monitor a situation and provide associated data streams. When the statistical distribution of the data elements that indicate the situation changes, information in the data stream changes, and the predictive capability of the model becomes less effective. Identifying changes in the data stream is the domain of the concept drift field of research.

If classification models are paired with effective concept drift detection algorithms, we are able to determine when something important in the statistics of the data stream has changed. The drift detection algorithm acts as a diagnostic mechanism to determine when a classification model is no longer effective and needs to be updated. The drift detection algorithms described in this thesis base their operations on changes in the data stream's underlying probabilistic distribution.

A. OBJECTIVE

The objective of our research is to develop an algorithm to automatically identify and mitigate concept drift in a data stream, and we have deliberately introduced drift into the datasets used in our research. We show the viability of a novel concept drift detection

method that monitors the feature space of datapoints assigned to the same class. We test our algorithm on multiple datasets containing different drift types. We apply an ensemble classifier supervised machine learning training model to a simulated drifting data stream and monitor the ensemble performance using an unsupervised concept drift detection scheme. Finally, we compare the performance of our novel detection and mitigation scheme against an established supervised detection method as our baseline and find that our unsupervised approach has comparable performance.

B. RELATED WORK

Concept drift detection is a complex problem, and numerous research papers focus on extending our understanding of the issue. Initial efforts describe drift over time as a result of non-stationary data properties [1] and outline the issues that non-stationarity causes when analyzing business and medical field data streams [2]. In many cases, the same researchers expand on the non-stationary issues and add modifications to established drift detection schemes [3].

Understanding what type of concept drift is contained in a dataset is important. Early definitions were relatively simplistic but used widely [3], [4]. The data streams studied in the literature have evolved and become more complex over time, as have the definitions of concept drift. Webb et al. proposed significantly more nuanced definitions [5]. Both the old and new definitions are included in this thesis.

The topic of concept drift is wide ranging and several surveys have been published [4]-[6]. Many papers focus on specific aspects of concept drift, including modifications to existing detection models [7], ensemble classification [8], and unique detection methods [9], [10]. One set of unique detection approaches focuses on Principal Component Analysis (PCA) [11] and distance measures [12], respectively. But no approach employing both concepts together has been proposed, which is the focus of our novel detection method.

Central to concept drift detection are the datasets to which detector algorithms are applied. While concept drift research is wide ranging, there are a few dataset types that occur frequently through many of the studies, which can be termed traditional datasets.

These traditional datasets are commonly drawn from several fields, including intelligence, computer security, telecommunications, finance, and medicine [1].

A few specific datasets have been studied multiple times in greater detail. Examples include the Australian electricity market price and French terrain satellite data [13], [14]. Another is the Knowledge Discovery and Data Mining Cup '99 (KDD) dataset [15], which was chosen for use in this thesis.

As a baseline of comparison for our novel approach we consider one relatively early and influential research paper written by Gama et al. [16]. Their research proposed a drift detection model based on directly checking the prediction accuracy of the classifiers being monitored. The Gama model utilized a supervised approach, comparing classifier output and labelled data to identify concept drift. This stream processing approach was adopted for our initial research work. Gama's supervised detection scheme is a best-case method and therefore provides a good benchmark from which to gauge our novel method.

C. ORGANIZATION

This thesis is organized into six chapters. Chapter II contains information describing concept drift and drift detection, as well as key concepts fundamental to employing our detection method. Chapter III describes the detection algorithms developed in our research, along with the supporting theory for each approach. Chapter IV covers the implementation and flow of each method. Chapter V details the results and analysis from implementing our methods. Chapter VI concludes the thesis with an overall summary of our objectives and work as well as recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

In this chapter we present the fundamental concepts and vocabulary of concept drift. We start with concept drift characterization, followed by an overview of the major approaches to concept drift detection. The remaining sections in this chapter describe the key concepts that apply to our detection and mitigation research.

A. DESCRIPTION OF CONCEPT DRIFT

Concept drift is the primary focus of this thesis. The terms "concept" and "concept drift" have specific meanings with respect to machine learning. A concept is a model representing the unknown and hidden relationship between input and output variables. A weather data concept is a good example, where the season is not identified in the temperature information but affects the temperature information.

In our research, a concept is the relation between variables contained in a data point to the identity of the data point. Specifically, the information in the data stream can be used to accurately predict a data point type or group. The concept is the probabilistic relation between the data point's variables and the data point's associated category.

A concept can be represented as a probabilistic relationship and defined mathematically. X represents the random variables contained in a set of vectors, or features, attributed to a data point, and Y is the random variable identifying the data point type, or label. X and Y variables for a data point are related, where $P(X)$ is the prior probability distribution over the labels, and $P(Y)$ is the distribution over covariates [5]. A concept, C, is defined in terms of joint distribution as [5]

$$C = P(X, Y). \tag{1}$$

Separate points in time are denoted by time t and time u. A concept at time t is defined as [5]

$$P_t(X, Y), \tag{2}$$

and a concept at time u is defined as [5]

$$P_u(X,Y). \tag{3}$$

Concept drift is described in numerous research papers and the exact definition tends to vary depending on the authors. The definition from the Machine Learning Encyclopedia [17] is appropriate for use in this thesis:

> Concept drift occurs when the values of hidden variables change over time. That is, there is some unknown context for concept learning and when that context changes, the learned concept may no longer be valid and must be updated or relearned. [17]

The hidden variables of the dataset are not invisible in the physical sense, but rather, their effect on the concept is not readily apparent. Concept drift occurs when there is a probabilistic distributional shift in the underlying feature data in a dataset, from one point in time to another. In other words, the previous concept relationship no longer represents the current concept. The concept has drifted from the previous relationship. The probabilistic definition of concept drift is given by [5]

$$P_t(X,Y) \neq P_u(X,Y), \tag{4}$$

where $P_t(X,Y)$ and $P_u(X,Y)$ are concepts at different points in time.

Machine learning algorithms are used to create models, and the models are used to analyze and understand robust datasets. The models are applied to datasets for predicting and classifying data points. When concept drift occurs in a dataset, the machine-learned classification models are unable to properly predict points in that set. The model must be updated with current information from the dataset.

Prediction models, created through machine learning, are a critical piece of our research. We do not assess the dataset probabilistic distribution changes directly, but rather we identify changes to the underlying data by analyzing changes in the feature space of classifier prediction results. If we are able to accurately identify changes in classifier behavior, we are able to detect and mitigate the effects of concept drift. After a prediction model is created, accurate analysis of the dataset feature space can be performed. As we will show, suitable analysis can be performed when the testing data point's classes are unknown as well.

B. TYPES OF CONCEPT DRIFT – EARLY DEFINITIONS

Early concept drift type descriptions are relatively simplistic compared to recent characterizations but are still used extensively [4]. The early definitions are briefly included here, as most current research efforts continue to use them. Recent drift descriptions are outlined in Appendix A.

Concept drift is a change in a data stream probabilistic distribution. Four categories were originally identified: Abrupt, Incremental, Gradual, and Reoccurring. Abrupt drift is a sudden change in distribution. Incremental drift is a slower, constant change in distribution. Gradual drift may appear random at first but grows unevenly until the new concept is reached. Reoccurring drift can include any of the three previous types of drift but will eventually return to the original concept distribution. Reoccurring drift may repeat at any time, in regular or variable frequencies. Figure 1 visually depicts the four drift types [3].

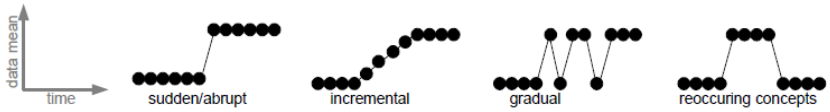

Figure 1. Early Concept Drift Descriptions. Source: [3].

C. CONCEPT DRIFT DETECTION

The probabilistic distribution in data streams rarely remains constant over time. The change in distribution could involve changes in the proportion of various classes, the emergence of new classes, or a change in the feature vectors associated with members of certain classes. As discussed above, these changes are related to types of concept drift. If the change in distribution is not detected and corrected for, classifiers operating on the data stream may be prone to error. For this reason, it is desirable to have a method to automatically identify and correct for the drift. Such methods are referred to as concept drift detection. Numerous methods to detect concept drift have been proposed. Many methods operate under a similar three-part process: A weighted training instance selection, classifier ensemble selection, and detection algorithm employment. While the general

process may be comparable between methods, each step allows for many possible implementations [4], [18].

A training instance should represent the concept currently contained within the test data stream. The training instance can come from recent or relatively old data. If the training set is weighted, it will include a mix of old and new data. Learning algorithms which place more weighting on older data include more old data in the set relative to newer data, which is also referred to as having more memory. Conversely, if there is no memory, the training set will include only the most recent data [4].

No machine learning algorithm works for every situation; therefore, an ensemble of classifiers is often used to develop prediction models. The ensemble can range from two similar classifier types to a pool of fundamentally different classifiers. Each classifier competes with the other classifiers in the ensemble to determine the best performing algorithm. Classifier selection affects drift detection capability. Furthermore, the metric used to determine the best performing classifier varies, depending on the drift detection mechanism [4]. For example, our baseline detection method uses prediction accuracy to determine the best classifier. Our novel method, which employs distance-based calculations to identify drift, utilizes classified point path length.

Detection algorithms are diverse, but the metrics used to determine drift can be grouped. For example, Gama et al. uses prediction accuracy scores to identify concept drift [16] while other types of detectors use Artificial Neural Networks (ANN) to estimate class distribution changes, which indicate concept drift [4]. Dimensional clustering is also used, where cluster density changes indicate concept drift via new classes. An example of dimensional clustering is the Online Novelty and Drift Detection Algorithm (OLINDDA) proposed by Spinoza et al. [19].

This thesis introduces a novel method obtained by combining dimensionality reduction and multi-dimensional distance measurements in the feature space of classified data points. We compare this to a baseline method which is a modification of the Gama et al. accuracy-based model [16]. Both methods are outlined in Chapter III.

D. DATA CLASSIFICATION AND CONCEPT DRIFT DETECTION

Machine learning classifiers are integral to our concept drift detection approach. We use the feature space describing the output of the classifiers as our primary indicator of drift; however, the detection and mitigation methods are the primary focus of this thesis rather than machine learning concepts and classification algorithms. A brief explanation of machine learning and classification fundamentals is provided in Appendix B, along with a description of the specific classifiers used in our research. This section focuses on the two-step process, training and testing, a classifier performs in the detection process. The process illustrates the important relationship between the features of data points associated with a class to the subsequent classification.

1. Training

Classifiers must be trained prior to predicting data points labels in the testing set. The training phase is commonly called fitting. This is accomplished by using the training set with the classifier's fit parameter to model the training data.

Initial, pre-concept drift classifier accuracy is a key metric in determining if drift occurs in the testing set. The trained model must be able to accurately predict tested data points for the current concept in order to determine if the concept changes later. There are two ways to ensure a model is suitably accurate: using a representative training set size and properly tuning the underlying algorithms.

For a model to predict test data points properly, the training set must represent the classes in the test set while also containing enough sample data points to train to each class. The best way to train with a representative dataset is to choose a large training set, from the current concept. A training set is too small if it does not represent the current concept class distribution or variation. However, an overly large training set could overfit a model, and the model may be inflexible when predicting test data points. Also, computer processing requirements increase significantly as training set sizes increase.

Each classification algorithm has several parameters which can be adjusted, commonly referred to as tuning. Properly tuning each algorithm is necessary to achieve initial classifier accuracy. The tuning must be done prior to training the algorithm, and

several iterations of adjustment may be needed for suitable fitting. But an algorithm can also be over-tuned to datasets. This means the algorithm settings are precisely correct for a specific dataset, in the given concept, but may not perform as well if trained with a different concept or dataset. Each algorithm used in this thesis was only tuned enough to achieve suitable accuracy for drift detection. The desire is to retain maximum utility across datasets, not to precisely tune to a specific dataset or concept.

2. Testing

Once the model has been created, each classification algorithm can begin testing the test dataset, using the prediction parameter. The algorithms evaluate one portion of the test data, or test chunk, at a time and sequentially processes each test chunk until complete. A vector is created for each successive chunk, containing a predicted classification for each data point.

E. ENSEMBLE OF CLASSIFIERS

There are many versatile classification algorithms available, able to handle a wide range of dataset types. As mentioned earlier, there are no universally effective algorithms [20]. One classifier may outperform other classifiers in the early iterations of a test dataset, while a different classifier may be superior later in the same dataset. Therefore, an ensemble of classifiers is used, competitively, to create the best performing detection algorithm. The criteria for selecting the best performing classifier depends on the drift detection method. For accuracy-based detectors, classifier accuracy scores are used. For distance-based detectors, the drift path length for each classifier is used.

F. DATASETS

Concept drift detection and mitigation algorithms are applied to data streams, or datasets in the case of our research. This section briefly describes the datasets used to create the simulated data streams considered in our research.

1. Knowledge Discovery and Data Mining Cup '99 Dataset

The KDD dataset group is commonly used in the concept drift research field. The KDD dataset is based on computer network intrusion attacks but was generated in a virtual environment. This means the attacks and related distributions are similar to real network data streams but are developed by simulation [15]. The original, full KDD file was used in this thesis. It is a labeled set, containing abrupt covariate concept drift. The KDD dataset contains 41 feature vectors, listed in Table 1.

Table 1. KDD Dataset Feature List. Source: [15].

Index	Feature name	Description
1	duration	Length of connection
2	protocol type	Type of protocol (TCP, UDP...)
3	service	Destination service (ftp, telnet...)
4	flag	Status of connection
5	source bytes	No. of B from source to destination
6	destination bytes	No. of B from destination to source
7	land	If the source and destination address are the same land=1/if not, then 0
8	wrong fragments	No. of wrong fragments
9	urgent	No. of urgent packets
10	hot	No. of hot indicators
11	failed logins	No. of unsuccessful attempts at login

Table 1. KDD Dataset Feature List (Continued). Source: [15].

Index	Feature name	Description
12	logged in	If logged in=1/if login failed 0
13	# compromised	No. of compromised states
14	root shell	If a command interpreter with a root account is running root shell=1/if not, then 0
15	su attempted	If an su command was attempted su attempted=1/if not, then 0 (temporary login to the system with other user credentials)
16	# root	No. of root accesses
17	# file creations	No. of operations that create new files
18	# shells	No. of active command interpreters
19	# access files	No. of file creation operations
20	# outbound cmds	No. of outbound commands in an ftp session
21	is hot login	is host login=1 if the login is on the host login list/if not, then 0
22	is guest login	If a guest is logged into the system, is guest login=1/if not, then 0
23	count	No. of connections to the same host as the current connection at a given interval
24	srv count	No. of connections to the same service as the current connection at a given interval
25	serror rate	% of connections with SYN errors
26	srv error rate	% of connections with SYN errors
27	rerror rate	% of connections with REJ errors
28	srv rerror rate	% of connections with REJ errors
29	same srv rate	% of connections to the same service
30	diff srv rate	% of connections to different services
31	srv diff host rate	% of connections to different hosts
32	dst host count	No. of connections to the same destination
33	dst host srv count	No. of connections to the same destination that use the same service
34	dst host same src rate	% of connections to the same destination that use the same service
35	dst host srv rate	% of connections to different hosts on the same system
36	dst host same srv port rate	% of connections to a system with the same source port
37	dst host srv diff host rate	% of connections to the same service coming from different hosts
38	dst host serror rate	% of connections to a host with an S0 error
39	dst host srv serror rate	% of connections to a host and specified service with an S0 error
40	dst host serror rate	% of connections to a host with an RST error
41	dst host srv serror rate	% of connections to a host and specified service with an RST error

2. Gradual Drift Profile

The KDD dataset works well for our research, as will be shown in Chapter V, but the KDD dataset contains only abrupt concept drift. Differing rates of drift are important when determining the viability of detection and mitigation algorithms. In order to test our approach for detection of incremental concept drift, a new artificial dataset was developed. The data generation algorithm produced controlled incremental concept drift, where the user could specify the concept drift rate. The algorithm was based on the KDD attack types, classes, and distributions.

The original KDD statistical distributions were calculated, along with attack type frequencies. The artificial information was generated in chunks, containing similar distributions and frequencies to the KDD dataset. The dataset was assembled based on the chunks, beginning with the original concept. At a designated point, chunks with increasing concept drift magnitude in the form of new attack types were added. The final dataset contained the original concept for a duration, followed by incremental concept drift. Two datasets were generated for use in our research. One dataset contained a relatively slow drift rate, and the other, a faster drift rate.

3. Individual Data Point Types and Classes

The datasets used in our research are based on network intrusion detection and contain similar data point types. The KDD dataset has 39 unique network attack types distributed among the normal network activity data points, while the incremental dataset has 26 unique attack types. The attacks types are grouped into four attack classes. The KDD attack types and associated classes are listed in Table 2, and the incremental types and associated classes are listed in Table 3.

Table 2. KDD Dataset Attack Types and Associated Classes

Attack Type	Class
apache2	dos
back	dos
land	dos
mailbomb	dos
neptune	dos
pod	dos
processtable	dos
smurf	dos
teardrop	dos
udpstorm	dos
worm	dos
ipsweep	probe
mscan	probe
nmap	probe
portsweep	probe
saint	probe
satan	probe
ftp_write	r2l
guess_passwd	r2l
httptunnel	r2l
imap	r2l
multihop	r2l
named	r2l
phf	r2l
sendmail	r2l
snmpgetattack	r2l
snmpguess	r2l
spy	r2l
warezclient	r2l
warezmaster	r2l
xlock	r2l
xsnoop	r2l
buffer_overflow	u2r
loadmodule	u2r
perl	u2r
ps	u2r
rootkit	u2r
sqlattack	u2r
xterm	u2r

Table 3. Incremental Dataset Attack Types and Associated Classes

Attack Type	Class
apache2	dos
back	dos
mailbomb	dos
neptune	dos
pod	dos
processtable	dos
smurf	dos
teardrop	dos
ipsweep	probe
mscan	probe
nmap	probe
portsweep	probe
saint	probe
satan	probe
guess_passwd	r2l
httptunnel	r2l
multihop	r2l
named	r2l
sendmail	r2l
snmpgetattack	r2l
snmpguess	r2l
warezmaster	r2l
buffer_overflow	u2r
ps	u2r
rootkit	u2r
xterm	u2r

Individual definitions of the attack types and classes are not necessary to understand this research, but a brief explanation of each class is provided in Appendix C.

G. DATASET PREPROCESSING

As mentioned earlier, real-world drift detection is applied to data streams. Datasets represent data streams in our research. Many datasets, when initially loaded into the programming environment, are in a format which is partially incompatible with the way the program accepts and calculates data. This section discusses the preprocessing necessary to prepare the data for the follow-on algorithm use.

1. Grouping Data Types

For machine learning training, data points can be grouped based on associated categories. For example, Table 3 contains numerous unique data attack types such "neptune" and "teardrop." Both attack types can be grouped in the "denial of service" class. Grouping in this sense can only be performed if data labels are present. If so, the current attack type label vector is replaced by a new grouped class label vector. Data points used in this thesis are grouped by attack class, versus the original individual attack types. A brief explanation of each associated class is located in Appendix C.

2. Data Label Extraction

If the dataset used in the algorithm contains data labels, the label vector is removed from the dataset matrix prior to training. Supervised learning models train concurrently with the training set and labels, but they are processed separately. The label vector may be discarded if unsupervised learning is used.

3. One-Hot Encoding

A common issue encountered in diverse datasets is a mix of string type and integer type vectors. In other words, there is a mix of feature vectors containing letters and other feature vectors containing numbers. The KDD dataset is an example of mixed types. Many classification algorithms process integer type vectors only.

To resolve the data type incompatibility, the strings are converted to binary values via one-hot encoding programming [21]. The advantage of one-hot encoding is that the converted vectors do not numerically skew the dataset features. The process of one-hot encoding replaces the original string vector with numerous integer vectors. The number of replacement vectors created depends on the number of unique string values in the original vector. A numeric 1 representing a unique string is placed is its associated vector, with a 0 in all others replacement vectors, which occurs for all unique strings. Therefore, each unique string is represented binarily, with scaling occurring automatically in the process.

Commonly, the label vector is in string format, but does not need to be one-hot encoded. The encoding is not needed because the comparison portion of the program is able to process the strings, in the form of words in this case.

4. Feature Scaling

Diverse datasets contain many feature vectors, and data values can vary widely within the vectors. This applies to the datasets used in this thesis. For example, the "count" feature tallies the number of connections to a host, which ranges from one to hundreds of connections, and the "duration" feature measures the time connected to a host, which ranges from zero to thousands of seconds. Uneven data spread among the feature vectors cause many classification algorithms to perform poorly [22]. To resolve this issue, each feature vector is standardized by removing the mean and scaling to unit variance [23].

5. Data Chunking

The final preprocessing step is data chunking. Each test chunk extracted from the dataset is assessed as a unit, and the chunk size represents the "time window" of the data. Larger chunks represent more time in a data stream than smaller chunks. The chunk size is also mathematically significant, as larger and smaller chunks provide larger and smaller sample sizes, respectively. Selecting the appropriate data chunk size is crucial to detecting concept drift.

