Feature Space Modeling for Accurate and Efficient Learning From Non-Stationary Data

Ayesha Akter
Louisiana Tech University

Follow this and additional works at: https://digitalcommons.latech.edu/dissertations

Part of the Artificial Intelligence and Robotics Commons, and the Computer Engineering Commons

Recommended Citation
https://digitalcommons.latech.edu/dissertations/823

This Dissertation is brought to you for free and open access by the Graduate School at Louisiana Tech Digital Commons. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Louisiana Tech Digital Commons. For more information, please contact digitalcommons@latech.edu.
FEATURE SPACE MODELING FOR ACCURATE AND EFFICIENT
LEARNING FROM NON-STATIONARY DATA

by

Ayesha Akter, M. S.

A Dissertation Presented in Partial Fulfillment
of the Requirements of the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

November 2019
We hereby recommend that the dissertation prepared under our supervision by Ayesha Akter, M. S.,

Entitled **FEATURE SPACE MODELING FOR ACCURATE AND EFFICIENT LEARNING FROM NON-STATIONARY DATA**

be accepted in partial fulfillment of the requirements for the Degree of

**Doctor of Philosophy in Computational Analysis and Modeling**

Supervisor of Dissertation Research

Head of Department

**Computational Analysis and Modeling**

Department

Recommendation concurred in:

Advisory Committee

Approved:

Director of Graduate Studies

Dean of the Graduate School

Dean of the College
A non-stationary dataset is one whose statistical properties such as the mean, variance, correlation, probability distribution, etc. change over a specific interval of time. On the contrary, a stationary dataset is one whose statistical properties remain constant over time. Apart from the volatile statistical properties, non-stationary data poses other challenges such as time and memory management due to the limitation of computational resources mostly caused by the recent advancements in data collection technologies which generate a variety of data at an alarming pace and volume. Additionally, when the collected data is complex, managing data complexity, emerging from its dimensionality and heterogeneity, can pose another challenge for effective computational learning. The problem is to enable accurate and efficient learning from non-stationary data in a continuous fashion over time while facing and managing the critical challenges of time, memory, concept change, and complexity simultaneously.

Feature space modeling is one of the most effective solutions to address this problem. For non-stationary data, selecting relevant features is even more critical than stationary data due to the reduction of feature dimension which can ensure the best use a computational resource to produce higher accuracy and efficiency by data mining algorithms. In this dissertation, we investigated a variety of feature space modeling techniques to improve the overall performance of data mining algorithms. In particular, we built Relief based feature sub selection method in combination with data complexity
analysis to improve the classification performance using ovarian cancer image data collected in a non-stationary batch mode. We also collected time series health sensor data in a streaming environment and deployed feature space transformation using Singular Value Decomposition (SVD). This led to reduced dimensionality of feature space resulting in better accuracy and efficiency produced by Density Ration Estimation Method in identifying potential change points in data over time. We have also built an unsupervised feature space modeling using matrix factorization and Lasso Regression which was successfully deployed in conjugate with Relative Density Ratio Estimation to address the botnet attacks in a non-stationary environment.

Relief based feature model improved 16% accuracy of Fuzzy Forest classifier. For change detection framework, we observed 9% improvement in accuracy for PCA feature transformation. Due to the unsupervised feature selection model, for 2% and 5% malicious traffic ratio, the proposed botnet detection framework exhibited average 20% better accuracy than One Class Support Vector Machine (OSVM) and average 25% better accuracy than Autoencoder. All these results successfully demonstrate the effectives of these feature space models.

The fundamental theme that repeats itself in this dissertation is about modeling efficient feature space to improve both accuracy and efficiency of selected data mining models. Every contribution in this dissertation has been subsequently and successfully employed to capitalize on those advantages to solve real-world problems. Our work bridges the concepts from multiple disciplines ineffective and surprising ways, leading to new insights, new frameworks, and ultimately to a cross-production of diverse fields like mathematics, statistics, and data mining.
APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that “proper request” consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author ________________________________

Date ________ SEPTEMBER 16, 2019 ___________
DEDICATION

To my parents and family

--

You all gave me the support, courage, spirit and inspiration to dream, to hope, to love,

and above all, to live with vision.
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................................................ iii

APPROVAL FOR SCHOLARLY DISSEMINATION ........................................................................... v

DEDICATION ...................................................................................................................................... vi

LIST OF FIGURES .......................................................................................................................... xi

LIST OF TABLES ............................................................................................................................. xiii

ACKNOWLEDGMENTS .................................................................................................................... xvi