If test chunks are assessed for classification accuracy, the chunk size affects the accuracy score average. Larger chunks contain more data points, while smaller chunks contain less. If the chunk is too large, it may proportionally lower the error rate, and raise the accuracy score. If the chunk is too small, the error rate may be inflated, and accuracy lowered. For example, a chunk size with 500 data points and one classification error has twice the error rate compared to a chunk size with 1000 data points and one error.

The same concept applies to changes and variations in the dataset. Large chunks may wash out significant change trends. Conversely, chunks that are too small may exaggerate data variance and outliers.

Improper chunk size selection directly affects drift detection triggering. Overly large chunks may miss the instances where warnings or detections should be triggered. Chunks that are too small may detect drift where there is none.

Chunk size directly impacts concept drift detection timing. Since each test chunk is assessed as a unit, drift can only be detected once the entire chunk is processed. If the chunk associated with the time window is small, drift occurring in that chunk will be detected with finer time resolution. Naturally, larger chunks mean longer time to detect and a wider timeframe to consider where drift occurred. Thus, we have a trade-off in performance based on chunk size where we trade our ability to determine *whether* drift is occurring against our ability to determine *where* drift is occurring.

Data chunking is a two-step process. First, the dataset is split into training and testing sections. The split is determined by the initial user designated training set size value. For this thesis, the training set is taken from the beginning of the dataset, starting at the first data point and ending at the data point designated by the training size value.

Second, the testing section is split into chunk sizes based on the user designated chunk size value. The chunk sizes represent the number of data points designated to form a testing unit. Real-world applications normally draw from a data stream, where data points accumulate until reaching the test chunk value and would subsequently be tested. The dataset used in this thesis is finite; hence, the total testing section is uniformly divided into the designated chunk sizes, with the last chunk as the remainder of the division. The same process is applied to the label vector, if present.

H. DRIFT MITIGATION DETAILS

Concept drift mitigation involves retraining the classifiers to update the associated prediction models. There are modifications to the training set and testing windows following a mitigation cycle. This section outlines those changes.

1. Cued Retraining and Retraining Set Selection

The classifier algorithms provide the statistical data to evaluate for concept drift, but the drift detection algorithm triggers retraining. The drift detection algorithm will automatically retrain the classifier models if detection criteria is reached.

The retraining dataset selection is based on the concept drift detection trigger. The new training set is pulled from the retraining point and immediately preceding data. For

example, a mitigation cycle may be triggered on the 100th test chunk. The retraining set is taken from the 100th test chunk and preceding testing set, such as the 95th through the 99th test chunks. The retraining algorithm uses data labels, simulating a use case where a human would be cued to inspect the last few datasets because of the suspected drift.

Additionally, classification and testing for drift will continue on the test data immediately following mitigation. From the previous example, mitigation occurs at the 100th test chunk. Once the prediction model is updated in this case, testing will resume at test chunk 101.

2. Reduced Testing Window Size

The test chunk is the representation of the testing window. There are two testing window sizes specified by the user. The first is the initial test window size, which is used by the classification and drift detection algorithms until concept drift is detected. The second testing window size, used after retraining is cued, should be smaller than the initial window in order to reduce detection time delay and improve accuracy [3], [16], [18], [24]. The detection algorithm becomes more sensitive to changes and reacts more quickly to drift in smaller windows, which is good practice if concept drift has already occurred and further drift is expected.

I. BASELINE DRIFT DETECTION METHOD

In order to evaluate the performance of our data-driven unsupervised drift detection approach, we established a baseline of supervised drift detection. The baseline detection method is a modification to a model that Gama et al. proposed in 2004 [16]. This section will outline the original method, and our modifications will be explained in the Chapter III.

The baseline model derived from Gama is an accuracy-based concept drift detection method (DDM). It uses classification accuracy scores measured against labeled data to detect concept drift. Both the training and testing data points have labels. The classification model is developed through supervised training. The model is used to classify test data points, and the predicted classes are compared to the actual labeled classes.

Gama et al. developed the detection metrics used in this method. The test set is separated into chunks, which represent each test window containing a given number of data points. The error probability per chunk is used to determine the standard deviation of the prediction error. Both the error probability and standard deviation are monitored throughout the detection process.

Several variables must be defined before outlining the formulas Gama used in his original method. The variable i is the point set in a given data sequence, such as the "i^{th} chunk." The variable p_i is the probability of misclassification, or the classification error rate. The variable s_i is the standard deviation. The variable p_{min} is the minimum value of the running p_i value. The variable s_{min} is the minimum value of the running s_i value [16].

The standard deviation for the prediction error in the i^{th} chunk is given by [16]

$$s_i = \sqrt{p_i(1-p_i)/i}\,. \tag{5}$$

The p_{min} value changes to $p_{min} = p_i$ and the s_{min} value changes to $s_{min} = s_i$ if [16]

$$p_i + s_i < p_{min} + s_{min}\,. \tag{6}$$

The original detection scheme contains two tiers [16]. The first is the warning level, which notifies the user that concept drift has possibly occurred within the test set. The warning level threshold formula is given by [16]

$$p_i + s_i \geq p_{min} + 2s_{min}\,. \tag{7}$$

The second threshold is the detection level. This threshold notifies the user that concept drift has occurred within the dataset. The detection level threshold formula is given by [16].

$$p_i + s_i \geq p_{min} + 3s_{min}\,. \tag{8}$$

III. CONCEPT DRIFT DETECTION AND MITIGATION ALGORITHMS

In this section we describe the details of our novel approach to drift detection, as well as the modifications made in the baseline approach. The baseline approach establishes an upper-bound on our expected performance, which relies on knowing the ground truth labels for each data point and directly measures classification accuracy. In practice, this level of knowledge is often not available. Our novel approach does not rely on ground truth labels and may be implemented in many practical situations. It is significant therefore that our novel unsupervised method compares favorably with the baseline method. From this point, the two drift detection methods will be referred to as the "Baseline Method" for the modified version of Gama's original DDM, and Reduced Dimensionality Drift Detection (RD3) for our novel detection algorithm. Both methods use statistical data analysis to determine if concept drift occurs in a dataset [25], and the remainder of this chapter focuses on explaining the algorithms for each method.

A. NOVEL METHOD: REDUCED DIMENSIONALITY DRIFT DETECTION

We developed the RD3 Method as a novel, unsupervised drift detection algorithm that looks directly at the characteristics of the incoming data stream. No data labels are needed during in the drift detection process. A PCA algorithm, applying dimensionality reduction, is combined with Euclidean distance calculations of the feature space to determine shifting cluster and group sizes [24], [26].

Previous methods utilizing PCA algorithms have been proposed or developed, but such methods did not use Euclidian distance in feature space to detect concept drift. For example, one method mentions the promise of PCA algorithms, but does not develop associated detection means [12]. Other research develops a PCA detection scheme but focuses on computationally intensive eigenspace transformations [11].

RD3 uses supervised training to develop the classification model, similar to the Baseline Method. RD3, however, takes an unsupervised approach to assess test data for

concept drift. The test data does not need to be labeled. Figure 2 contains the steps of the RD3 detection process, which the rest of this section will outline.

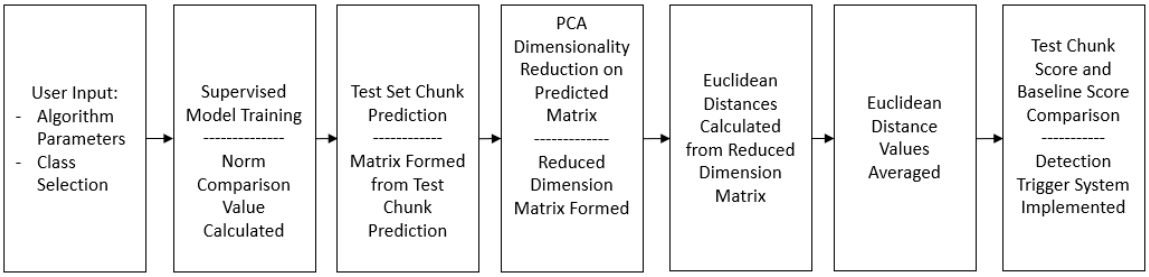

Figure 2. The RD3 Process

In RD3, a single class at a time is selected for concept drift detection. The selected class data points are extracted from the training set and used to fit the PCA algorithm. Supervised training is used to develop the classifier ensemble models. The classification algorithms models are used to predict classes for each data point in each test chunk. The configuration for the i^{th} test chunk matrix in the test data series is given by

$$test\_chunk_i = \begin{bmatrix} x1_{11} & x2_{21} & \dots & xn_{m1} \\ x1_{12} & x2_{22} & \dots & xn_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ x1_{1n} & x2_{2n} & \dots & xn_{mn} \end{bmatrix}, \tag{9}$$

where x is a feature vector, n is the number of feature vectors, and m is the number of data points contained in the i^{th} chunk.

The features of the data points with the predicted class that matches the RD3 selected class are extracted and formed into a separate matrix, with one class matrix for each test chunk. The matrix for the respective test chunk is then transformed via the fitted PCA algorithm, and the mean of each vector is retranslated to the origin. The number of PCA vectors are predetermined by user input of the PCA loading coding. Three PCA vectors are used for all testing in our research.

A transformed PCA vector example for the i^{th} test chunk is given by

$$PCA\_vector_i = \begin{bmatrix} PC1_{11} & PC2_{21} & PC3_{31} \\ PC1_{12} & PC2_{22} & PC3_{32} \\ \vdots & \vdots & \vdots \\ PC1_{1j} & PC2_{2j} & PC3_{3j} \end{bmatrix}, \quad (10)$$

where PC is a principal component vector and j is the total number of classified points.

The Euclidean distance from the mean is calculated for each respective PCA vector point, followed by the mean calculation for all Euclidean distances in the given chunk. The final value, $e\_dist\_avg_i$, represents the i^{th} test chunk's average Euclidean distance value.

The PCA vector, $PCA\_vector'_i$, with the mean, M, at the origin is given by

$$\begin{bmatrix} PC1_{11} - M & PC2_{21} - M & PC3_{31} - M \\ PC1_{12} - M & PC2_{22} - M & PC3_{32} - M \\ \vdots & \vdots & \vdots \\ PC1_{1m} - M & PC2_{2m} - M & PC3_{3m} - M \end{bmatrix} = \begin{bmatrix} PC1'_{11} & PC2'_{21} & PC3'_{31} \\ PC1'_{12} & PC2'_{22} & PC3'_{32} \\ \vdots & \vdots & \vdots \\ PC1'_{1m} & PC2'_{2m} & PC3'_{3m} \end{bmatrix}, \quad (11)$$

the formula for the Euclidean distance of each data point in a test chunk is given by

$$\begin{bmatrix} \sqrt{PC1'^2_{11} + PC2'^2_{21} + PC3'^2_{31}} \\ \sqrt{PC1'^2_{12} + PC2'^2_{22} + PC3'^2_{32}} \\ \vdots \\ \sqrt{PC1'^2_{1m} + PC2'^2_{2m} + PC3'^2_{3m}} \end{bmatrix} = \begin{bmatrix} e\_dist_1 \\ e\_dist_2 \\ \vdots \\ e\_dist_m \end{bmatrix}, \quad (12)$$

and the distance average is given by

$$e\_dist\_avg_i = \frac{1}{m}\sum_{k=1}^{m} e\_dist_k . \quad (13)$$

Each test data chunk from the dataset is assessed in this fashion and given its own average Euclidean distance value.

We calculate a norm value for Euclidean distance against which we compare and assess each test chunk Euclidean distance value point. For the initial detection sequence, the norm value is calculated from the training set. The initial norm is given by

$$e\_norm_{initial} = e\_dist\_avg_{train\_set} . \quad (14)$$

If concept drift is detected, the model is retrained and the initial principal components must be recalculated. The post-retraining norm value is derived from the mean of the first five Euclidean distance point values of the current test dataset. In this case, the decision to use five values was based on empirical experience gained from repeated testing trials, which suits the streaming data approach. The norm value following retraining is given by

$$e\_norm_{retrain} = \frac{1}{5}\sum_{k=1}^{5} e\_dist_k \,. \tag{15}$$

Detection values early in the dataset are relatively close to the norm value, provided concept drift does not occur immediately after retraining. If the data features contain any significant noise or other variance, the percentage comparison values may fluctuate sharply. A moving average filter, set to five points, is implemented to smooth out the fluctuations, which also aligns with the five-point retraining norm value.

The comparison of this final calculated vector, in absolute value form, is the basis for threshold detection. For drift detection, we create trigger counts where warning and detection levels have a set number of occurrences before retraining is triggered. We also create a threshold termed massive drift. Table 4 contains the standard RD3 threshold formulas and triggers used in our research.

Table 4. RD3 Concept Drift Detection Thresholds and Triggers

Threshold	**Formula**	**Trigger Count**
Warning	Pct diff from norm value ≥ 20%	10 (adjustable)
Detect	Pct diff from norm value ≥ 35%	5 (adjustable)
Massive	Pct diff from norm value ≥ 50%	Immediate (on/off)

RD3 trigger counts for the warning and detection thresholds are adjustable. The massive drift threshold can be turned on or off. Also, the formula values can be adjusted

relatively easily. This provides the algorithm flexibility to adjust to different types of datasets.

A concept drift detection run starts with the initial settings. Once a threshold and trigger count combination is reached, the algorithm retrains the model using data labels and retests the remaining dataset using the updated parameters and reduced test chunk size.

B. BASELINE METHOD: GAMA'S DRIFT DETECTION METHOD WITH MODIFIED THRESHOLDS

The Gama DDM was outlined in Chapter II, and our method makes two significant modifications to the originally published approach. First, Gama's original approach declares concept drift in a binary fashion. A drift warning indication is proposed, but drift detection is decided in one instance, suggesting immediate retraining. In our modifications to the original Gama model, we implement trigger counts as mentioned in the previous section, where warning and detection levels have a set number of occurrences before retraining is triggered. We also implement the massive drift trigger in the modified approach. Table 5 outlines the criteria for each threshold.

Table 5. Baseline Method Concept Drift Detection Thresholds and Triggers

Threshold	**Formula**	**Trigger Count**
Warning	$p_i + s_i \geq p_{min} + 2s_{min}$	10 (adjustable)
Detect	$p_i + s_i \geq p_{min} + 3s_{min}$	5 (adjustable)
Massive	$p_i + s_i \geq p_{min} + 5s_{min}$	Immediate (on/off)

Our Baseline Method trigger counts for warning and detection thresholds are adjustable. Also, the detection threshold trigger iterates the warning count, meaning that a drift detection trigger also adds to the warning trigger count. Massive drift detection triggers immediate retraining but may be turned off if desired.

Within the code, the individual threshold values can be adjusted. The modifications to the original method have greater flexibility in this case. As noted in early drift detection

research, the basic threshold values may miss slow and gradual drift. Therefore, the model may be adjusted to increase sensitivity [7]. The trigger count numbers for each level were not chosen randomly. They were selected based on experience gained from trial runs during code development.

Prior to any retraining, the Baseline Method uses normal test chunk sizes. Once a threshold and trigger count combination is reached, the algorithm switches to a retraining and retesting cycle. The retraining test chunk size is adjustable and should be smaller than the pre-concept drift detection test set size, mentioned in Chapter II. From this point, the algorithm will continue retraining and retesting at the adjusted test chunk size.

IV. IMPLEMENTATION

Chapter III contained the theory and related formulas for the RD3 drift detection algorithm and the baseline algorithm. This chapter will detail the implementation process for each method, with the focus on detection and mitigation.

Concept drift detection and mitigation strategies are central to this thesis. The classifiers provide the data point predictions, and each detection strategy assesses if concept drift occurs in the test data. If concept drift mitigation is needed, the detection algorithm passes the relevant information to the mitigation algorithm. Each method follows the same general process: Train the classification models, use them to classify the dataset, assess for concept drift, retrain the model if concept drift is present, and repeat the process, as necessary.

There are several additional steps in the process which are not covered in this chapter, such as initial user input and detailed data processing. Appendix E contains flowcharts for each method, with a broader view of the algorithms. Chapter II can also be referenced for additional details related to the overall process.

We begin each method at the data-preprocessing and initial model training phase, and end at the summary output phase.

A. RD3: SUPERVISED TRAINING, UNSUPERVISED DETECTION

Figure 3 contains a flowchart for the RD3 detection and mitigation process which can be referenced throughout this section.

Once the classification algorithm has been trained, an initial model is developed. The model is used to evaluate the test set and predict the data point types in each test chunk. The PCA algorithm has been fitted and separately, a norm value derived from the training set data during this phase as well.

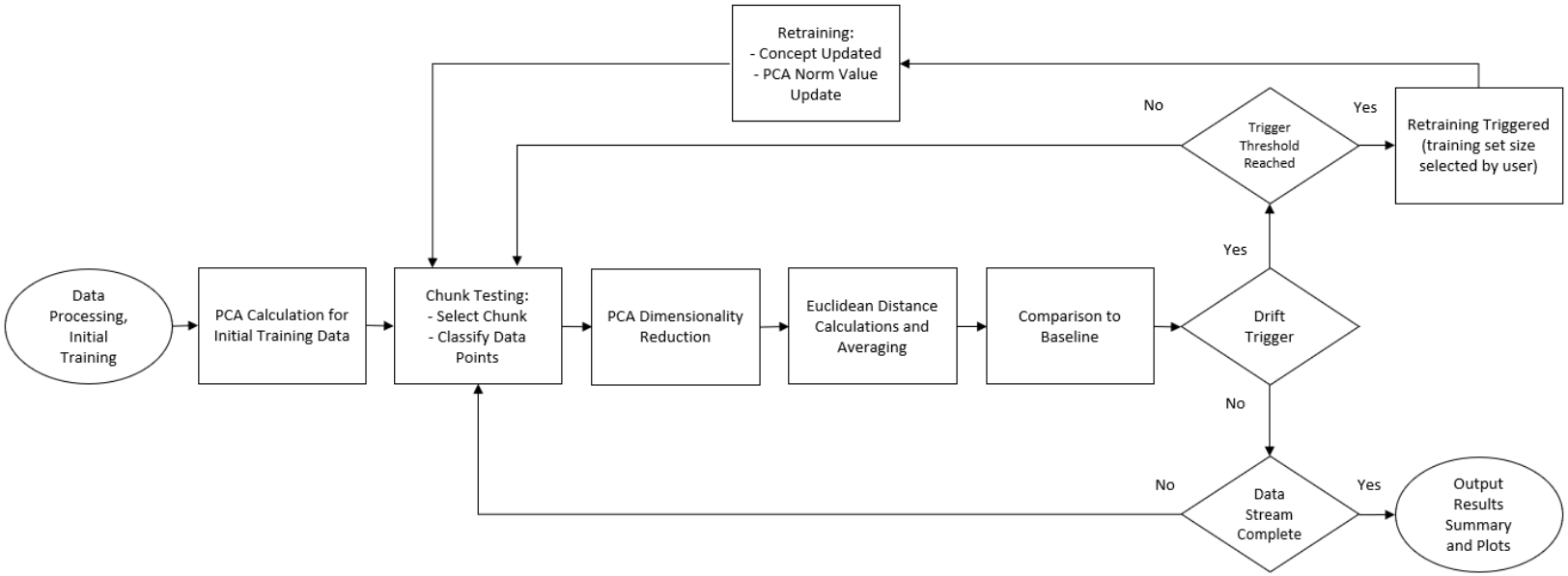

Figure 3. RD3 Flowchart

The RD3 Method is applied to a single class at a time in our research. In principle, multiple classes could run on separate, dedicated RD3 threads in parallel. The "probe" attack class will be used as an example. The classification algorithm predicts data point class types in a test chunk. The data points classified as probe attacks from the test chunk are formed into a separate class matrix. The class matrix contains the associated feature data for each selected data point.

A PCA dimensionality reduction transformation is performed on the class matrix, which produces a principal component matrix. The Euclidean distance of each point in the principal component matrix is calculated, followed by the mean of all Euclidean points. The Euclidean distance mean value for the test chunk is recorded in a vector. This process occurs for all test chunks, populating the mean value vector with one calculated value from each test chunk.

The mean value vector is used in the detection algorithm to assess for concept drift. Each mean value score is compared to the initial Euclidean distance value and converted into a percentage difference from the norm value. The percentage difference scores are sequentially assessed. If a difference score exceeds a trigger threshold, the associated trigger action occurs. For example, the massive drift trigger will immediately cue retraining, while the other triggers will iterate counts until the retraining count for the trigger is reached, which will cue retraining. If a drift trigger is reached, but not a trigger retraining count, the testing process will continue.

Once retraining is cued, the detection algorithm passes the trigger chunk data to the retraining algorithm. The retraining algorithm uses the trigger chunk information to select the retraining set, and the classifier retrains with that set. Concurrently, the algorithm takes the remaining test data, after the trigger chunk, and restructures the test data based on the user designated retraining window size and number of test data points remaining.

Testing resumes with the remaining test data, using an updated model and test window size. If concept drift trigger thresholds are reached again, the retraining algorithm will repeat the pattern. This process continues until the end of the test set is reached, and summary results and plots are output from the algorithm.

B. BASELINE METHOD: SUPERVISED TRAINING AND DETECTION

Figure 4 contains a flowchart for the Baseline Method detection and mitigation process which can be referenced throughout this section. The structure is similar to the RD3 Method, but the drift detection mechanisms are fundamentally different.

Once the classification algorithm has been trained, an initial model is developed. The model is used to evaluate the test set and predict the data point types in each test chunk.

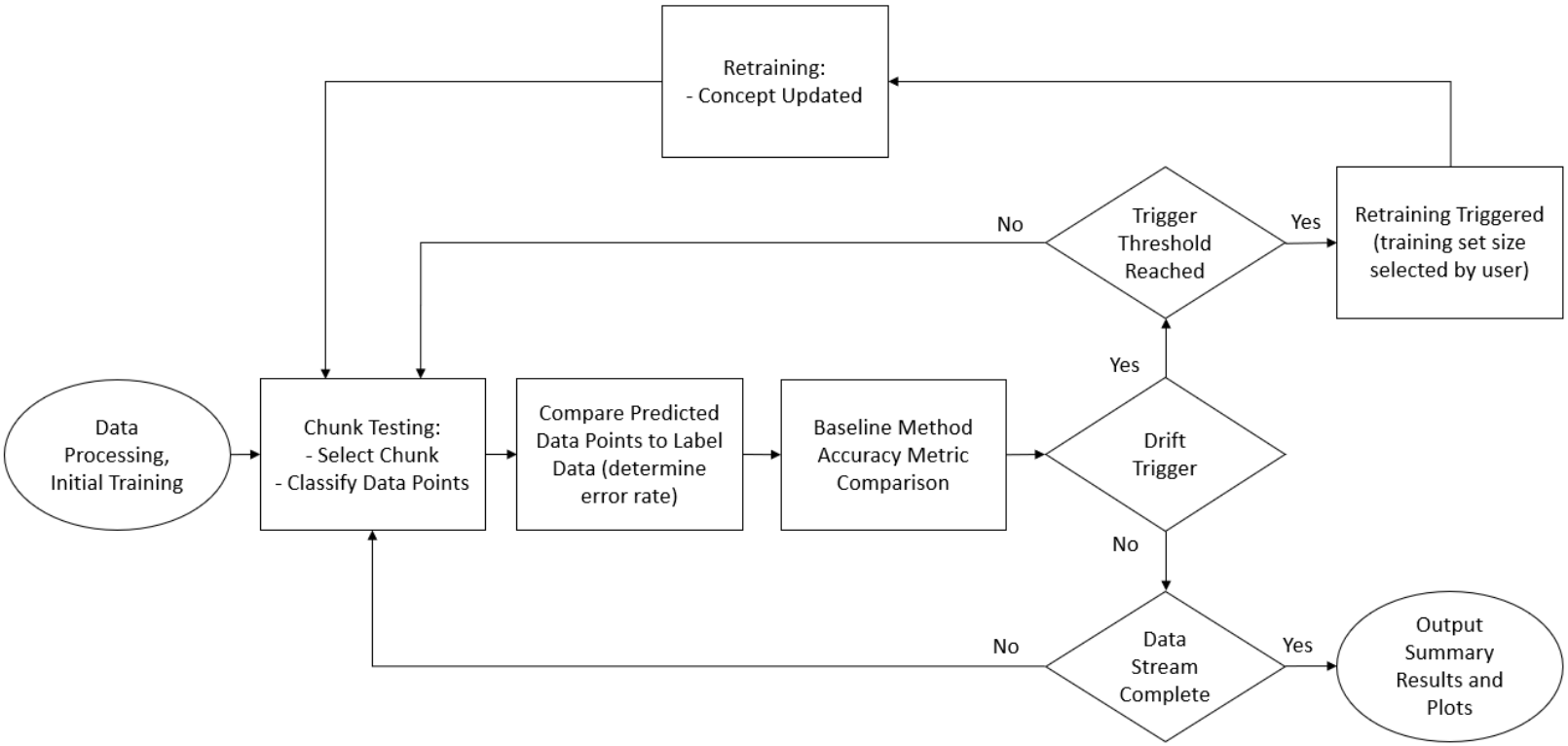

Figure 4. Baseline Method Flowchart

The classifier predicted points are compared to the labeled data, determining if each data point is correctly classified. The mean accuracy scores, per test chunk, are calculated and recorded in a score vector.

The score vector contains the mean accuracy score of each test chunk. The Baseline Method drift detection algorithm is applied to the score vector, sequentially assessing each score. If an accuracy score exceeds a trigger threshold, the associated trigger action occurs. For example, the massive drift trigger will immediately cue retraining, while the other triggers will iterate counts until the retraining count for the trigger is reached, which will cue retraining. If a drift trigger is reached, but not a trigger retraining count, the testing process will continue.

Once retraining is cued, the detection algorithm passes the trigger chunk data to the retraining algorithm. The retraining algorithm uses the trigger chunk information to select the retraining set, and the classifier retrains with that set. The algorithm takes the remaining test data, after the trigger chunk, and restructures the test data based on the user designated retraining window size and number of test data points remaining.

Testing resumes with the remaining test data, using an updated model and test window size. If concept drift trigger thresholds are reached again, the retraining algorithm will repeat the pattern. This will continue until the end of the test set is reached, and summary results and plots are output from the algorithm.

V. RESULTS

This chapter contains the results gathered from comparing our RD3 drift detection and mitigation algorithm to the Baseline Method with analysis based on selected metrics. We will first explain the selected metrics, followed by outlining trials and data gathered, and then move to data evaluation and comparison. We approach this section from the perspective of analyzing the collected trial data for each detection method and making comparisons after the data has been presented. The evaluation will contain a mix of data presentation and visual analysis.

Through the results of our research, we show that the unsupervised RD3 Method performs as well as the supervised Baseline Method in nearly every aspect. The implication is that RD3 could enable truly automated drift detection. Therefore, the final portion of this chapter contains summary comparative analyses of the parameters and methods, respectively. The cumulative results contained in this chapter support the evaluations made in each comparative analysis. The conclusions in those sections provide an explanation of the most important results.

A. SELECTED METRICS

Three metrics were selected to evaluate our results: accuracy, precision, and execution time. A qualitative explanation of the accuracy and precision metrics are provided in this section, and a quantitative description will be provided in the data section of this chapter. Execution time metrics are discussed in Appendix D.

1. Accuracy Metrics

The accuracy metric is evaluated from two perspectives. First, we ask: did the method detect concept drift correctly? Specifically, did the algorithm detect and mitigate concept drift where drift was located in a dataset, and bypass the areas where there was no drift?

The second aspect of the accuracy metric pertains to classifier accuracy. Each classifier ensemble is tuned to achieve a suitable prediction accuracy at the beginning of

dataset testing. Once concept drift occurs, detection accuracy is lowered, concept drift is detected, and retraining occurs. We ask: after retraining, do classifiers achieve suitable accuracy again? If suitable accuracy is not reached, continued drift detection may not be effective.

2. Precision Metric

The precision metric is a measure of relative distance between where concept drift begins, and mitigation is first triggered. We ask: was concept drift mitigation triggered suitably close to where concept drift started? A suitable distance is dependent on several factors, as will be discussed in the analysis section.

B. TRIALS

Twenty-seven trials were developed to assess the performance of the two detection and mitigation methods. The trial group was designed to represent every dataset and detection method pairing, each with a diverse set of parameters. The trials and associated parameters are listed in Table 6, which will be used as reference for the rest of this section.

Each dataset was paired with a drift detection and mitigation method. There are three datasets and two detection methods, for a total of six different pairs. In Table 6, the first part of the trial identifier, a number, is the method used, while the letter represents sequential values. The Baseline Method is method 1, and RD3 is method 2. For example, Trial 2-D is the fourth trial using the RD3 algorithm.

To develop diverse but related parameters for each trial, a set of standard parameters were established. The standard parameters were based on experience gained from thousands of undocumented trial runs conducted during code development, debugging, and test runs. The standard parameters were set at 10,000 training points, 1,000-point initial chunk size, 500-point retrain chunk size, ten warning triggers, and five detect triggers. The massive drift detect trigger is enabled for the standard settings. Most parameter values are adjustments from the standard, and the shifts are denoted in yellow highlight in the trial tables.

One section diverged significantly from the standard parameters. Pairing the abrupt drift dataset with the RD3 Method did not work with the standard trigger settings, as the detector was too sensitive to the abrupt drift dataset variations. A modification was made, where the warning and detect triggers were disabled, and the massive drift setting was moved from 50% to 200%. While the detection modification diverges significantly from the standard parameters, the data gathered from the resulting trials is still quite useful.

The trials are designed to exercise one portion of the parameters at a time. For example, only the training set size will change, while chunk sizes and triggers remain at the standard values. The next set of iterations will change the chunk sizes only, while returning the training set size to the standard, and so forth. Also, the Baseline Method uses well-established detection measures, while the RD3 algorithm is novel, so the trials focus on RD3.

Table 6. Trial List

Trial Number	Trial Identifier	Detection and Mitigation Method	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect
standard parameters				**10000**	**1000**	**500**	**10**	**5**	**on**
1	1-A	1	KDD	**5000**	1000	500	10	5	on
2	1-B	1	KDD	10000	**500**	**250**	10	5	on
3	1-C	1	KDD	10000	1000	500	10	5	on
4	1-D	1	Inc Drift (fast)	**5000**	1000	500	10	5	on
5	1-E	1	Inc Drift (fast)	10000	**500**	**250**	10	5	on
6	1-F	1	Inc Drift (fast)	10000	1000	500	10	5	on
7	1-G	1	Inc Drift (slow)	**5000**	1000	500	10	5	on
8	1-H	1	Inc Drift (slow)	10000	**500**	**250**	10	5	on
9	1-I	1	Inc Drift (slow)	10000	1000	500	10	5	on
10	1-J	1	KDD	10000	**2000**	**1000**	10	5	on
11	1-K	1	KDD	10000	**3000**	**1500**	10	5	on
12	2-A	2	KDD	10000	**500**	**250**	**-**	**-**	**on (mod)**
13	2-B	2	KDD	10000	1000	500	**-**	**-**	**on (mod)**
14	2-C	2	KDD	10000	**2000**	**1000**	**-**	**-**	**on (mod)**
15	2-D	2	KDD	10000	**3000**	**1500**	**-**	**-**	**on (mod)**
16	2-E	2	Inc Drift (fast)	**1000**	1000	500	10	5	on
17	2-F	2	Inc Drift (fast)	**5000**	1000	500	10	5	on
18	2-G	2	Inc Drift (fast)	10000	**500**	**250**	10	5	on
19	2-H	2	Inc Drift (fast)	10000	1000	500	10	5	on
20	2-I	2	Inc Drift (fast)	10000	**2000**	**1000**	10	5	on
21	2-J	2	Inc Drift (fast)	10000	1000	500	**5**	**1**	**off**
22	2-K	2	Inc Drift (slow)	**1000**	1000	500	10	5	on
23	2-L	2	Inc Drift (slow)	**5000**	1000	500	10	5	on
24	2-M	2	Inc Drift (slow)	10000	**500**	**250**	10	5	on
25	2-N	2	Inc Drift (slow)	10000	1000	500	10	5	on
26	2-O	2	Inc Drift (slow)	10000	**2000**	**1000**	10	5	on
27	2-P	2	Inc Drift (slow)	10000	1000	500	**5**	**1**	**off**

C. DATA GATHERED TO SUPPORT METRICS ANALYSIS

Several data measures were documented from the trial runs. The information requires qualitative explanation, and quantitative formulas are provided where appropriate. This section clarifies each piece of information and how the information was obtained.

1. Return to Accuracy

As a concept drifts in the dataset, the machine learning model prediction accuracy decreases. Once a drift mitigation threshold is reached, retraining occurs. The updated model must regain a prediction accuracy level where further drift detection is possible. We will refer to this as *return to accuracy*. Empirically, prediction accuracy should be at 95% or greater for accurate concept drift detection. We examine the accuracy scores of each detector, following retraining cycles, to determine if there is a suitable return to accuracy. This analysis is used to evaluate the accuracy metric.

2. Test Set Drift Location

Three unique datasets were used to test the detection and mitigation algorithms. Concept drift occurs in different locations within each respective set. Every set was measured to determine where concept drift starts, with respect to the beginning of the set. For example, the fast incremental drift dataset first encounters concept drift 60% of the way through the set. That means the first 60% of the set is one consistent concept, while the remaining 40% of the data is incrementally drifting away from that concept. In this case, a properly working concept drift detection algorithm will identify the drift at some point past the 60% mark.

The relative location of the drift in each dataset changes based on the number of training set data points used to initially train the model. This is because the training set is taken from the beginning of the dataset and the remaining data points becomes the test set. Therefore, the exact location of the drift is adjusted for each dataset and training set size combination. Table 7 contains the drift locations for each respective dataset and training set size combination. The drift start location for each instance is defined as $d_{start\_dataset}$.

Table 7. Datasets and Related Drift Starting Locations

Dataset	Training Set Size	Drift Start Location
KDD	1000	76%
	5000	79%
	10000	82%
Inc Drift (fast)	1000	54%
	5000	57%
	10000	60%
Inc Drift (slow)	1000	20%
	5000	23%
	10000	26%

3. Initial Retrain Cycle

The location where the first retraining cycle is triggered in the test set is important. It provides us insight to drift detection responsiveness. Therefore, we focus on analyzing the associated relationship between where drift starts in the test set and where the first mitigation cycle is triggered.

The first mitigation cycle is recorded, based on the chunk number in the test set. Test chunks sizes range from 500 to 3000 data points. As the concept drift may have occurred anywhere in the chunk, the measurement is recorded from the chunk center point in order to provide measurement consistency.

The distance is the measure from the start of concept drift to the location where the first retraining is triggered. The formula is given by

$$d_{detect\_init} - d_{start\_dataset} = detect\_precision, \tag{16}$$

where d_{detect_init} is the location of the first drift mitigation cycle for a trial. Smaller $detect\_precision$ values are more precise than larger values.

Both the dataset drift start location and the first drift detection location are in percentages. The difference between the two values is calculated as a percentage value. The relative distance values are used to evaluate the drift detection precision metric.

4. False Alarms

False alarms are retraining triggers that occur in test set locations where there is no concept change. For example, retraining should be triggered once in an abrupt concept drift dataset, following the concept change. Retraining before concept change is a false alarm, as is retraining following the concept stabilization retrain cycle. A false alarm indicates the detection algorithm is not functioning properly and is triggering on noise. The false alarm measurements were determined by examining the location of each retraining cycle, and verifying whether drift was occurring in that location, or not. False alarms are used to evaluate the accuracy metric.

D. METRIC ANALYSIS

This section contains the results of the data gathered from the test trials, with analysis of the data as it applies to the three selected metrics. Table 8 contains the tabulated results of each trial. 27 trials were conducted in support of the analysis, but we focus on 6 representative trials to properly support metric analysis in each respective section. Pertinent information from Table 8 will be extracted to provide focused results for each relevant metric. Select figures from trial runs will also be included.

Table 8. Trial List with Results

Trial Number	Trial Identifier	Detection and Mitigation Method	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Drift Location (from beginning of test stream)	Drift Detected (from beginning of test stream)	Pct Detect From Actual Drift	Total Execution Time (secs)	Initial Training Cycle Execution Time (secs)	Initial Test Cycle Execution Time (secs)	Number of Retrains	Number of False Alarms
standard parameters				10000	1000	500	10	5	on	-	-	-	-	-	-	-	-
1	1-A	1	KDD	5000	1000	500	10	5	on	79%	38%	-41%	457	52	51	6	5
2	1-B	1	KDD	10000	500	250	10	5	on	82%	12%	-70%	1954	92	72	10	9
3	1-C	1	KDD	10000	1000	500	10	5	on	82%	29%	-53%	1255	90	58	7	6
4	1-D	1	Inc Drift (fast)	5000	1000	500	10	5	on	57%	65%	8%	235	9	77	4	0
5	1-E	1	Inc Drift (fast)	10000	500	250	10	5	on	60%	63%	3%	583	17	141	4	0
6	1-F	1	Inc Drift (fast)	10000	1000	500	10	5	on	60%	65%	5%	527	18	140	3	0
7	1-G	1	Inc Drift (slow)	5000	1000	500	10	5	on	23%	31%	8%	457	9	92	6	0
8	1-H	1	Inc Drift (slow)	10000	500	250	10	5	on	26%	29%	3%	986	15	134	6	0
9	1-I	1	Inc Drift (slow)	10000	1000	500	10	5	on	26%	29%	3%	738	15	132	5	0
10	1-J	1	KDD	10000	2000	1000	10	5	on	82%	69%	-13%	678	88	53	2	1
11	1-K	1	KDD	10000	3000	1500	10	5	on	82%	83%	1%	394	90	53	1	0
12	2-A	2	KDD	10000	500	250	-	-	on (mod)	82%	38%	-44%	813	90	87	4	4
13	2-B	2	KDD	10000	1000	500	-	-	on (mod)	82%	90%	8%	307	89	73	1	0
14	2-C	2	KDD	10000	2000	1000	-	-	on (mod)	82%	92%	10%	280	90	65	1	0
15	2-D	2	KDD	10000	3000	1500	-	-	on (mod)	82%	96%	14%	255	92	62	1	0
16	2-E	2	Inc Drift (fast)	1000	1000	500	10	5	on	54%	69%	15%	116	1	47	5	0
17	2-F	2	Inc Drift (fast)	5000	1000	500	10	5	on	57%	69%	12%	249	8	84	4	0
18	2-G	2	Inc Drift (fast)	10000	500	250	10	5	on	60%	49%	-11%	995	16	148	7	4
19	2-H	2	Inc Drift (fast)	10000	1000	500	10	5	on	60%	67%	7%	609	16	136	4	0
20	2-I	2	Inc Drift (fast)	10000	2000	1000	10	5	on	60%	69%	9%	366	17	130	2	0
21	2-J	2	Inc Drift (fast)	10000	1000	500	5	1	off	60%	65%	5%	660	16	136	5	0
22	2-K	2	Inc Drift (slow)	1000	1000	500	10	5	on	20%	38%	18%	200	1	47	6	0
23	2-L	2	Inc Drift (slow)	5000	1000	500	10	5	on	23%	35%	12%	448	9	99	5	0
24	2-M	2	Inc Drift (slow)	10000	500	250	10	5	on	26%	30%	4%	1423	15	160	8	0
25	2-N	2	Inc Drift (slow)	10000	1000	500	10	5	on	26%	33%	7%	875	15	147	5	0
26	2-O	2	Inc Drift (slow)	10000	2000	1000	10	5	on	26%	36%	10%	547	16	141	3	0
27	2-P	2	Inc Drift (slow)	10000	1000	500	5	1	off	26%	30%	4%	1162	16	158	6	0

1. Accuracy Metrics Analysis

As mentioned previously in this chapter, we will look at accuracy in two aspects. We first assess if drift is detected and retraining occurs in the appropriate locations in the test sets. Second, we will evaluate classification return to accuracy, to determine if suitable prediction accuracy is regained following retraining.

a. Detection and Mitigation Accuracy

To determine if concept drift is identified in the correct areas, we look at the test data locations where drift detection retraining cycles occur. As mentioned earlier, detections in the test sets where the concept is not changing are false alarms. The false alarm trials, contained in Table 9, occur in six of the twenty-seven runs. False alarm detections may occur before or after the concept has changed, and in the six trials containing false alarms, each occurred with at least one false alarm prior to concept drift.

Five of the six false alarm trials occurred with abrupt, or sudden, change in the dataset. This indicates, in this case, that the detectors are more susceptible to noise during the quiescent phase of the abrupt concept drift in the KDD dataset. The increased test window size in trials 1-K and 1-J was designed to better tune the detectors to the abrupt drift dataset, as was the modifications to the trigger thresholds in Trials 2-A, 2-B, 2-C and 2-D.

False alarms roughly correlate with using smaller testing chunks. We see false alarms in 3 of the 6 cases using a test window of 500 data points, 2 of the 14 cases using a test window of 1000, and 1 of the 4 cases using 2000. Smaller test chunks represent smaller testing windows, which carry increased error scores relative to larger chunks or test windows. This indicates the need for averaging out noise by using adequately large test chunk sizes.

Table 10 contains the remaining 21 trials with no false alarms. The wide range of parameter settings highlights the robustness of each detector method. Specifically, each detector is relatively accurate at identifying concept drift where drift occurs. Precision factors are important as well, which will be discussed in a later section.