CHAPTER 1 INTRODUCTION .......................................................................................................... 1

1.1  Overview of Dissertation ......................................................................................................... 1

1.2  Data Mining Definitions and Primitives .................................................................................. 1

    1.2.1 Overview of KDD Process ................................................................................................. 1

    1.2.2 Details of Data Mining Step ........................................................................................... 3

    1.2.3 Supervised Vs. Unsupervised Learning .......................................................................... 4

    1.2.4 Stationary vs. Non- Stationary Data Mining ................................................................. 5

    1.2.5 Feature Space Modeling and Dimensionality Reduction ................................................. 6

1.3  Problem Statement, Objective and Solution Methods ............................................................ 8

1.4  Related Mathematical and statistical methods ........................................................................ 9

    1.4.1 Matrix Decomposition .................................................................................................... 9

    1.4.2 Pairwise t-Test ............................................................................................................... 10

    1.4.3 Lasso Regularized Regression ....................................................................................... 11

    1.4.4 Density Ratio Estimation ............................................................................................... 12

1.5  Associated Validation and Metrics of Performance ............................................................... 12

vii
1.5.1 k-Fold Cross Validation ................................................................. 12
1.5.2 Metrics of Performance for Balanced Dataset ......................... 12
1.5.3 Metrics of Performance for Unbalanced Dataset ....................... 14
1.6 Organization of Dissertation .......................................................... 14

CHAPTER 2 DISTANCE BASED FEATURE SPACE MODELING TO IMPROVE CLASSIFICATION PERFORMANCE .......................................................... 16

2.1 Chapter Overview ............................................................................. 16
2.2 Related Works ................................................................................. 17
2.3 Dataset Information ......................................................................... 19
2.4 Algorithm and Methodology ............................................................ 20
  2.4.1 Normalization ............................................................................. 20
  2.4.2 Filtering of Useless Features ...................................................... 20
  2.4.3 Relief Based Feature Space Model (Feature Ranking and Selection) ............................................................................. 20
  2.4.4 Data Complexity Analysis ......................................................... 21
  2.4.5 Measures of Overlap of Individual Feature Values (F1, F3) ...... 22
  2.4.6 Measures of Inseparability of Classes ((L1, L2), (N1, N2)) ......... 23
  2.4.7 Non-linearity (L3, N4) ................................................................. 23
  2.4.8 Single Classifier Vs. Ensemble Classifier .................................. 24
  2.4.9 Non-Fuzzy vs. Fuzzy Classifier .................................................. 25
  2.4.10 Fuzzy Forests .......................................................................... 25
2.5 Results ............................................................................................. 30
  2.5.1 Feature Selection ......................................................................... 30
  2.5.2 Linear Inseparability of Data ...................................................... 31
  2.5.3 Data Complexity Analysis ......................................................... 33
  2.5.4 Validation using Crisp and Fuzzy Classifiers ............................. 35
  2.5.5 Results of Runtime Complexity Analysis ................................. 38
CHAPTER 3 FEATURE SPACE TRANSFORMATION FOR CHANGE POINTS DETECTION FRAMEWORK

3.1 Chapter Overview ........................................................................................................ 42
3.2 Related Works .............................................................................................................. 43
3.3 Dataset Information .................................................................................................... 46
  3.3.1 Artificial Dataset: Type # 01 .................................................................................. 46
  3.3.2 Artificial Dataset: Type # 02 .................................................................................. 47
  3.3.3 Artificial Dataset: Type # 03 .................................................................................. 47
  3.3.4 Public Dataset ......................................................................................................... 48
3.4 Algorithm and Methodology ....................................................................................... 49
  3.4.1 Framework MCD-PuLSIF ..................................................................................... 49
  3.4.2 Defining Temporal Intervals .................................................................................. 51
  3.4.3 PCA (Principle Component Analysis) ...................................................................... 52
  3.4.4 Dissimilarity Measurement Based on Divergence .................................................. 53
  3.4.5 Mathematical Formulation of KLIEP Method ......................................................... 55
  3.4.6 Mathematical Formulation of uLSIF Method ......................................................... 57
  3.4.7 Dynamic Cutoff Point ............................................................................................ 58
  3.4.8 Validation Criteria ................................................................................................... 59
3.5 Results .......................................................................................................................... 60
  3.5.1 Results: Type # 01 Dataset .................................................................................... 61
  3.5.2 Results: Type # 02 Dataset .................................................................................... 63
  3.5.3 Results: Type # 03 Dataset .................................................................................... 64
  3.5.4 Results: Public Datasets ........................................................................................ 69
  3.5.5 Computation Complexity Analysis .......................................................................... 72
3.6 Findings and Discussion .............................................................................................. 74
CHAPTER 4 UNSUPERVISED FEATURE SPACE MODELING FOR BOTNETS DETECTION FRAMEWORK