Table 9. Trials Containing False Alarms

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Drift Location (from beginning of test stream)	Drift Detected (from beginning of test stream)	Pct Detect From Actual Drift	Number of Retrains	Number of False Alarms
1-A	KDD	5000	1000	500	10	5	on	79%	38%	-41%	6	5
1-B	KDD	10000	500	250	10	5	on	82%	12%	-70%	10	9
1-C	KDD	10000	1000	500	10	5	on	82%	29%	-53%	7	6
1-J	KDD	10000	2000	1000	10	5	on	82%	69%	-13%	2	1
2-A	KDD	10000	500	250	-	-	on (mod)	82%	38%	-44%	4	4
2-G	Inc Drift (fast)	10000	500	250	10	5	on	60%	49%	-11%	7	4

Table 10. Trials with Accurate Detection

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Drift Location (from beginning of test stream)	Drift Detected (from beginning of test stream)	Pct Detect From Actual Drift	Number of Retrains	Number of False Alarms
1-D	Inc Drift (fast)	5000	1000	500	10	5	on	57%	65%	8%	4	0
1-E	Inc Drift (fast)	10000	500	250	10	5	on	60%	63%	3%	4	0
1-F	Inc Drift (fast)	10000	1000	500	10	5	on	60%	65%	5%	3	0
1-G	Inc Drift (slow)	5000	1000	500	10	5	on	23%	31%	8%	6	0
1-H	Inc Drift (slow)	10000	500	250	10	5	on	26%	29%	3%	6	0
1-I	Inc Drift (slow)	10000	1000	500	10	5	on	26%	29%	3%	5	0
1-K	KDD	10000	3000	1500	10	5	on	82%	83%	1%	1	0
2-B	KDD	10000	1000	500	-	-	on (mod)	82%	90%	8%	1	0
2-C	KDD	10000	2000	1000	-	-	on (mod)	82%	92%	10%	1	0
2-D	KDD	10000	3000	1500	-	-	on (mod)	82%	96%	14%	1	0
2-E	Inc Drift (fast)	1000	1000	500	10	5	on	54%	69%	15%	5	0
2-F	Inc Drift (fast)	5000	1000	500	10	5	on	57%	69%	12%	4	0
2-H	Inc Drift (fast)	10000	1000	500	10	5	on	60%	67%	7%	4	0
2-I	Inc Drift (fast)	10000	2000	1000	10	5	on	60%	69%	9%	2	0
2-J	Inc Drift (fast)	10000	1000	500	5	1	off	60%	65%	5%	5	0
2-K	Inc Drift (slow)	1000	1000	500	10	5	on	20%	38%	18%	6	0
2-L	Inc Drift (slow)	5000	1000	500	10	5	on	23%	35%	12%	5	0
2-M	Inc Drift (slow)	10000	500	250	10	5	on	26%	30%	4%	8	0
2-N	Inc Drift (slow)	10000	1000	500	10	5	on	26%	33%	7%	5	0
2-O	Inc Drift (slow)	10000	2000	1000	10	5	on	26%	36%	10%	3	0
2-P	Inc Drift (slow)	10000	1000	500	5	1	off	26%	30%	4%	6	0

b. Classifier Return to Accuracy Following Retraining

The second accuracy measure is the accuracy performance of the classifier ensemble following retraining cycles. Specifically, we assess if an updated model regains suitable accuracy, previously lost due to concept drift once retraining occurs. The Baseline Method is a supervised learning accuracy detection method, and the return to accuracy can be assessed directly from the detection method results.

The RD3 Method does not use labels for supervised detection of drift, but the labels are still available for each dataset. Because of this, the prediction accuracy of the RD3 Method can be calculated. This provides an accuracy measure to assess algorithm performance.

Table 11 contains the trials used to assess this metric. Tables 12 - 17 contain the parameters and detailed results, while Figures 6 - 22 are visual representations of the results.

The six trials used in this section represent the standard parameters and associated results, highlighting the utility of the standard parameter approach. The two exceptions are Trials 1-K and 2-B. As mentioned before, Trial 1-K required larger chunk sizes to avoid false alarms, and Trial 2-B contained the modified trigger parameter, which is necessary when the abrupt drift dataset and RD3 detector are paired.

Table 11. Trials Used to Assess Classifier Return to Accuracy

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect
1-F	Inc Drift (fast)	10000	1000	500	10	5	on
1-I	Inc Drift (slow)	10000	1000	500	10	5	on
1-K	KDD	10000	**3000**	**1500**	10	5	on
2-B	KDD	10000	1000	500	-	-	**on (mod)**
2-H	Inc Drift (fast)	10000	1000	500	10	5	on
2-N	Inc Drift (slow)	10000	1000	500	10	5	on

The individual tables for the six trials (tables 12 - 17) contain the parameters and results for the respective trials, while the associated figures visually represent the accuracy and detection information. Figures 5, 7, 9, 11, 15, and 19 depict the initial accuracy benchmarks for each trial. As stated earlier, initial high accuracy is important to detecting concept drift, and each classifier ensemble begins with at least one classifier with suitably

high accuracy. Empirical experience gained while programming each algorithm revealed that a classifier prediction accuracy of 95% or higher worked well with both detection algorithms. Each ensemble performed well within this range, with most associated classifiers exceeding 99% initial accuracy. The remainder of this section will be used to individually evaluate the accuracy of each representative trial, following retraining cycles.

Trial 1-F is represented by Table 12, and Figures 5 and 6. There are three retraining mitigation cycles in this trial. Prior to the first retraining cycle, the ensemble accuracy drops from > 99% to approximately 97%. Following the first retrain cycle, the ensemble performs well with most classifiers achieving 98% accuracy, but only one classifier maintains an average close to the original score, which was the k-Nearest Neighbor (KNN) at 99.5%. The accuracy drops slightly below 99% as incremental drift continues, and retraining triggers again. From this point, the average ensemble accuracy remains above 99%. Overall, suitable accuracy was maintained throughout the trial.

Table 12. Trial 1-F Data

Trial ID: 1-F					
Parameters		**Results**			
				Trigger Type	**Classifier**
Train/Retrain Set Sizes	10000	Trial Time (secs)	527		
Chunk Size (initial)	1000	Initial Training Cycle Time (secs)	18		
Chunk Size (post-retrain)	500	Initial Testing Cycle Time (secs)	140		
Dataset	Inc Drift (fast)	Total Retraining Cycles	3		
Dataset Size (total data points)	155,434	First Retrain Chunk	94	Detect	SVM
Drift Location (from test set start)	60%	First Retrain (from test set start)	64.4%		
Number of Chunks (initial)	146	Number of Chunks (post-retrain)	197		
Chunk Drift Begins	88				
		Remaining Retraining Locations:			
Massive Drift Threshold	On	Retrain 2 Chunk	119	Warning	KNN
Drift Detect Threshold	5	Retrain 3 Chunk	157	Warning	KNN
Drift Warning Threshold	10				

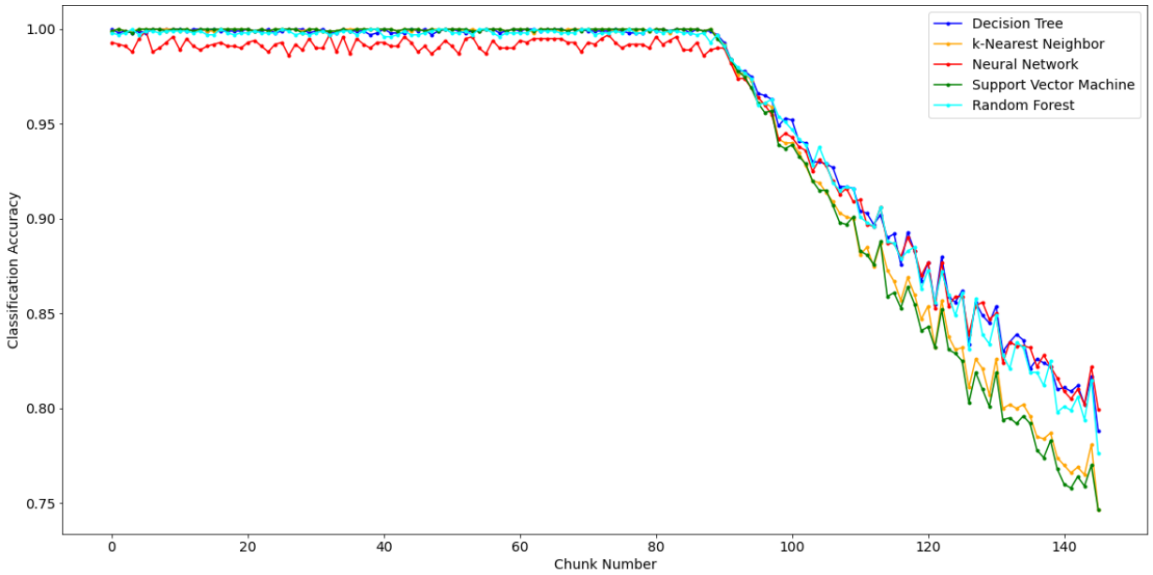

Figure 5. Trial 1-F Initial Classification Accuracy Benchmark

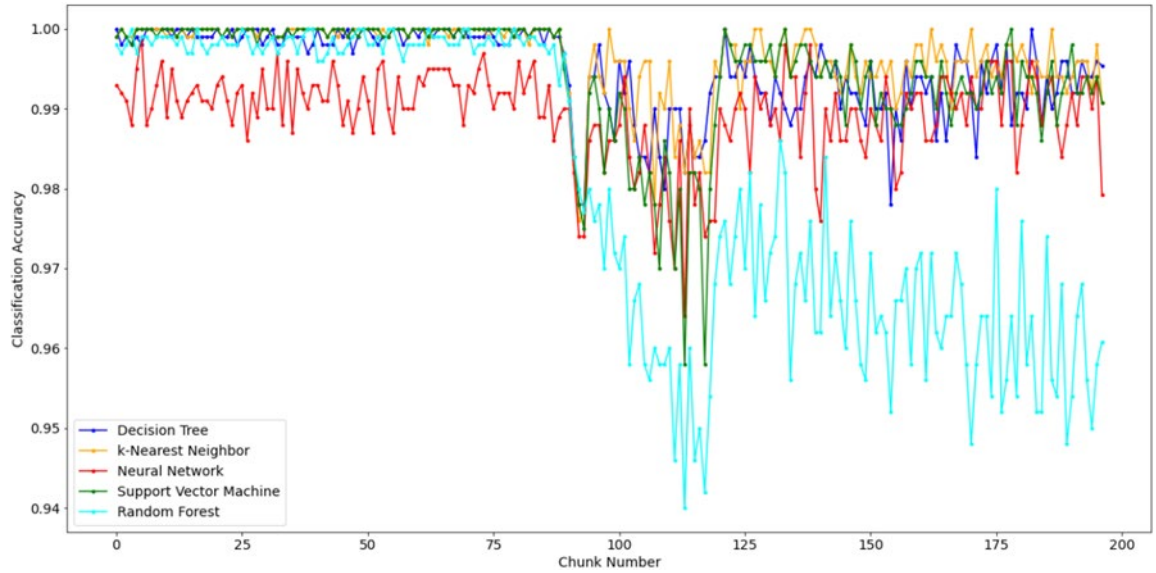

Figure 6. Trial 1-F Final Classification Accuracy Scores

Trial 1-I is represented by Table 13, and Figures 7 and 8. Five retraining cycles are triggered in this trial. The average ensemble accuracy drops to approximately 98% prior to the first retraining trigger, and the top classifier in the ensemble rebounds to approximately 99% following the retrain. Accuracy once again falls to 98% as drift continues but recovers to an average accuracy of 99.5% following the second retraining cycle and maintains this

average throughout the remaining test stream and retrain triggers. The value of the ensemble classifier approach is confirmed in this trial, as the Random Forest (RF) algorithm continually performs well below the rest of the ensemble. The ANN also performs poorly in the initial and final portions of the trial. The remaining classifiers in the ensemble ensure accurate drift detection.

Table 13. Trial 1-I Data

Trial ID: 1-I					
Parameters		**Results**			
				Trigger Type	**Classifier**
Train/Retrain Set Sizes	10000	Trial Time (secs)	738		
Chunk Size (initial)	1000	Initial Training Cycle Time (secs)	15		
Chunk Size (post-retrain)	500	Initial Testing Cycle Time (secs)	132		
Dataset	Inc Drift (slow)	Total Retraining Cycles	5		
Dataset Size (total data points)	161,279	First Retrain Chunk	43	Massive	DT
Drift Location (from test set start)	26%	First Retrain (from test set start)	28.3%		
Number of Chunks (initial)	152	Number of Chunks (post-retrain)	260		
Chunk Drift Begins	40				
		Remaining Retraining Locations:			
Massive Drift Threshold	On	Retrain 2 Chunk	82	Massive	DT
Drift Detect Threshold	5	Retrain 3 Chunk	131	Warning	KNN
Drift Warning Threshold	10	Retrain 4 Chunk	178	Warning	KNN
		Retrain 5 Chunk	219	Warning	KNN

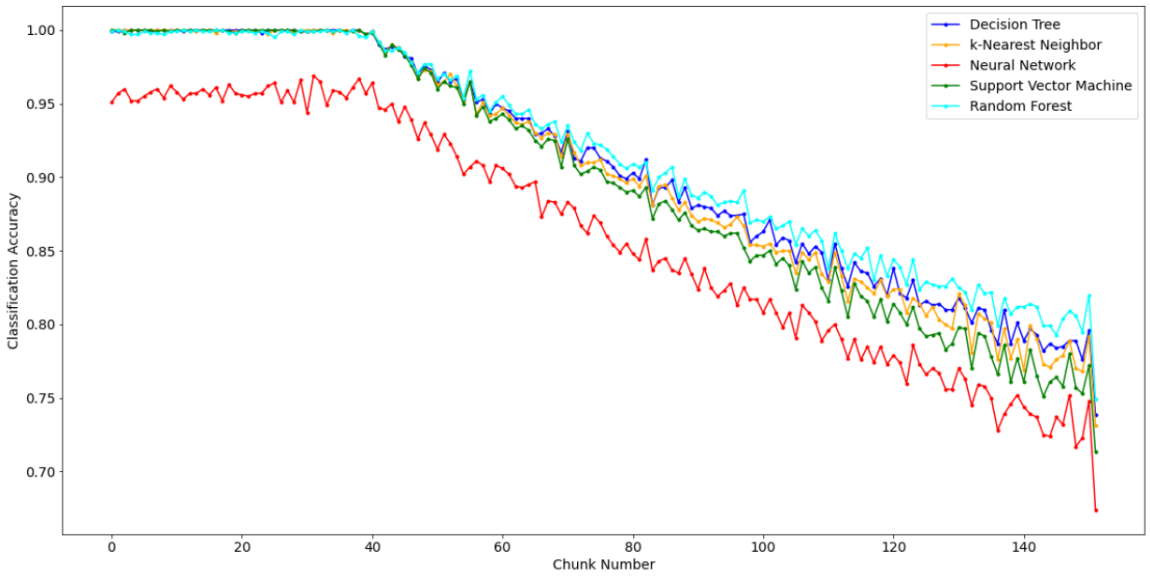

Figure 7. Trial 1-I Initial Classification Accuracy Benchmark

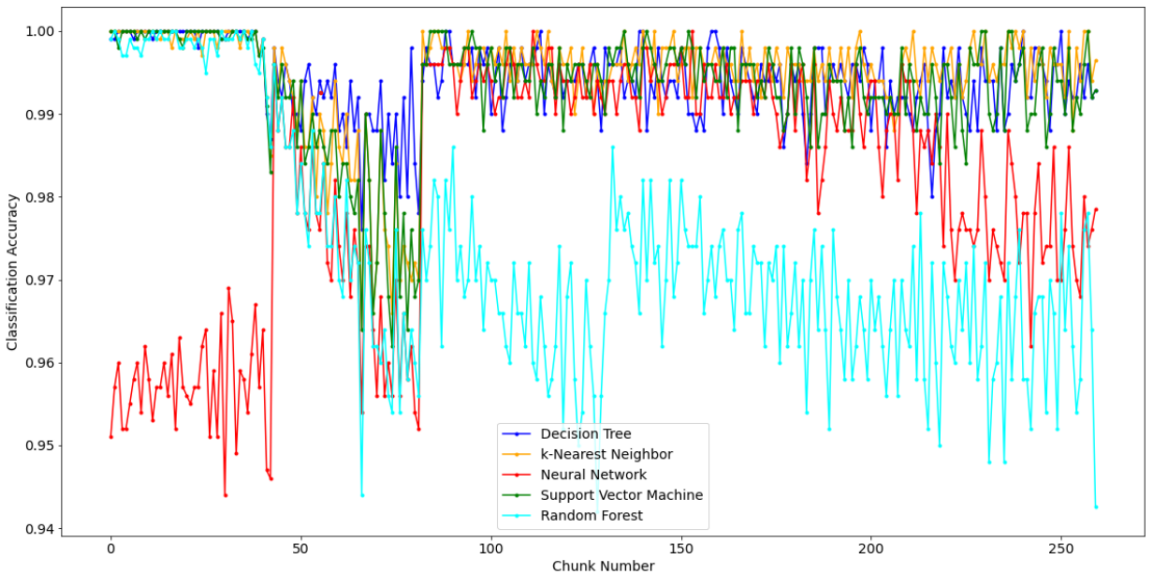

Figure 8. Trial 1-I Final Classification Accuracy Scores

Trial 1-K is represented by Table 14, and Figures 9 and 10. It contains a single retrain cycle, based on the abrupt concept drift occurring in the test dataset. The average ensemble accuracy falls to approximately 90%, and retraining is triggered. The average accuracy of the top two performing classifiers climbs back to 96%, which is suitable for continued accurate concept drift detection. This is lower than the previous two trials discussed in this section due to the selection of the data window used for retraining. In this trial, retraining is triggered quickly after concept drift occurs. This causes the training set to contain both old and new concept training data points, leading to reduced accuracy in the portion of the test set with the new concept. Overall, this is still an acceptable accuracy score for continued drift detection.

Table 14. Trial 1-K Data

Trial ID: 1-K					
Parameters		**Results**			
				Trigger Type	**Classifier**
Train/Retrain Set Sizes	10000	Trial Time (secs)	394		
Chunk Size (initial)	3000	Initial Training Cycle Time (secs)	90		
Chunk Size (post-retrain)	1500	Initial Testing Cycle Time (secs)	53		
Dataset	KDD	Total Retraining Cycles	1		
Dataset Size (total data points)	148,516	First Retrain Chunk	39	Massive	KNN
Drift Location (from test set start)	82%	First Retrain (from test set start)	83.0%		
Number of Chunks (initial)	47	Number of Chunks (post-retrain)	54		
Chunk Drift Begins	39				
		Remaining Retraining Locations:			
Massive Drift Threshold	On				
Drift Detect Threshold	5				
Drift Warning Threshold	10				

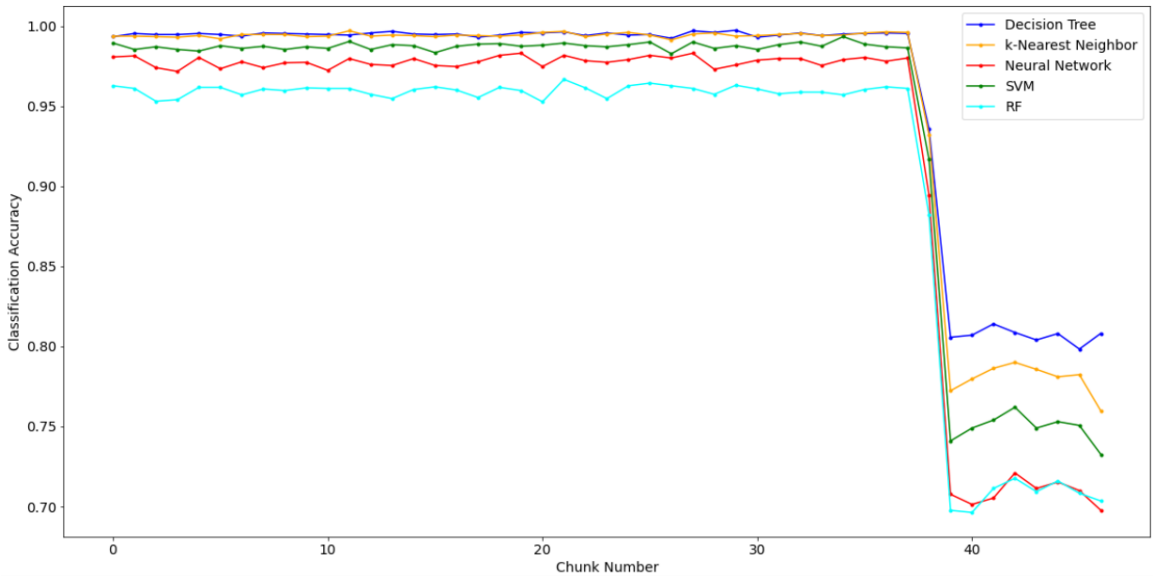

Figure 9. Trial 1-K Initial Classification Accuracy Benchmark

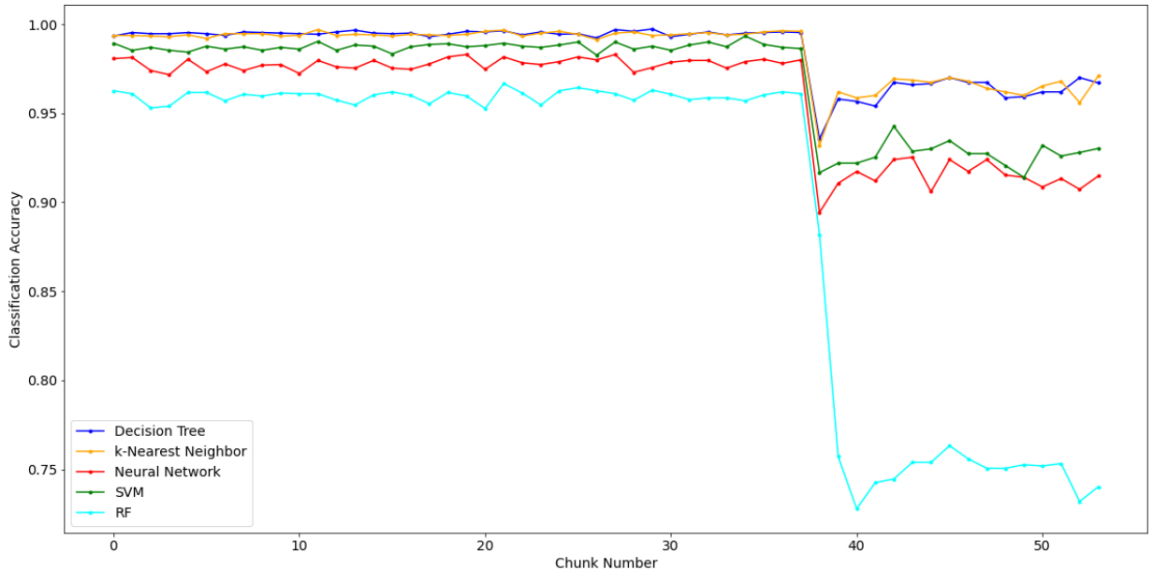

Figure 10. Trial 1-K Final Classification Accuracy Scores

Trial 2-B is represented by Table 15, and Figures 11 - 14. It contains one retraining cycle. In this case, high accuracy is maintained both before and after retraining. Even though the accuracy drops to 70%, the retraining set consists entirely of the new concept. Therefore, the accuracy of some classifiers in the ensemble is on par with the original concept accuracy of 99%. Figure 13 provides an excellent visual example of the original and new concept classification accuracies. Figure 14 depicts the large Euclidian distance spike, which triggers the sole retraining cycle.

Table 15. Trial 2-B Data

Trial ID: 2-B					
Parameters		**Results**			
				Trigger Type	**Classifier**
Train/Retrain Set Sizes	10000	Trial Time (secs)	307		
Chunk Size (initial)	1000	Initial Training Cycle Time (secs)	89		
Chunk Size (post-retrain)	500	Initial Testing Cycle Time (secs)	73		
Dataset	KDD	Total Retraining Cycles	1		
Dataset Size (total data points)	148,516	First Retrain Chunk	124	Massive	DT
Drift Location (from test set start)	82%	First Retrain (from test set start)	89.2%		
Number of Chunks (initial)	139	Number of Chunks (post-retrain)	154		
Chunk Drift Begins	114				
		Remaining Retraining Locations:			
Massive Drift Threshold	Mod				
Drift Detect Threshold	-				
Drift Warning Threshold	-				

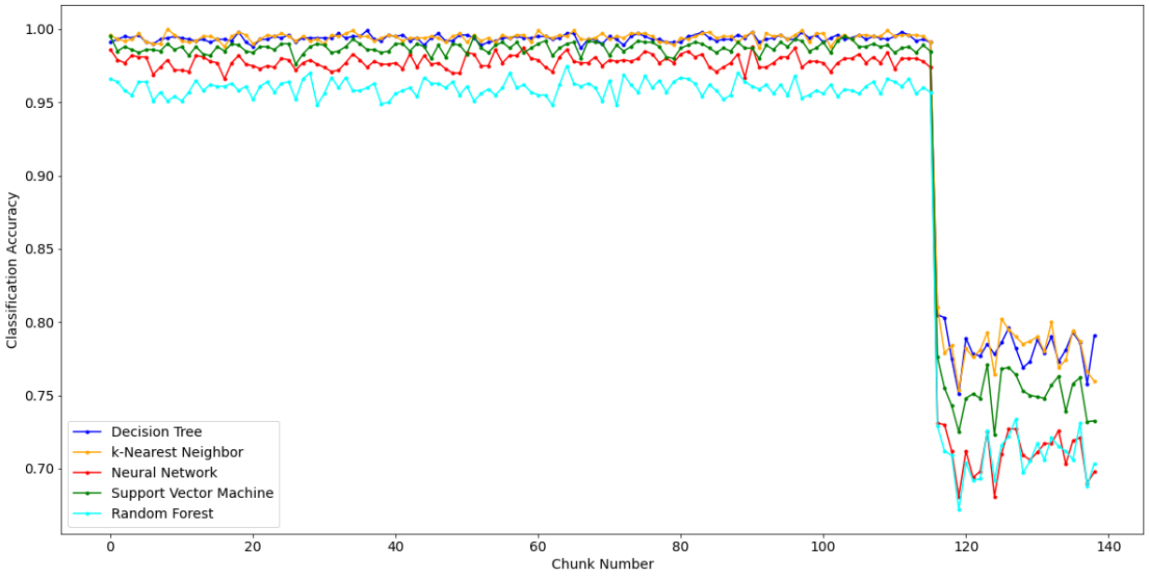

Figure 11. Trial 2-B Initial Classification Accuracy Benchmark

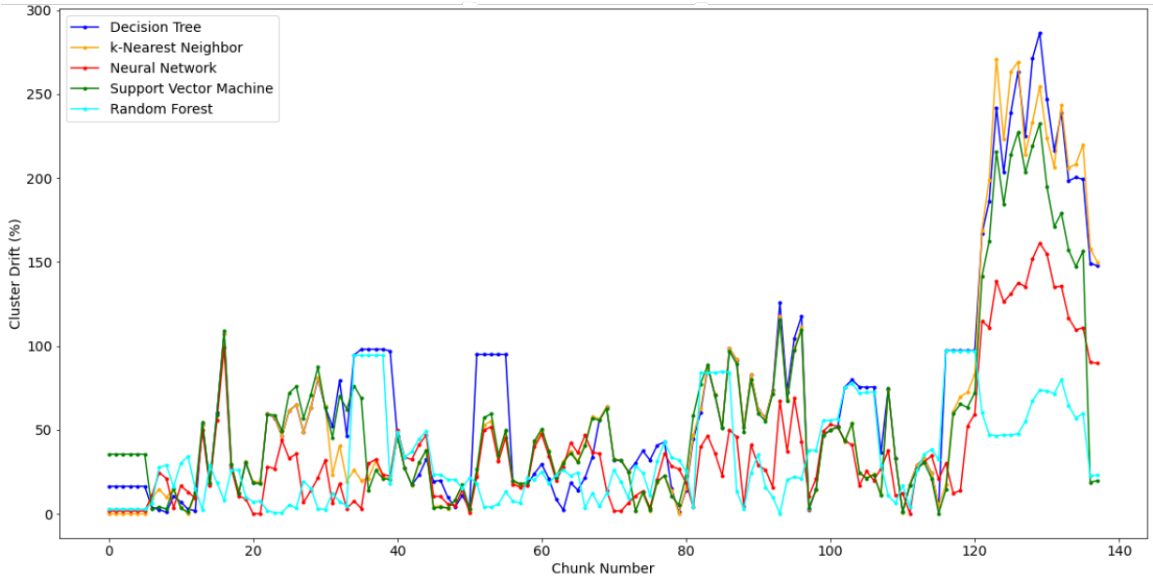

Figure 12. Trial 2-B Initial Detection Scores

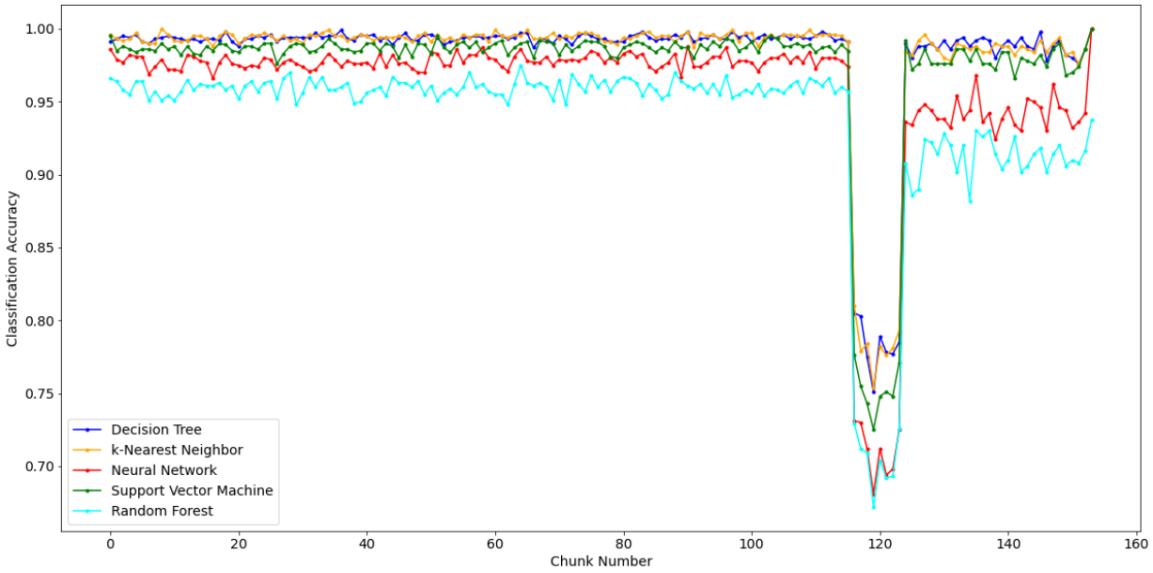

Figure 13. Trial 2-B Final Classification Accuracy Scores

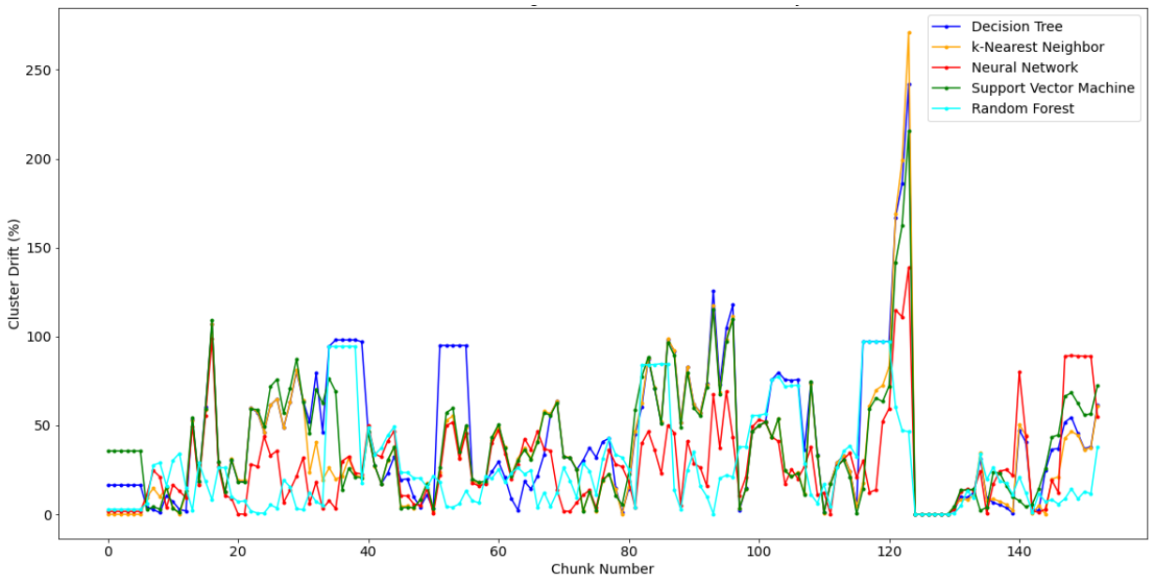

Figure 14. Trial 2-B Full Detection and Retraining Profile

Trial 2-H is represented by Table 16, and Figures 15–18. This trial contains four retraining cycles. A pattern similar to the previous trials analyzed in this section emerges. The first retraining cycle achieves suitable but slightly lower accuracy than the initial concept. However, the following retraining cycles raise the overall ensemble accuracy to a level closer to initial accuracy. This result is similar to Trial 1-K and is also due to the mix of old and new concept training set data points in the first retrain cycle.

Table 16. Trial 2-H Data

Trial ID: 2-H					
Parameters		**Results**			
				Trigger Type	**Classifier**
Train/Retrain Set Sizes	10000	Trial Time (secs)	609		
Chunk Size (initial)	1000	Initial Training Cycle Time (secs)	16		
Chunk Size (post-retrain)	500	Initial Testing Cycle Time (secs)	136		
Dataset	Inc Drift (fast)	Total Retraining Cycles	4		
Dataset Size (total data points)	155,434	First Retrain Chunk	97	Massive	KNN
Drift Location (from test set start)	60%	First Retrain (from test set start)	66.4%		
Number of Chunks (initial)	146	Number of Chunks (post-retrain)	194		
Chunk Drift Begins	88				
		Remaining Retraining Locations:			
Massive Drift Threshold	On	Retrain 2 Chunk	128	Detect	SVM
Drift Detect Threshold	5	Retrain 3 Chunk	168	Warning	RF
Drift Warning Threshold	10	Retrain 4 Chunk	191	Warning	NN

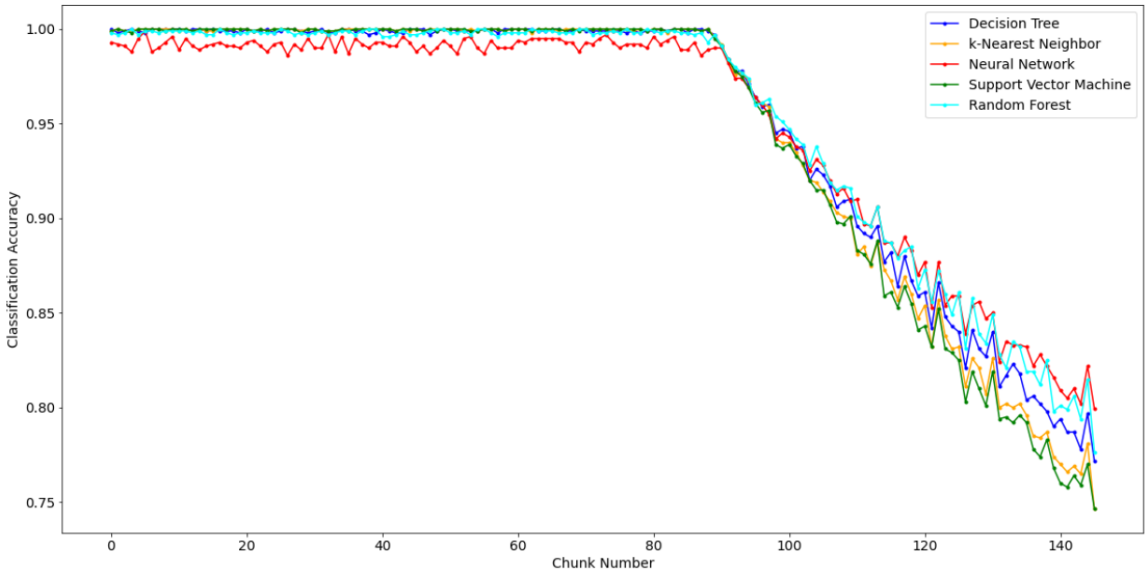

Figure 15. Trial 2-H Initial Classification Accuracy Benchmark

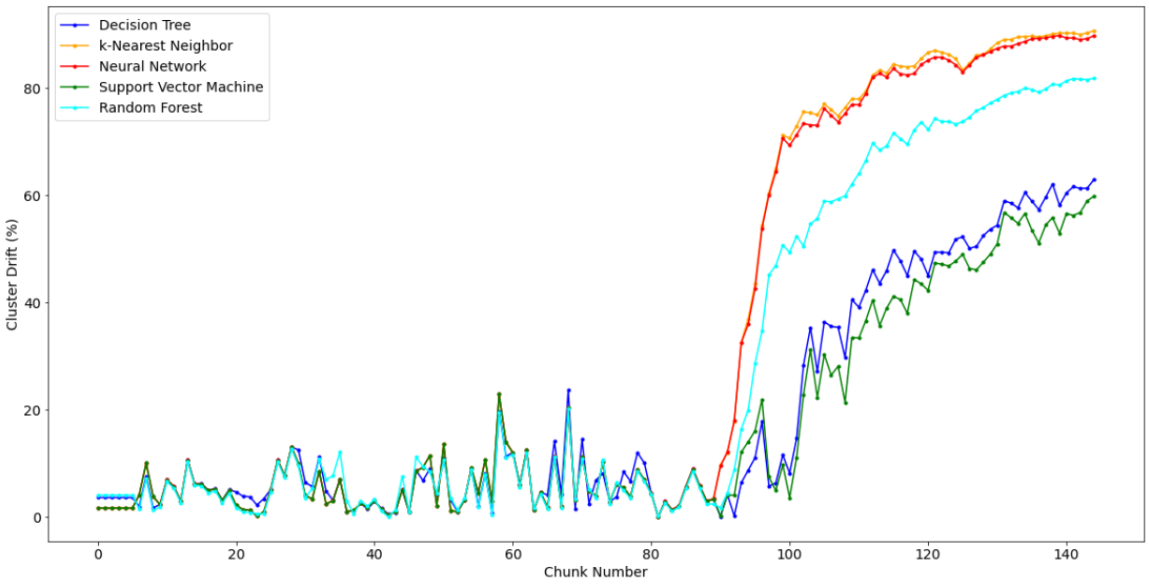

Figure 16. Trial 2-H Initial Detection Scores

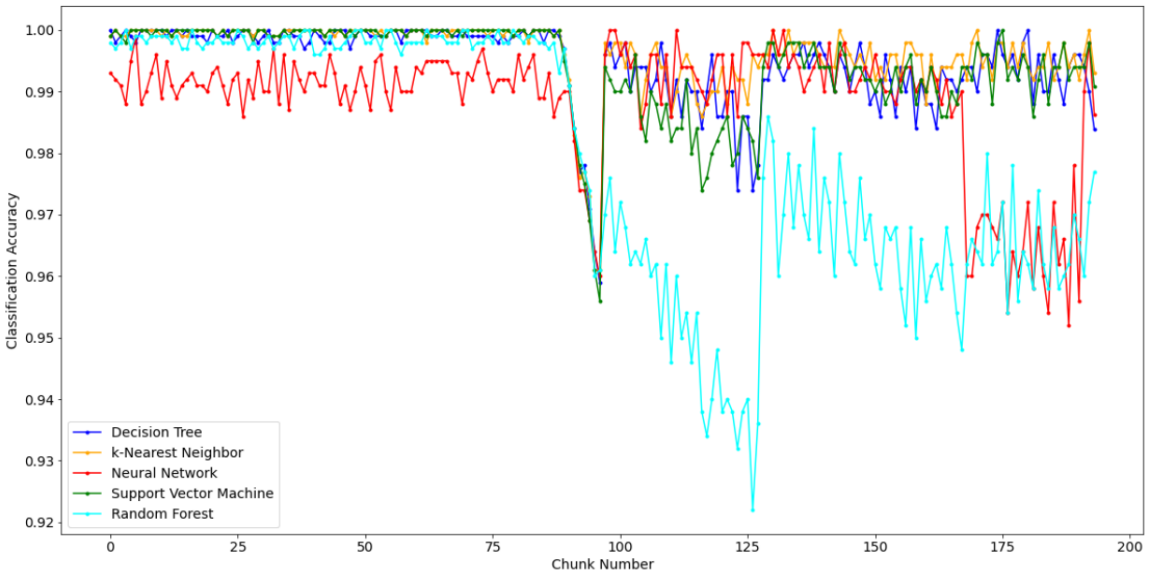

Figure 17. Trial 2-H Final Classification Accuracy Scores

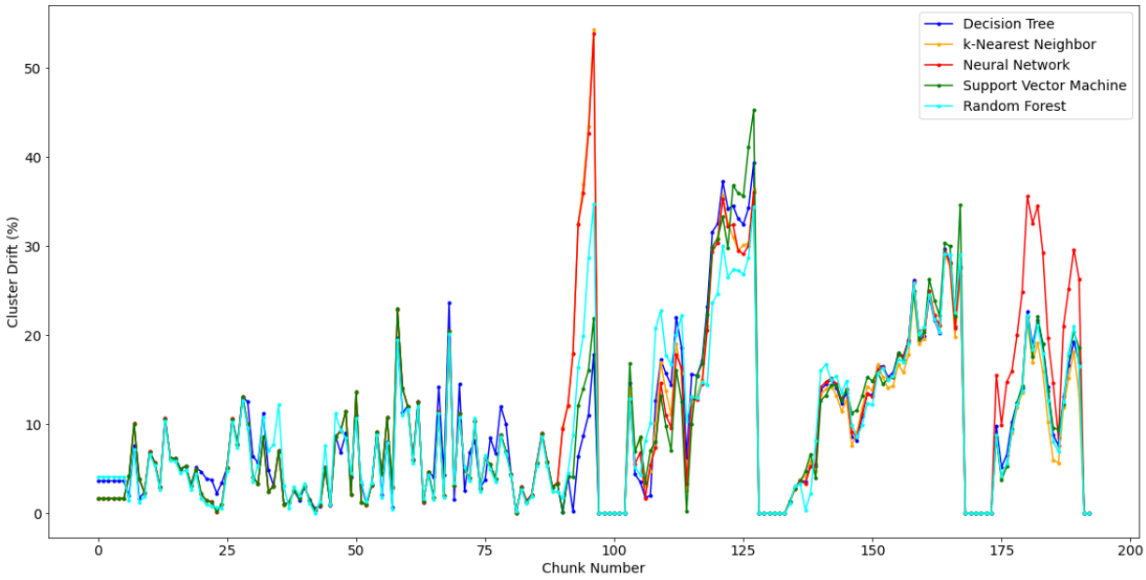

Figure 18. Trial 2-H Full Detection and Retraining Profile

Trial 2-N is represented by Table 17, and Figures 19 - 22. It contains five retraining cycles. As with Trial 2-H and the other select trials used in this section, suitable accuracy is achieved after the first retraining, with particularly high accuracy following the second and follow-on retrain cycles.