4.1 Chapter Overview ................................................................. 76
4.2 Related Works ........................................................................ 77
4.3 Data Set Information ............................................................... 84
4.4 Algorithm and Methodology ...................................................... 84
  4.4.1 Unsupervised Feature Selection Using Matrix Factorization .......... 85
  4.4.2 Relative Density Ratio Estimation and Defining Safety Score .......... 89
  4.4.3 $k$-Fold Dynamic Threshold .................................................. 92
  4.4.4 Algorithm of Proposed Framework ........................................... 93
4.5 Results ...................................................................................... 96
4.6 Findings and Discussions ......................................................... 108

CHAPTER 5 CONCLUSION AND FUTURE WORK ............................... 109
5.1 Conclusion .............................................................................. 109
  5.1.1 Relief based feature space modeling to reduce data complexity to improve classification performance ........................................ 109
  5.1.2 SVD based feature space transformation to build a novel change detection system with better accuracy and efficiency ................................. 110
  5.1.3 Unsupervised feature space modeling to build a novel IoT botnet detection system with better accuracy and efficiency ........................................ 110
5.2 Future Work .............................................................................. 111

REFERENCES .................................................................................. 112
LIST OF FIGURES

Figure 1-1: The step diagram of the KDD process ...................................................... 2
Figure 1-2: (left) classification plot, (right) regression plot ........................................ 4
Figure 1-3: Clustering process ..................................................................................... 5
Figure 1-4: AUC-ROC curve ...................................................................................... 13
Figure 2-1: A typical illustration of a normal ovary and ovary with cancer .......... 18
Figure 2-2: Single vs. ensemble classifier ..................................................................... 24
Figure 2-3: Flow diagram for fuzzy forest classifier. (a) screening step (b) selection step (c) final step .............................................................................................................................................. 26
Figure 2-4: Performance of relief ranked features ....................................................... 31
Figure 2-5: PC1 Vs. PC2 of 796 features prior to relief feature selection ................. 32
Figure 2-6: PC1 Vs. PC2 of 39 features post relief feature selection ........................... 32
Figure 2-7: (a-h) Comparison of measures of eight data complexity descriptors pre and post feature selection process ........................................................................................................ 34
Figure 2-8: Results of module/partition membership distribution ................................ 37
Figure 2-9: Comparison of performance of non-fuzzy and fuzzy classifiers .......... 37
Figure 2-10: ROC curve (sensitivity vs specificity) for fuzzy forest ......................... 38
Figure 2-11: Comparison between actual versus theoretical runtime with respect to number of instances ....................................................................................................................... 39

Figure 3-1: Potential change points detection for type 01 datasets where KL-divergence and PE-divergence undergoes noticeable changes. (a) & (b) time vs. KL-divergence between PCA transformed data segments (original feature dimension equals to 25, and 75, respectively). (c) & (d) time vs. PE-divergence between PCA transformed data segments with original feature dimension equals to 75, and 100, respectively .......................................................................................................................... 62
Figure 3-2: Potential change points detection for type 02 datasets where KL-divergence or PE-divergence undergoes noticeable changes. (a) & (b) Time vs. KL-divergence between PCA transformed data segments with original feature dimension equals to 50 and 100 respectively. (c) & (d) Time vs. PE-divergence between PCA transformed data segments with original feature dimension equals to 25 and 75 respectively. .......................................................... 63

Figure 3-3: Potential change points detection for type 03 datasets where KL-divergence or PE-divergence undergoes noticeable changes. (a) & (b) Time vs. KL-divergence between PCA transformed data segments with original feature dimension equals to 25 and 100 respectively. (c) & (d) Time vs. PE-divergence between PCA transformed data segments with original feature dimension equals to 50 and 75 respectively. .......................................................... 65

Figure 3-4: The results of performance improvement of KLIEP method in terms of TP, FP, FN, and execution time for PCA .......................................................... 67

Figure 3-5: The results of performance improvement of uLSIF method in terms of TP, FP, FN, and execution time for PCA. .......................................................... 68

Figure 3-6: Comparison of performance improvement between KLIEP and uLSIF method in terms of TP, FP, FN, and execution time after dimensionality reduction by PCA.......................................................... 69