Table 17. Trial 2-N Data

Trial ID: 2-N					
Parameters		**Results**			
				Trigger Type	**Classifier**
Train/Retrain Set Sizes	10000	Trial Time (secs)	875		
Chunk Size (initial)	1000	Initial Training Cycle Time (secs)	15		
Chunk Size (post-retrain)	500	Initial Testing Cycle Time (secs)	147		
Dataset	Inc Drift (slow)	Total Retraining Cycles	5		
Dataset Size (total data points)	161,279	First Retrain Chunk	49	Detect	KNN
Drift Location (from test set start)	26%	First Retrain (from test set start)	32.2%		
Number of Chunks (initial)	152	Number of Chunks (post-retrain)	254		
Chunk Drift Begins	40				
		Remaining Retraining Locations:			
Massive Drift Threshold	On	Retrain 2 Chunk	66	Detect	SVM
Drift Detect Threshold	5	Retrain 3 Chunk	110	Detect	SVM
Drift Warning Threshold	10	Retrain 4 Chunk	161	Warning	NN
		Retrain 5 Chunk	222	Warning	KNN

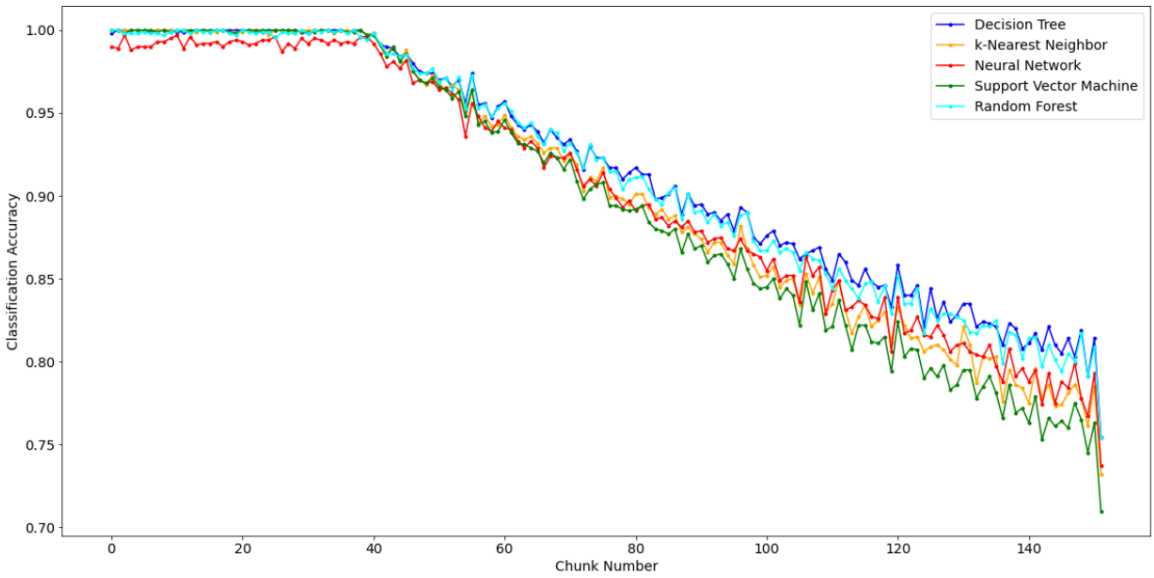

Figure 19. Trial 2-N Initial Classification Accuracy Benchmark

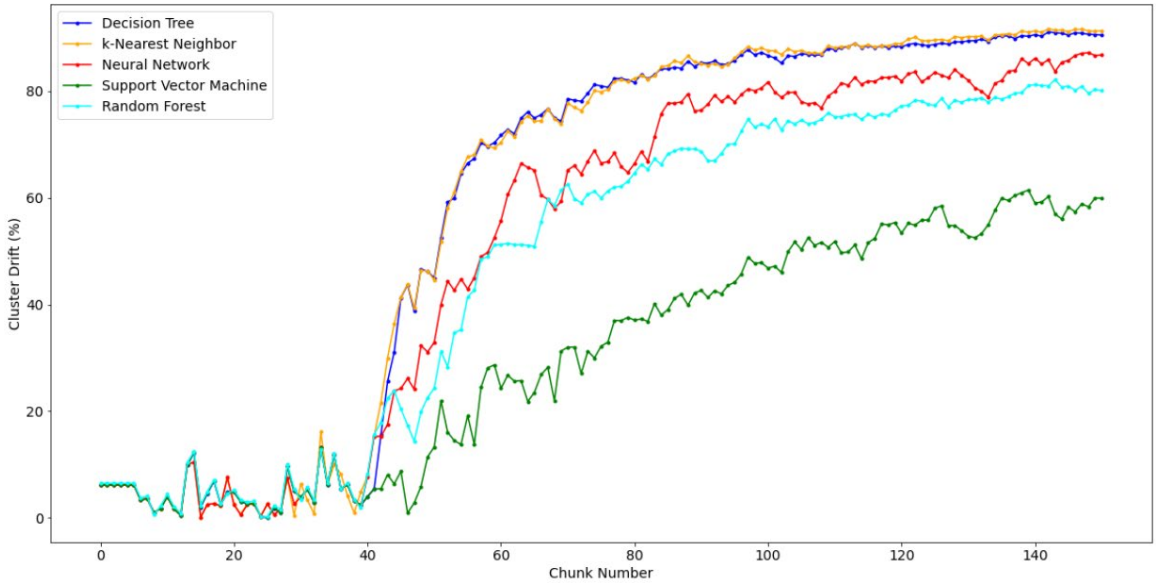

Figure 20. Trial 2-N Initial Detection Scores

Figure 21. Trial 2-N Final Classification Accuracy Scores

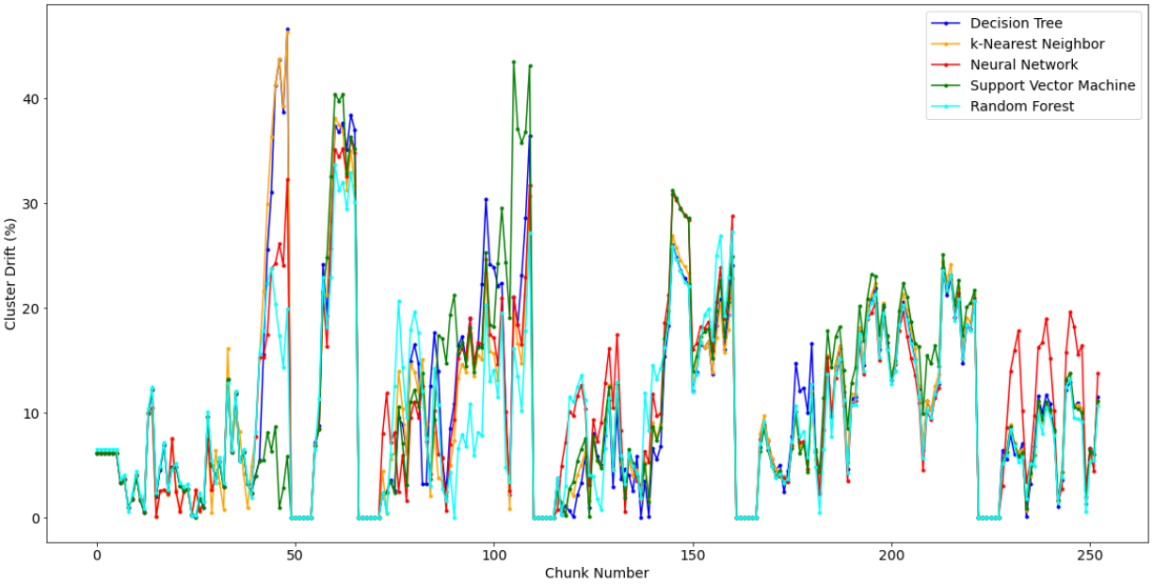

Figure 22. Trial 2-N Full Detection and Retraining Profile

Overall, these select trials demonstrate that suitable classifier prediction accuracy can be recovered relatively quickly following retraining. The unsupervised RD3 Method performs just as well as the supervised Baseline Method. Some retraining cycles show that parameters could be tuned for better results, such as Trial 1-K, but even with the reduced

accuracy, continued accurate detection is still possible within the neighborhood of the parameters selected for each of these trials.

2. Precision Metric Analysis

In this section we will evaluate how precisely a mitigation cycle is triggered, in relation to where concept drift starts in a given dataset. A detection trial is more precise when the calculated value is smaller, meaning mitigation was triggered closer to where the drift occurs. Table 18 contains the precision measurements for each trial, highlighted in green, ordered from smallest detection distance to largest. In this case, false alarms occurring prior to concept drift start are removed from consideration, as they are errors in detection.

Table 18. Detection Metric Measurements

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Drift Location (from beginning of test stream)	Drift Detected (from beginning of test stream)	Pct Detect From Actual Drift
1-K	KDD	10000	3000	1500	10	5	on	82%	83%	1%
1-I	Inc Drift (slow)	10000	1000	500	10	5	on	26%	29%	3%
1-H	Inc Drift (slow)	10000	500	250	10	5	on	26%	29%	3%
1-E	Inc Drift (fast)	10000	500	250	10	5	on	60%	63%	3%
2-P	Inc Drift (slow)	10000	1000	500	5	1	off	26%	30%	4%
2-M	Inc Drift (slow)	10000	500	250	10	5	on	26%	30%	4%
1-F	Inc Drift (fast)	10000	1000	500	10	5	on	60%	65%	5%
2-J	Inc Drift (fast)	10000	1000	500	5	1	off	60%	65%	5%
2-N	Inc Drift (slow)	10000	1000	500	10	5	on	26%	33%	7%
2-H	Inc Drift (fast)	10000	1000	500	10	5	on	60%	67%	7%
2-B	KDD	10000	1000	500	-	-	on (mod)	82%	90%	8%
1-D	Inc Drift (fast)	5000	1000	500	10	5	on	57%	65%	8%
1-G	Inc Drift (slow)	5000	1000	500	10	5	on	23%	31%	8%
2-I	Inc Drift (fast)	10000	2000	1000	10	5	on	60%	69%	9%
2-C	KDD	10000	2000	1000	-	-	on (mod)	82%	92%	10%
2-O	Inc Drift (slow)	10000	2000	1000	10	5	on	26%	36%	10%
2-F	Inc Drift (fast)	5000	1000	500	10	5	on	57%	69%	12%
2-L	Inc Drift (slow)	5000	1000	500	10	5	on	23%	35%	12%
2-D	KDD	10000	3000	1500	-	-	on (mod)	82%	96%	14%
2-E	Inc Drift (fast)	1000	1000	500	10	5	on	54%	69%	15%
2-K	Inc Drift (slow)	1000	1000	500	10	5	on	20%	38%	18%

Ordering the precision measures in this manner allows us to quickly determine a few of the key influential parameters affecting precision. The precision measurement from the trials range from the 1% to 18%, with the smaller values translating to shorter or quicker detection times, and larger distances to longer or slower detection times. After inspecting the results, two parameters are clearly related to detection precision: Training set size and test chunk size.

The 11 most precise values belong to trials consisting of 10,000-point training set sizes. The larger training set sizes indicate that models developed from larger training sets detect concept drift more precisely. To further reinforce this point, the trials with the largest detection time consist of 1,000-point size training sets, the lowest values for the training set size parameter.

The other significant parameter is the test chunk size. In this case, smaller test chunks generally result in better detection precision. This is demonstrated by three of the first six trials, which consist of 500-point size initial test chunks. Furthermore, four of the last eight trials contain some of the largest test chunks. The one exception is the first trial, with a 3,000-point test chunk size, scoring a 1% detection distance. But this trial was specially designed to test the abrupt drift dataset and Baseline Method pairing. All identical pairings with smaller test chunk sizes resulted in false alarms prior to dataset concept drift, resulting in the 3,000-point size as the only valid trial.

As mentioned in the previous section, highly precise detections have drawbacks. Trial 1-K is a good example of this. This trial has a 1% detection distance, which is very close to where dataset concept drift starts. But retraining that close to the drift resulted in relatively low rebounding prediction accuracy scores. With 10,000 training points, an ideal detection distance is closer to 7%, which is represented by Trial 2-N. The return to accuracy in Trial 2-N is near the level of initial accuracy.

In the case of precision, RD3 did not perform as well as the Baseline Method. The RD3 Method lags the Baseline Method by 3 to 4% when comparing the standard parameter trials. Parameter tuning is the best approach to bring RD3 performance closer to the baseline scores.

E. COMPARATIVE ANALYSIS: PARAMETERS

After analyzing each of the metrics individually, a brief comparative analysis of important parameters is in order. We will focus the comparison on the training set size and test chunk size parameters. Adjusting these parameters may bring benefits in one performance aspect, but drawbacks in another.

Training set sizes significantly affected the results in each metric. Larger training sets translated into more accurate and more precise detection. Conversely, execution time was significantly increased with larger training set sizes, particularly if additional retraining cycles were triggered. Furthermore, the trials selected to assess classifier return to accuracy following retraining were intentional, as they best represented the results for accuracy measurements. See Appendix D for further details on program execution times.

Test chunk size was the other significant parameter factor. In this case, smaller chunk sizes led to greater precision, but accuracy was adversely affected in the form of increased false alarms. And, although not conclusive, smaller chunk sizes generally triggered more retraining cycles, increasing execution times.

Based on our results, with the proper parameter tuning each algorithm was able to perform detection and mitigation accurately, precisely, and efficiently. However, we must be aware that parameter adjustments made to benefit one aspect of performance may adversely affect performance in another area. Prioritization of the performance metrics, determining which aspects require the best performance, should be considered before parameters are adjusted.

F. COMPARATIVE ANALYSIS: METHODS

To conclude this chapter, we compare the performance of the RD3 detection and mitigation method with the Baseline Method derived from Gama's DDM algorithm. The Gama's DDM is a proven, widely used supervised drift detector [7], [16]. As such, the Baseline Method we created based on Gama's DDM provides a good benchmark to assess the performance of the RD3 Method. Table 19 contains detection precision and return to accuracy values from the select trials presented earlier in this chapter. Based on the results of our trials with the two methods, this analysis is relatively short. The execution times are

similar, as the variation pertains to dataset type, rather than method. Also, the return to accuracy following retraining is similar. In some instances, the RD3 return to accuracy is better when comparing Baseline Method trials, such as cases 1-K (Baseline) at 96%, and 2-B (RD3) at 99%.

Method performance diverges when assessing initial detection precision. The RD3 Method generally lags the Baseline Method by 3 to 4% when comparing the standard parameter trials. Reducing the RD3 trigger parameters to lower detection thresholds will increase detection sensitivity, and consequently, detection precision. Therefore, with additional tuning, the performance of RD3 is fully comparable to the Baseline Method.

Table 19. Comparison of Select Trials

Trial Identifier	Dataset	Pct Detect From Actual Drift	Return To Accuracy
1-F	Inc Drift (fast)	5%	99.5%
2-H	Inc Drift (fast)	7%	99.0%
1-I	Inc Drift (slow)	3%	99.0%
2-N	Inc Drift (slow)	7%	99.5%
1-K	KDD	1%	96.0%
2-B	KDD	8%	99.0%

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

A. CONCLUSION

The objective of our research was to develop an automated concept drift detection and mitigation algorithm. We proposed a novel detection scheme, RD3, which utilized supervised machine learning to develop a prediction model, paired with an unsupervised drift detection algorithm. Inherent to this proposal was the need for the scheme to demonstrate detection accuracy and precision commensurate with proven drift detection models. Furthermore, the detection scheme must automatically trigger a responsive mitigation algorithm. The novel detection scheme was validated through numerous testing trials, which were analyzed through the lens of selected, pertinent metrics.

Our baseline approach implemented a modified version of a proven detection scheme, first proposed by Gama et al. [16]. We validated suitable performance of the Baseline Method and applied our experience with extending Gama's detection method to incorporate retraining when developing the novel RD3 Method. The RD3 Method uses an ensemble of classifiers to detect drift through dimensionality reduction and Euclidean distance measures. The RD3 Method was subjected to a series of trials to assess concept drift detection. The novel method was validated through three separate datasets, containing different types of concept drift.

Finally, we assessed the performance of the RD3 Method alongside the proven Baseline Method, using identical datasets, parameters, and settings. The RD3 Method was found to be comparable to the Baseline Method in nearly all metric aspects. Furthermore, the RD3 Method was programmed with many adjustable parameters, providing the novel approach with adaptability to different datasets and situations.

B. FUTURE WORK

There are several ways our concept drift detection and mitigation research may be continued and advanced. The RD3 detection and mitigation algorithm was tested using three different types of concept drift, but there are many other ways a concept changes. These ways include both class specific drift and reoccurring drift. Testing with additional

drift types will provide further insights on parameter tuning, as well as further stress testing the algorithm.

Detecting novel classes in a data stream is critical, but difficult. Both methods developed in our research implicitly detect novel class concept drift, through misclassification of novel class data points. An explicit process to detect novel classes would greatly increase the value of the RD3 algorithm. Potential avenues to accomplish novel class detection include cluster analysis of the PCA vectors and concurrent comparison of different classes through PCA.

APPENDIX A. TYPES OF CONCEPT DRIFT—CURRENT DEFINITIONS

In 2016, Webb et al., developed several new measures to characterize concept drift as well as more nuanced definitions for existing terms [5]. Figure 23 depicts a taxonomy of concept drift categories and types. This section contains a general summary of each taxonomy category and additional related drift quantities.

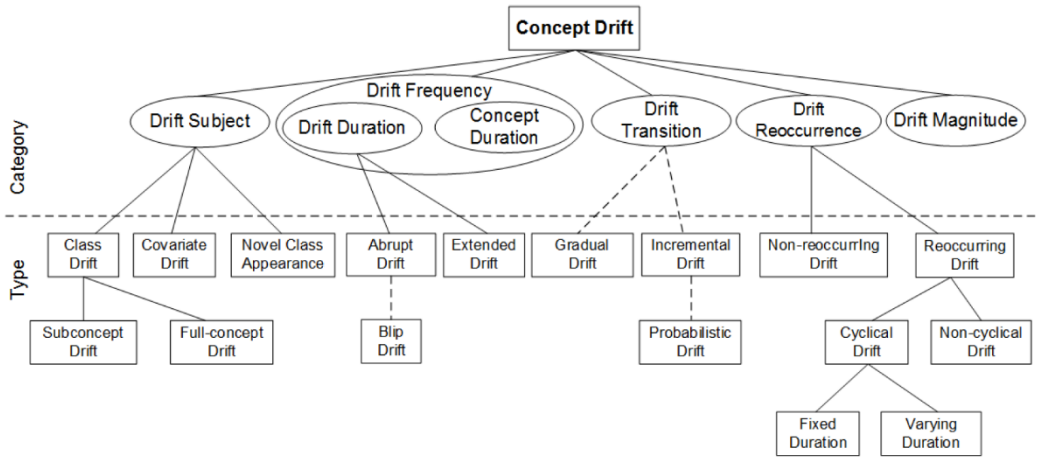

Figure 23. Concept Drift Taxonomy. Source: [5].

A. DRIFT MAGNITUDE

Drift magnitude, Mag , is the total magnitude of the distance between concepts once drift has occurred [5]. The magnitude is a relative value, based on dataset type and situation. For example, if prediction accuracy measures are used to assess drift in a dataset, the accuracy difference between the past and current concept measures is the magnitude. Drift magnitude is given by [5]

$$Mag_{t,u} = D(t,u), \tag{17}$$

where D is the distance measure function between two different concepts in the same dataset.

Drift magnitude can be nuanced, and situation dependent. For example, a threshold can be employed separating minor drift and major drift. The threshold value is relative, depending on the situation, where the magnitude determines if the current concept model is still effective or must be discarded [5].