Figure 3-7: Comparison of experimental results of MCD-PuLSIF method with CD-LLH, CD-MKL, and CD-Area using cover type and activity datasets separately. .......... 72

Figure 3-8: Comparison of runtime against window size and feature dimension for MCD-PuLSIF.......................................................... 73

Figure 4-1: Comparison of performance in terms of F1-score of proposed IBDS framework with OSVM, LOF, and ISF for device 9 IoT devices. ...................... 106

Figure 4-2: Comparison of run time ratio of proposed IBDS framework with OSVM, LOF, and ISF for device 9 IoT devices. ........................................... 106

Figure 4-3: (a) Comparison of runtime ratio of proposed IBDS with OSVM, LOF, ISF, and Deep Autoencoder concerning window size (b) Comparison of average runtime in seconds of proposed IBDS with OSVM, and deep autoencoder concerning window size. ........................................... 107
LIST OF TABLES

Table 2-1: Data complexity analysis pre-relief feature selection........................................ 33
Table 2-2: Data complexity analysis post-relief feature selection....................................... 33
Table 2-3: Accuracy, sensitivity, and specificity obtained using 10-fold cross-validation................................. 36
Table 2-4: Actual runtime with respect to the number of instances................................. 39
Table 2-5: Summary of the state-of-the-art CAD techniques for ovarian classification . . 40
Table 3-1: Brief information about real datasets available in UCI machine repository. .. 48
Table 3-2: Change detection in multi-dimensional feature space with PCA and uLSIF . 49
Table 3-3: Comparison of performance of KLIEP on full features and PCA reduced features for four type #01 synthetic datasets................................................................. 62
Table 3-4: Comparison of performance of uLSIF on full features and PCA reduced features for four type #01 synthetic datasets.................................................................................. 63
Table 3-5: Comparison of performance of KLIEP on full features and PCA reduced features on four type #02 datasets.................................................................................................. 64
Table 3-6: Comparison of performance of uLSIF on full features and PCA reduced features on four type #02 datasets.................................................................................................. 64
Table 3-7: Comparison of performance of KLIEP on full features and PCA reduced features on four type #03 datasets.................................................................................................. 65
Table 3-8: Comparison of performance of uLSIF on full features and PCA reduced features on four type #03 datasets.................................................................................................. 66
Table 3-9: Performance improvement of KLIEP and uLSIF methods in terms of TP, FP, FN and execution time due to PCA................................................................. 66
Table 3-10: Overall comparison of performance improvement between KLIEP and uLSIF on PCA transformed data........................................................................................................... 68
Table 3-11: Comparison of experimental results of MCD-PuLSIF method with CD-LLH, CD-MKL, and CD-Area using real world datasets................................................................. 70
Table 3-12: Comparison of runtime against window size and feature dimension for MCD-PuLSIF

Table 4-1: ALGORITHM 1: framework IBDS

Table 4-2: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D1

Table 4-3: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D2

Table 4-4: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D3

Table 4-5: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D4

Table 4-6: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D5

Table 4-7: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D6

Table 4-8: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D7

Table 4-9: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D8

Table 4-10: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D9

Table 4-11: Hyperparameter used for deep autoencoder for 9 IoT devices

Table 4-12: Performance of deep autoencoder in terms of F1-score for D1, D3, D4, and D5 IoT devices

Table 4-13: Performance of Deep Autoencoder in terms of F1-score for D6, D7, and D8 IoT devices

Table 4-14: Performance of Deep Autoencoder of F1-score for D2, and D9 IoT devices

Table 4-15: Pairwise t-test results at 95% confidence level using average F1-score of proposed IBDS with OSVM, LOF, ISF, and Autoencoder for 9 IoT devices

Table 4-16: Pairwise t-test results 95% confidence level using average run time (in sec) of proposed IBDS with OSVM, LOF, ISF, and Autoencoder for 9 IoT devices
ACKNOWLEDGMENTS

First of all, I would like to thank my Creator for every blessing, for the strength I got during this period when the tide was high, and the hope was low. Special thanks to the three most important persons in my life, my mother Sakila Banu, my father Mirza Golam Ambia, and my husband A Z M Nowzesh Hasan. All of them fought their life so that I can be successful. I could never have achieved this dream without Dr. Sumeet Dua believing in me. I also would like to thank Dr. Pradeep Chowriappa, Dr. Weizhong Dai, Dr. Jean Gourd, Dr. Jinko Kanno, Dr. Collin Wick, and all the faculty at Louisiana Tech University for their wonderful courses that helped me in my studies. I especially appreciated all of the professors' ability to explain difficult concepts so well. I'd also like to thank all the staff and students that have helped me in ways that differed but were so helpful, especially Norman John Mapes Jr. and Hatwib Mugasa.
CHAPTER 1