B. DRIFT DURATION

Drift duration, Dur, is the total time from when concept drift begins to when it ends [5]. Drift duration is a time measurement, given by [5]

$$Dur_{t,u} = u - t \,. \tag{18}$$

C. PATH LENGTH

Path length, PL, is the total length of the path drift traverses when concepts are changing. It accounts for the total deviation encountered along the way [5]. For example, the total drift magnitude between two concepts may be relatively small, but if the changing concept fluctuates significantly during the transition, the path length will be large. The path length over n points is given by [5]

$$PL_{t,u} = \lim_{n\to\infty} \sum_{k=0}^{n-1} D\left(t + \frac{k}{n}(u-t), t + \frac{k+1}{n}(u-t)\right). \tag{19}$$

D. DRIFT RATE

Drift rate, R, is the rate at which drift change occurs [5]. A good example is the difference between gradual and abrupt drift. Gradual drift occurs at a slow or small rate while abrupt drift occurs at a fast or large rate. The drift rate at a given time t is represented by [5]

$$R_t = \lim_{n\to\infty} nD(t - 0.5/n, t + 0.5/n) \,. \tag{20}$$

The average drift rate, over the drift duration, is given by [5]

$$PL_{t,u} / (u - t) \,. \tag{21}$$

E. DRIFT SUBJECT

The circumstances by which concept drift occurs vary. This includes both the pieces of the dataset that drift, as well as the way the drift occurs [5].

A single class may be subject to drift. For example, a dataset containing several classes may encounter drift in only one class. This class may undergo a comprehensive distributional shift, which is full-concept, or a narrow number of aspects may change, which is sub-concept drift.

Covariate drift refers to a distributional change among independent variables. In the datasets used in this research, this type of drift affects multiple classes. For example, a feature vector contains information for each data point, which encompasses all classes in the dataset. If the data in one or more feature vectors undergoes distribution shift affecting the entire vector, it will affect multiple classes in the dataset.

Novel class appearance is particularly interesting. In this case, a new class enters the data stream or dataset [5]. The distributional changes are different, based on other types of concept drift. Rather than changes to the underlying data of an existing class, new distributions are added. Many detection methods assume a set number of classes and are unable to identify additional classes [27].

F. DRIFT FREQUENCY

Drift frequency refers to the temporal distance between drift occurrences. Drift frequencies can occur at different durations, such as abrupt and extended drift. Abrupt drift is immediate while extended drift is gradual. Both durations may have the same overall effect on the dataset. For example, a dataset may encounter abrupt drift at one interval and a short time later encounter extended drift. The time difference between occurrences is the frequency while abrupt and extended describe the rate of each [5].

G. DRIFT TRANSITION

Drift transition refers to the way one concept changes to another. The transition may be gradual or incremental distributional shifts. Gradual changes may be small, or relatively large and abrupt, while fluctuating towards or away from the original concept.

Conversely, incremental transition involves a relatively slow but ever-increasing drift away from the original concept [5].

H. DRIFT REOCCURRENCE

Drift reoccurrence is the re-entry of a pre-existing concept. This type of drift can be compared to repeating seasonal shifts in yearly weather cycles. The analogy is especially accurate if the recurrent drift has fixed duration and frequency. Recurrent drift can also occur at variable intervals and for variable durations [5].

APPENDIX B. DATA CLASSIFICATION AND CONCEPT DRIFT DETECTION

Machine learning classification is the automated process of arranging data into associated classes, which is key to understanding diverse datasets. Classification allows us to understand complex data relationships and when the relationships change. The associations developed through classification provide the basis for concept drift detection.

Classification algorithms use a training dataset to develop a model, and then classify testing data points based on the trained model. Algorithms are trained by one of three separate learning techniques: Supervised, Unsupervised and Semi-Supervised.

A wide variety of learning algorithms are available to classify data, and extensive research has been performed in this field as it relates to concept drift. A set of rigorously researched, proven algorithms were selected for use in our research [4], [6], [28], [29].

The following information in this section briefly outlines the three learning techniques, and the associated algorithms used in this thesis.

A. SUPERVISED

Supervised learning uses training datasets where the data points types are already known. Each data point contains a label, which identifies the point's type. The dataset features are used to develop a model, and the labels ensure the correct grouping for each data point [29].

Supervised learning generally provides a good training model, as labels provide accurate grouping. If the test dataset is also labeled, the model can be analyzed for accuracy throughout the testing phase. The following five supervised learning algorithms were employed in our research.

1. Decision Tree

A decision tree (DT) is a nonmetric classification algorithm which trains models similar to an actual tree, with branches and leaves. The branches link together leaf nodes,

which form a descending tree. Figure 24 depicts a DT example. The model is formed through a series of binary questions, which can be answered yes/no or true/false [30].

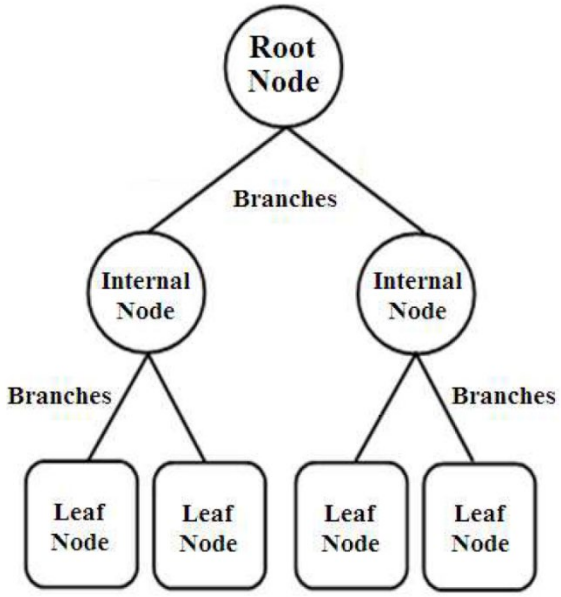

Figure 24. Decision Tree Example. Source: [31].

DTs work well through supervised learning, but diverse datasets can present problems. For example, if data varies widely, multiple binary branches may apply to the same leaf node. This is known as impurity. Mathematical impurity measures, which are not covered in this thesis, are implemented to improve prediction.

2. Random Forest

An RF algorithm combines multiple DT algorithms. For each intended tree in the RF, an independent random vector is created from a subset of the training data. Each random vector has identical probabilistic distribution. The random vector is paired with randomly selected input training data points, and a tree is formed. Once all tree models have been formed, each casts a vote for the most popular class [32].

3. Support Vector Machine

The Support Vector Machine (SVM) algorithm is a linear discriminant classifier. An SVM separates data points in multidimensional space by hyperplanes. The optimum hyperplanes are determined by maximum marginal distance between point sets. An example hyperplane is depicted in Figure 25. The optimum hyperplanes form the model used to predict data point types in the test dataset [8] and [30].

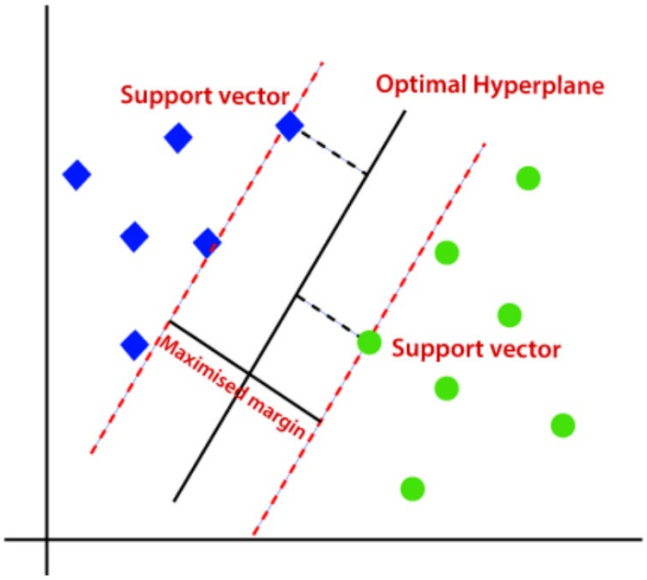

Figure 25. Support Vector Machine Hyperplane Example. Source: [33].

4. k-Nearest Neighbor

The KNN algorithm is a distance-based classifier. The algorithm develops the model by mapping the multidimensional Euclidean distance of each training set data point. The user designates a *k* value, and the Euclidean distance of each test set data point is compared to the model. The *k* number of classified training points nearest to a given test point, or the *k* nearest neighbors, determine the classification of the test point [34], [35]. Figure 26 visually depicts a test point, and *k* value of four.

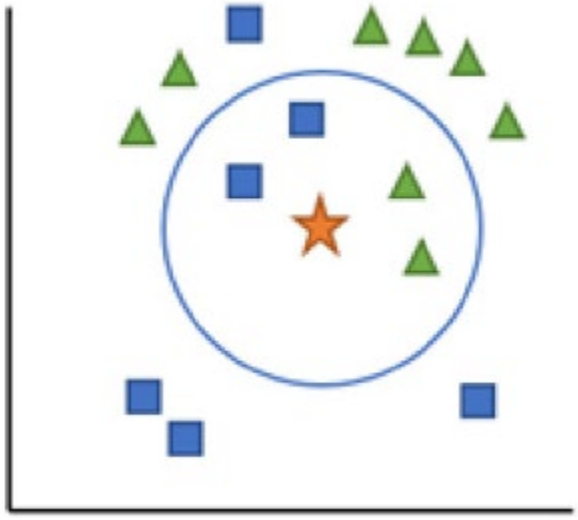

Figure 26. K-Nearest Neighbor Cluster Example ($k = 4$). Source: [35].

5. Artificial Neural Network

An ANN algorithm is modelled after neurons in a human brain. There are several types of ANNs, and this thesis uses a feed-forward neural network. A feed-forward ANN consists of input, hidden, and output layers, an example of which is depicted in Figure 27. There is always one input layer and one output layer, but there can be multiple hidden layers. The input layer connects to the hidden layer. The hidden layer approximates functions and transforms data, providing results to the output layer [9], [10], [30]. ANN are relatively complicated algorithms, and further explanation is outside the scope of this thesis.

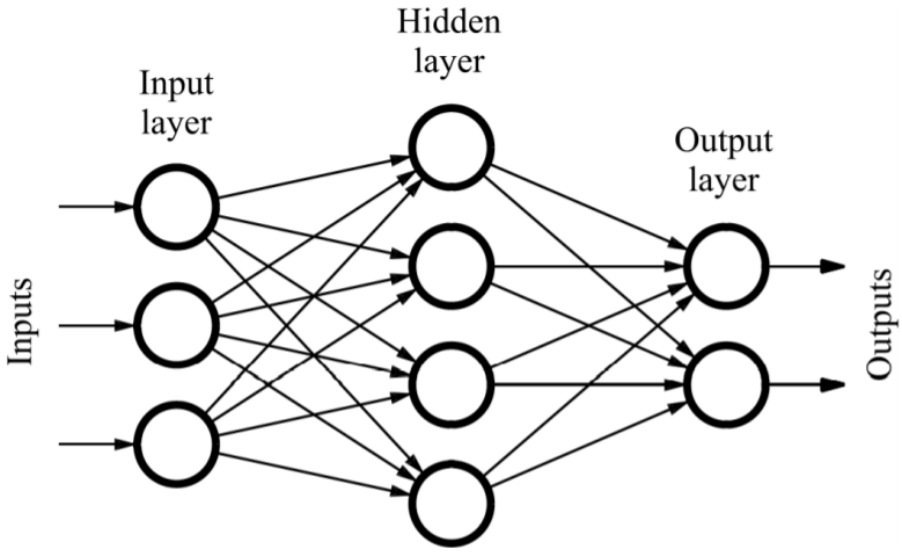

Figure 27. Feed-Forward Artificial Neural Network Example. Source: [36].

B. UNSUPERVISED

Unsupervised learning uses training datasets where the data point types are not known. In this method, clustering or component analysis algorithms are commonly used to develop models [29]. Although unsupervised learning is generally less accurate than supervised methods, unsupervised learning can identify data patterns supervised methods may miss.

A PCA algorithm was used for drift detection in this thesis. PCA algorithms use the eigenvectors of a dataset's covariance matrix to perform a change of basis transformation [26], [30]. The largest eigenvectors are calculated, which become the principal components. The top three principal components were selected in this thesis. With this process, a test matrix with more than 40 feature vectors is reduced to a principal component matrix with three vectors.

C. SEMI-SUPERVISED

Semi-supervised learning is a mix of both supervised and unsupervised methods. In this case, the training set contains both labeled and unlabeled data points. There are usually relatively few labeled points compared to unlabeled points, but the labeled points

offer increased overall model accuracy [29]. Much recent research has focused on semi-supervised learning with ANNs [6].

APPENDIX C. ATTACK CLASSES

The datasets used in our research are based on computer network traffic. The traffic is split into two types: normal network traffic and attacks on the network. Individual definitions of the attack types and classes are not necessary to understand this research, but a brief explanation of each class is provided for context.

A. PROBE ATTACK CLASS

Probing attacks attempt to access and scan a network to determine known vulnerabilities and gather information about the system. The vulnerabilities may be exploited later in a separate attack [37].

B. DENIAL OF SERVICE ATTACK CLASS

Denial of Service (DOS) attacks are designed to overwhelm the resources of a target host or network. The attacker usually generates a large amount of pointless network traffic, which floods and disables the target [37].

C. USER TO ROOT ATTACK CLASS

The User to Root (U2R) attack starts with normal user access, as the attacker looks for network vulnerabilities. The goal is to gain administrator, or root, access to the system [15].

D. REMOTE TO LOCAL ATTACK CLASS

Remote to Local (R2L) attacks target remote systems to gain access to an internal network. The attacker does not have user access to the internal system [15].

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. EXECUTION TIME METRIC ANALYSIS

A. EXECUTION TIME METRIC

The execution time metric measures the amount of time it takes to run key parts of the concept drift detection and mitigation program. The execution time represents computation resources expended by the computer. We ask: what primary factors influence execution times? Answering this question allows us to identify ways to make our detection and mitigation algorithms more efficient.

B. EXECUTION TIMES

Execution times are the time intervals it takes for a drift detection algorithm to perform key programming functions. Three areas were evaluated: the time to model the first training set, the time to perform the first model test cycle, and the overall algorithm run time. Each process execution time is given by

$$p_{endclock} - p_{startclock} = p_{exec\_time}, \tag{22}$$

where $p_{startclock}$ is the time when the process starts, $p_{endclock}$ is the time when the process ends. The measurements are taken in seconds. In the case of the overall algorithm run time, the measurement starts when the program initiates, and ends when the program completes.

Each trial was performed discretely, on the same computer with no other programs running, to provide measurement uniformity. Execution times are used to evaluate the execution time metric.

C. RETRAIN TOTALS

Retrain totals are the number of retraining cycles triggered by the detection algorithm. Retraining should occur following a substantial concept change. For example, retraining should ideally occur one time in abrupt concept drift, following the concept change. Further retraining indicates an issue with the detector. Conversely, retraining should occur more frequently during incremental concept drift once the concept starts

shifting. The number of retrain cycles was tallied and recorded for each respective trial. Retrain totals are primarily used to evaluate the execution time metric.

D. EXECUTION TIME METRIC ANALYSIS

Drift detection and mitigation programs carry a computational burden associated with the repeated testing and retraining cycles. In this section we examine the time it takes to run key portions of the detection program, in order to identify means to reduce resource consumption. Table 20 contains the compiled timing measurements of each key element.

Table 20. Execution Time Measurements

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Total Execution Time (secs)	Initial Training Cycle Execution Time (secs)	Initial Test Cycle Execution Time (secs)
1-A	KDD	**5000**	1000	500	10	5	on	457	52	51
1-B	KDD	10000	**500**	**250**	10	5	on	1954	92	72
1-C	KDD	10000	1000	500	10	5	on	1255	90	58
1-D	Inc Drift (fast)	**5000**	1000	500	10	5	on	235	9	77
1-E	Inc Drift (fast)	10000	**500**	**250**	10	5	on	583	17	141
1-F	Inc Drift (fast)	10000	1000	500	10	5	on	527	18	140
1-G	Inc Drift (slow)	**5000**	1000	500	10	5	on	457	9	92
1-H	Inc Drift (slow)	10000	**500**	**250**	10	5	on	986	15	134
1-I	Inc Drift (slow)	10000	1000	500	10	5	on	738	15	132
1-J	KDD	10000	**2000**	**1000**	10	5	on	678	88	53
1-K	KDD	10000	**3000**	**1500**	10	5	on	394	90	53
2-A	KDD	10000	**500**	**250**	**-**	**-**	**on (mod)**	813	90	87
2-B	KDD	10000	1000	500	**-**	**-**	**on (mod)**	307	89	73
2-C	KDD	10000	**2000**	**1000**	**-**	**-**	**on (mod)**	280	90	65
2-D	KDD	10000	**3000**	**1500**	**-**	**-**	**on (mod)**	255	92	62
2-E	Inc Drift (fast)	**1000**	1000	500	10	5	on	116	1	47
2-F	Inc Drift (fast)	**5000**	1000	500	10	5	on	249	8	84
2-G	Inc Drift (fast)	10000	**500**	**250**	10	5	on	995	16	148
2-H	Inc Drift (fast)	10000	1000	500	10	5	on	609	16	136
2-I	Inc Drift (fast)	10000	**2000**	**1000**	10	5	on	366	17	130
2-J	Inc Drift (fast)	10000	1000	500	**5**	**1**	**off**	660	16	136
2-K	Inc Drift (slow)	**1000**	1000	500	10	5	on	200	1	47
2-L	Inc Drift (slow)	**5000**	1000	500	10	5	on	448	9	99
2-M	Inc Drift (slow)	10000	**500**	**250**	10	5	on	1423	15	160
2-N	Inc Drift (slow)	10000	1000	500	10	5	on	875	15	147
2-O	Inc Drift (slow)	10000	**2000**	**1000**	10	5	on	547	16	141
2-P	Inc Drift (slow)	10000	1000	500	**5**	**1**	**off**	1162	16	158

The initial training cycle develops the first concept model used for drift detection. While we need to consider the initial training time in our analysis, it is worth pointing out that this training phase is performed for any machine learning approach for processing streaming data, whether drift detection is later implemented or not. Table 21 contains the time measurements for the initial training cycle, ordered from the smallest to largest time measurements. We analyze two parameters: training set size and dataset type.

Smaller training set sizes are directly related to the initial training execution time, which is not surprising, given that fewer data points are used to create a model. Six of the seven smallest training sets are the fastest model training time. Smaller training sets clearly lead to faster training times.

The dataset type also plays a large role in training time. The longest process times are all associated with the abrupt drift dataset. This is likely due to the larger number of attack types contained in the KDD dataset, compared to the incremental drift sets. Regardless of the exact reason, the contrast between dataset training times clearly indicates that different datasets result in different training times.

Table 21. Initial Training Cycle Time Measurements

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Initial Training Cycle Execution Time (secs)
2-E	Inc Drift (fast)	**1000**	1000	500	10	5	on	1
2-K	Inc Drift (slow)	**1000**	1000	500	10	5	on	1
2-F	Inc Drift (fast)	**5000**	1000	500	10	5	on	8
1-D	Inc Drift (fast)	**5000**	1000	500	10	5	on	9
1-G	Inc Drift (slow)	**5000**	1000	500	10	5	on	9
2-L	Inc Drift (slow)	**5000**	1000	500	10	5	on	9
1-H	Inc Drift (slow)	10000	**500**	**250**	10	5	on	15
1-I	Inc Drift (slow)	10000	1000	500	10	5	on	15
2-M	Inc Drift (slow)	10000	**500**	**250**	10	5	on	15
2-N	Inc Drift (slow)	10000	1000	500	10	5	on	15
2-G	Inc Drift (fast)	10000	**500**	**250**	10	5	on	16
2-H	Inc Drift (fast)	10000	1000	500	10	5	on	16
2-J	Inc Drift (fast)	10000	1000	500	**5**	**1**	**off**	16
2-O	Inc Drift (slow)	10000	**2000**	**1000**	10	5	on	16
2-P	Inc Drift (slow)	10000	1000	500	**5**	**1**	**off**	16
1-E	Inc Drift (fast)	10000	**500**	**250**	10	5	on	17
2-I	Inc Drift (fast)	10000	**2000**	**1000**	10	5	on	17
1-F	Inc Drift (fast)	10000	1000	500	10	5	on	18
1-A	KDD	**5000**	1000	500	10	5	on	52
1-J	KDD	10000	**2000**	**1000**	10	5	on	88
2-B	KDD	10000	1000	500	**-**	**-**	**on (mod)**	89
1-C	KDD	10000	1000	500	10	5	on	90
1-K	KDD	10000	**3000**	**1500**	10	5	on	90
2-A	KDD	10000	**500**	**250**	**-**	**-**	**on (mod)**	90
2-C	KDD	10000	**2000**	**1000**	**-**	**-**	**on (mod)**	90
1-B	KDD	10000	**500**	**250**	10	5	on	92
2-D	KDD	10000	**3000**	**1500**	**-**	**-**	**on (mod)**	92

The initial test cycle time is applicable with our research but would not apply to real-time data stream detection. This is because the two methods employed in our research initially analyzes the entire test set prior to concept drift algorithm employment. This provides us the ability to view concept drift in the entire test set before mitigation is employed. Therefore, the initial test cycle time can be measured and compared from each trial. The results, contained in Table 22, are not as clear as with the training process. Larger

test chunks sizes result in fewer test chunks partitioned from a test dataset, relative to smaller chunk sizes. In general, most of the larger test chunk sizes are associated with faster process times, which may be a result of fewer test chunks, but the effect on timing is not conclusive.

The model size, based on the number of training set points, likely affects testing time. In this case, the trials with 1,000 training set points have the fastest initial testing times. This may be a result of smaller or less complex detection models.

Table 22. Initial Test Cycle Time Measurements

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Initial Test Cycle Execution Time (secs)
2-E	Inc Drift (fast)	**1000**	1000	500	10	5	on	47
2-K	Inc Drift (slow)	**1000**	1000	500	10	5	on	47
1-A	KDD	**5000**	1000	500	10	5	on	51
1-J	KDD	10000	**2000**	**1000**	10	5	on	53
1-K	KDD	10000	**3000**	**1500**	10	5	on	53
1-C	KDD	10000	1000	500	10	5	on	58
2-D	KDD	10000	**3000**	**1500**	-	-	**on (mod)**	62
2-C	KDD	10000	**2000**	**1000**	-	-	**on (mod)**	65
1-B	KDD	10000	**500**	**250**	10	5	on	72
2-B	KDD	10000	1000	500	-	-	**on (mod)**	73
1-D	Inc Drift (fast)	**5000**	1000	500	10	5	on	77
2-F	Inc Drift (fast)	**5000**	1000	500	10	5	on	84
2-A	KDD	10000	**500**	**250**	-	-	**on (mod)**	87
1-G	Inc Drift (slow)	**5000**	1000	500	10	5	on	92
2-L	Inc Drift (slow)	**5000**	1000	500	10	5	on	99
2-I	Inc Drift (fast)	10000	**2000**	**1000**	10	5	on	130
1-I	Inc Drift (slow)	10000	1000	500	10	5	on	132
1-H	Inc Drift (slow)	10000	**500**	**250**	10	5	on	134
2-H	Inc Drift (fast)	10000	1000	500	10	5	on	136
2-J	Inc Drift (fast)	10000	1000	500	**5**	**1**	**off**	136
1-F	Inc Drift (fast)	10000	1000	500	10	5	on	140
1-E	Inc Drift (fast)	10000	**500**	**250**	10	5	on	141
2-O	Inc Drift (slow)	10000	**2000**	**1000**	10	5	on	141
2-N	Inc Drift (slow)	10000	1000	500	10	5	on	147
2-G	Inc Drift (fast)	10000	**500**	**250**	10	5	on	148
2-P	Inc Drift (slow)	10000	1000	500	**5**	**1**	**off**	158
2-M	Inc Drift (slow)	10000	**500**	**250**	10	5	on	160

The overall program time is measured from when the drift detection and mitigation program starts running, until the entire program is complete. This includes all training, testing, and mitigation cycles. Table 23 contains the total program time measurements. There are two parameters with clear relationship to execution time: training set size and retrain cycles.

The four shortest program execution times are related to smaller training set sizes, while trials with the largest training set, 10,000 points, are associated with the longest execution times. This is consistent with the initial training run times.

Unsurprisingly, the trials with greater numbers of retraining cycles have the longest overall execution time, and longer execution times may reduce responsiveness of drift detection in a data stream. Each retraining cycle is an additional training evolution near or equal to the initial training time. The more sensitive a detector is, the more likely a retraining cycle is to be triggered, which may be excessive in some cases. The number of retraining cycles may be reduced through parameter tuning, such as adjusting the training set size appropriately.

Overall, both methods performed comparably in execution time. This is due to much of the computational requirements coming from training and retraining, which are performed in the same fashion for both detectors. Numerous retraining cycles are an issue with both detectors, if not properly tuned, respectively.

Table 23. Total Program Execution Time Measurements

Trial Identifier	Dataset	Training Set Size (data points)	Initial Test Chunk Size (data points)	Retest Chunk Size (data points)	Warning Triggers	Detect Triggers	Massive Drift Detect	Total Execution Time (secs)	Number of Retrains
2-E	Inc Drift (fast)	**1000**	1000	500	10	5	on	116	5
2-K	Inc Drift (slow)	**1000**	1000	500	10	5	on	200	6
1-D	Inc Drift (fast)	**5000**	1000	500	10	5	on	235	4
2-F	Inc Drift (fast)	**5000**	1000	500	10	5	on	249	4
2-D	KDD	10000	**3000**	**1500**	-	-	**on (mod)**	255	1
2-C	KDD	10000	**2000**	**1000**	-	-	**on (mod)**	280	1
2-B	KDD	10000	1000	500	-	-	**on (mod)**	307	1
2-I	Inc Drift (fast)	10000	**2000**	**1000**	10	5	on	366	2
1-K	KDD	10000	**3000**	**1500**	10	5	on	394	1
2-L	Inc Drift (slow)	**5000**	1000	500	10	5	on	448	5
1-A	KDD	**5000**	1000	500	10	5	on	457	6
1-G	Inc Drift (slow)	**5000**	1000	500	10	5	on	457	6
1-F	Inc Drift (fast)	10000	1000	500	10	5	on	527	3
2-O	Inc Drift (slow)	10000	**2000**	**1000**	10	5	on	547	3
1-E	Inc Drift (fast)	10000	**500**	**250**	10	5	on	583	4
2-H	Inc Drift (fast)	10000	1000	500	10	5	on	609	4
2-J	Inc Drift (fast)	10000	1000	500	**5**	**1**	**off**	660	5
1-J	KDD	10000	**2000**	**1000**	10	5	on	678	2
1-I	Inc Drift (slow)	10000	1000	500	10	5	on	738	5
2-A	KDD	10000	**500**	**250**	-	-	**on (mod)**	813	4
2-N	Inc Drift (slow)	10000	1000	500	10	5	on	875	5
1-H	Inc Drift (slow)	10000	**500**	**250**	10	5	on	986	6
2-G	Inc Drift (fast)	10000	**500**	**250**	10	5	on	995	7
2-P	Inc Drift (slow)	10000	1000	500	**5**	**1**	**off**	1162	6
1-C	KDD	10000	1000	500	10	5	on	1255	7
2-M	Inc Drift (slow)	10000	**500**	**250**	10	5	on	1423	8
1-B	KDD	10000	**500**	**250**	10	5	on	1954	10

APPENDIX E: MODEL FLOWCHARTS

This appendix contains flowcharts which describe the data pre-processing and overview of the process flow for the Baseline Method in Figure 28 and RD3 Method in Figure 29. Each flowchart outlines the overall process, from user input to algorithm output.

The model code for both concept drift detection methods developed in our research is available upon request. Anaconda Navigator (version 1.9.12) was the graphical user interface used to implement Python coding. Jupyter Notebook (version 6.0.3) was used as the coding and testing environment, which executed the Python (version 3.8.3) programming language.

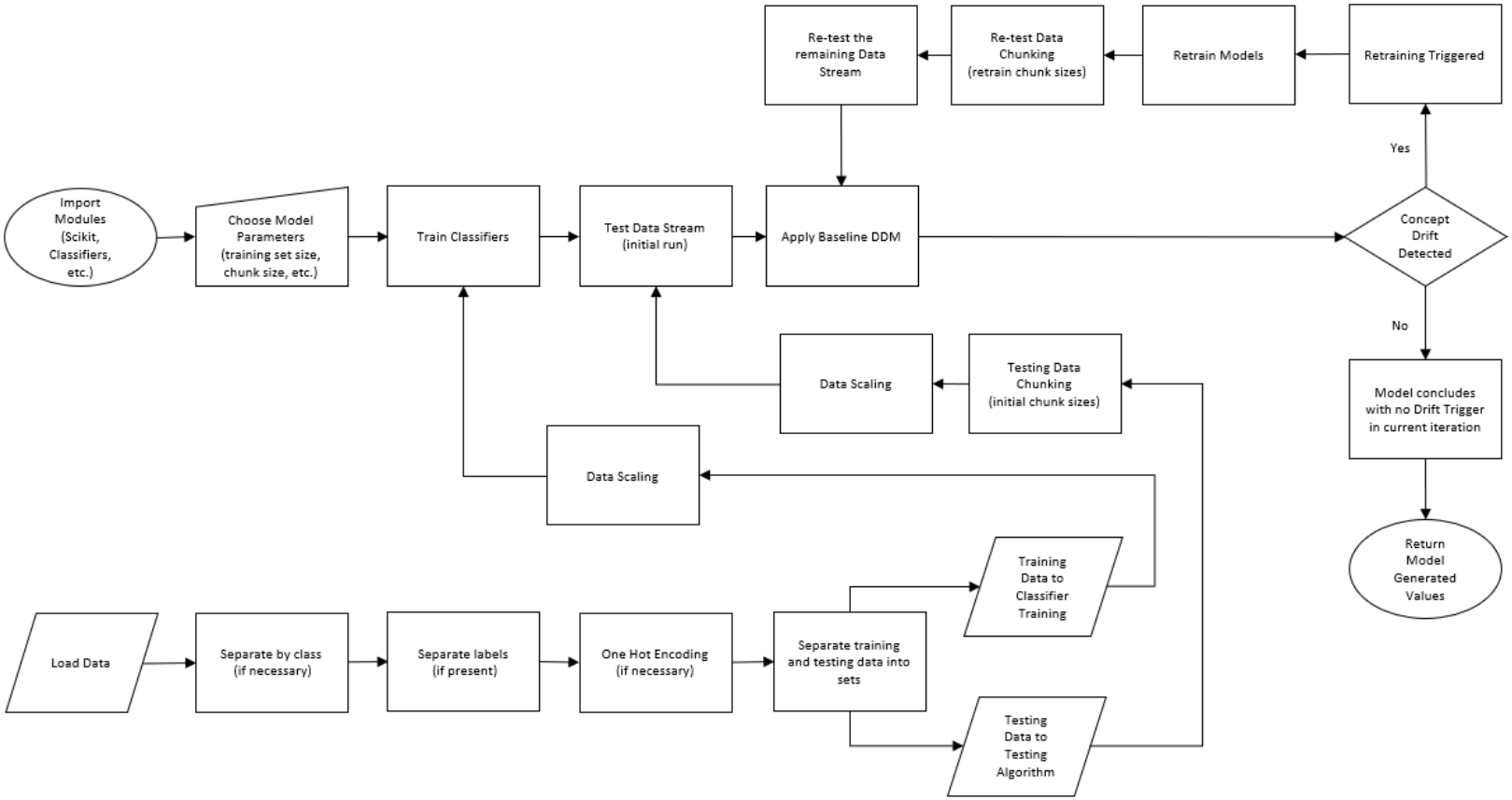

Figure 28. Baseline Method Flowchart

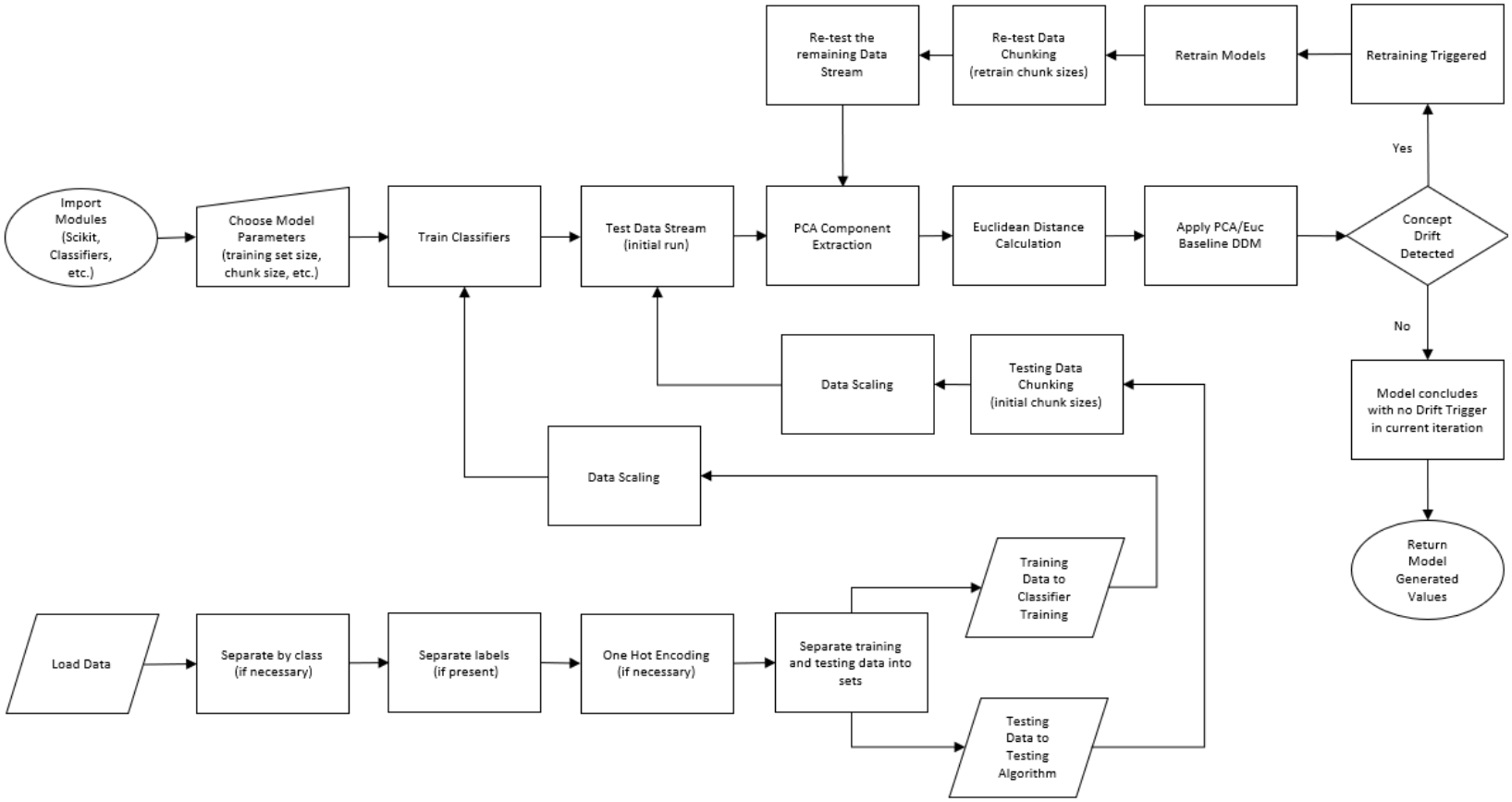

Figure 29. RD3 Flowchart

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] I. Žliobaitė, "Learning under concept drift: An overview," Vilnius University, Lithuania, Rep:1010:4784, 2010.

[2] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," *Big Data Analysis: New Algorithms for a New Society. Studies in Big Data*, vol 16, N. Japkowicz, J. Stefanowski, Eds. Springer, Cham., 2016, pp 91–114.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, Apr. 2014. [Online]. doi: 10.1145/2523813

[4] A. S. Iwashita and J. P. Papa, "An overview on concept drift learning," *IEEE Access*, vol. 7, no. 99, pp. 1532–1547, 2019. [Online]. doi: 10.1109/ACCESS.2018.2886026

[5] G. Webb, R. Hyde, H. Cao, H. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, Jul. 2016. [Online]. doi: 10.1007/s10618-015-0448-4

[6] A. Pesaranghader, H. Viktor, and E. Paquet, "Reservoir of diverse adaptive learners and stacking fast Hoeffding drift detection methods for evolving data streams," *Machine Learning*, vol. 107, no. 11, pp. 1711–1743, Nov. 2018. [Online]. doi: 10.1007/s10994-018-5719-z

[7] M. Baena-García, J. Campo-Ávila, José, R. Fidalgo-Merino, A. Bifet, R. Gavald, R. Morales-Bueno, "Early drift detection method," in *Proc. of the Fourth International Workshop on Knowledge Discovery from Data Streams*, Berlin, Germany, 2006. [Online]. Available: http://www.ecmlpkdd2006.org/ws-kdds.pdf

[8] X. Yuan, R. Wang, Y. Zhuang, K. Zhu, and J. Hao, "A concept drift based ensemble incremental learning approach for intrusion detection," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Jul. 2018, pp. 350–357. [Online]. doi: 10.1109/Cybermatics_2018.2018.00087

[9] H. Yu and G. I. Webb, "Adaptive online extreme learning machine by regulating forgetting factor by concept drift map," *Neurocomputing*, vol. 343, pp. 141–153, May 2019. [Online]. doi: 10.1016/j.neucom.2018.11.098

[10] C. A. S. da Silva and R. A. Krohling, "Semi-supervised online elastic extreme learning machine with forgetting parameter to deal with concept drift in data streams," in *Proc. of the 2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, 2019, pp. 1–8. [Online]. doi: 10.1109/IJCNN.2019.8852361

[11] S. Park, S. Seo, C. Jeong, and J. Kim, "Network intrusion detection through online transformation of eigenvector reflecting concept drift," in *Proc. of the First International Conference on data science, e-learning and information systems*, Oct. 2018, pp. 1–4. [Online]. doi: 10.1145/3279996.3280013

[12] I. Goldenberg and G. I. Webb, "Survey of distance measures for quantifying concept drift and shift in numeric data," *Knowledge and Information Systems*, vol. 60, pp. 1–25, 2018. Available: http://libproxy.nps.edu/login?url=https://search-proquest-com.libproxy.nps.edu/docview/204300488?accountid=12702.

[13] G. I. Webb, L. K. Lee, F. Petitjean, and B. Goethals. "Understanding concept drift," Australian Research Council, Canberra, Australia, Rep. DP140100087, 2017.

[14] G. Webb, L. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199, Sep. 2018. [Online]. doi: 10.1007/s10618-018-0554-1

[15] D. Protić, "Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets," *Military Technical Courier*, vol. 66, no. 3, pp. 580–596, Jul. 2018. [Online]. doi: 10.5937/vojtehg66-16670

[16] J. Gama, P. Medas, G. Castillo and P. Rodrigues, "Learning with drift detection," In *Proc. Advances in Artificial Intelligence – SBIA 2004*, 2004, pp. 286–295.

[17] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*. Boston, MA, USA: Springer U.S., 2010.

[18] J. Gama, Knowledge Discovery from Data Streams. Boca Raton, FL, USA: Chapman & Hall/CRC, 2011.

[19] E. Spinosa, A. Carvalho, and J. Gama, "OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams," in *Symposium on Applied Computing*, Seoul, South Korea, 2007.

[20] L. Minku, A. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, May 2010. [Online]. doi: 10.1109/TKDE.2009.156

[21] Scikit-learn, “One Hot Encoder,” Accessed September 7, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[22] J. Gama, P. Rodriguez, “An overview of mining data streams,” in *Foundations of Computational Intelligence Volume 6: Data Mining*, Heidelberg, Germany: Springer Berlin Heidelberg, 2009, pages 29–45.

[23] Scikit-learn, “Pre-Processing Data,” Accessed September 9, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html

[24] D. Stevanovic and N. Vlajic, “Next generation application-layer DDoS defences: Applying the concepts of outlier detection in data streams with concept drift,” in *2014 13th International Conference on Machine Learning and Applications*, Dec. 2014, pp. 456–462. [Online]. doi: 10.1109/ICMLA.2014.80

[25] K. Nishida and K. Yamauchi, “Detecting concept drift using statistical testing,” in *Discovery Science: 10th International Conference, DS 2007 Sendai, Japan, October 1–4, 2007. Proceedings, vol. 4755*, Heidelberg, Germany: Springer Berlin Heidelberg, 2007, pp. 264–269.

[26] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, Second edition. New York, NY: Pearson Education, Inc., 2019.

[27] T. Al-Khateeb, M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham, “Stream classification with recurring and novel class detection using class-based ensemble,” in *2012 IEEE 12th International Conference on Data Mining*, Dec. 2012, pp. 31–40. [Online]. doi: 10.1109/ICDM.2012.125

[28] Janardan and S. Mehta, “Concept drift in streaming data classification: Algorithms, platforms and issues,” *Procedia Computer Science*, 2017, vol. 122, pp. 804–811. [Online]. doi: 10.1016/j.procs.2017.11.440

[29] T.-M. Huang, V. Kecman, and I. Kopriva, *Kernel Based Algorithms for Mining Huge Data Sets Supervised, Semi-supervised, and Unsupervised Learning*, 1st ed. Heidelberg, Germany: Springer Berlin Heidelberg, 2006.

[30] “R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, New York, USA: John Wiley & Sons, 2001.

[31] J. Sá, A. Almeida, B. Pereira da Rocha, M. Mota, J. De Souza, and L. Dentel, “Lightning forecast using data mining techniques on hourly evolution of the convective available potential energy,” in *Proc. of the 10th Brazilian Congress on Computational Intelligence*, 2011. [Online]. doi: 10.21528/CBIC2011-27.1

[32] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. doi: 10.1023/A:1010933404324

[33] Javatpoint, “Support Vector Machine Algorithm,” Accessed September 13, 2020. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[34] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992. [Online]. doi: 10.2307/2685209

[35] N. Ali, D. Neagu, and P. Trundle, “Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets,” *Springer Nature Applied Sciences*, vol. 1:1559, 2019. [Online]. doi: https://doi.org/10.1007/s42452-019-1356-9

[36] J. Davim, *Machining of Hard Materials*, London, England: Springer-Verlag London, 2011.

[37] A. Ghorbani, W. Lu and M Tavallaee, *Network Intrusion Detection and Prevention*, New York, NY, USA: Springer U.S., 2010.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California