INTRODUCTION

1.1 Overview of Dissertation

The common thread in three of the major sections CHAPTER 2, CHAPTER 3, and CHAPTER 4 are original contributions in building a variety of feature space models to improve the accuracy and efficiency of data mining algorithms and explore their applicability to solve real life problems as well as overcoming computational challenges. In the following sections of this chapter, we described necessary concepts, mathematical, statistical, and data mining tools and techniques to justify the selection of methodologies used in CHAPTER 2, CHAPTER 3, and CHAPTER 4.

1.2 Data Mining Definitions and Primitives

1.2.1 Overview of KDD Process

Knowledge discovery from data (KDD) is the automated or convenient extraction of interesting patterns implicitly stored or captured in large databases, data warehouses, the Web, other massive information repositories, and data streams [1]. Interesting patterns represent non-trivial, implicit, previously unknown and potentially useful knowledge from a huge amount of data. Data selection, data preprocessing, data transformation, data mining, patterns evaluation, and knowledge discovery are key stages involved in the KDD process as shown in Figure 1-1.
In the Data Selection stage, we collected records from existing data sources to prepare target datasets to be considered for further processing in the KDD life cycle. The records of target dataset can be either unlabeled or labeled which plays the most significant role in defining the goal of a data mining task. Unlabeled data, for example, photos, audio recordings, videos, news articles, tweets, x-rays etc., consists of samples of natural or human-created artifacts that can be obtained relatively easily from the world without any "meaningful tags". Labeled data is a group of samples that have been tagged with one or more labels. Labeling typically takes a set of unlabeled data and augments each piece of that unlabeled data with meaningful tags that are informative.

For example, labels might indicate whether a photo contains a car or a bus, which words were uttered in an audio recording, what the topic of a news article is, what the overall sentiment of a tweet is, whether the dot in an x-ray is a tumor, etc. Another
important task of the Data Selection stage is the selection of relevant features or records for applicable and effective knowledge discovery.

In the Data Preprocessing stage, a selected dataset is cleaned by removing noise and outliers, missing data fields are imputed, time sequence information and known changes are also incorporated. Data integration is also a part of the Data Preprocessing stage where data from multiple heterogenous sources may be required to combine to form a single improved dataset to improve the efficiency of data mining. Data Transformation is the next stage which involves transforming data into appropriate format suitable for specific data mining tasks. Normalization, discretization, or smoothing of data, and feature construction are some of the key methods involved in the Data Transformation stage. In the Data Mining stage, we applied intelligent data modeling techniques to extract hidden data patterns from the target dataset. We elaborated Data Mining Stage in Section 1.2.2. Evaluation is the final stage of the KDD process where extracted patterns are analyzed to discover unknown and interesting patterns from the underlying dataset which is followed by knowledge representation using visualization tools to present mined knowledge to the end users of the system.

1.2.2 Details of Data Mining Step

The KDD process mainly focuses on the development of methods and techniques for making the best sense and use of data. Data Mining is the core of this KDD process for pattern discovery and knowledge extraction [2]. In the data mining stage of the KDD process, three major type of activities are involved as follows:
Selection of the data mining task: Based on selected dataset, Data Mining task starts with deciding whether the goal of the KDD process is clustering, anomaly detection, classification, or regression, and so forth.

Selection of the data mining algorithm(s): The next task is the selection of appropriate models and parameters to be used for searching unknown and useful patterns such that the selected data mining method is matched with the overall criteria of the KDD process.

Perform actual data mining: Finally, we applied selected algorithm on a target dataset to search for patterns of interest in a representational form or a set of such representations as classification rules or trees, regression, clustering, and so forth.

1.2.3 Supervised Vs. Unsupervised Learning

The objective of supervised learning is to identify specific relationships or structure in the input data that can effectively produce correct output data. Classification and regression are two major areas of supervised learning where input variables or features are mapped to the output labels or continuous values, respectively. Common supervised learning algorithms include decision tree, k-NN, linear regression, logistic regression, support vector machines, naive Bayes, artificial neural networks, random forests, fuzzy forest, and so on as shown in Figure 1-2.

Figure 1-2: (left) classification plot, (right) regression plot
On the contrary, the goal of unsupervised learning is to identify the previously unknown patterns of chosen dataset without pre-existing target variables. Description analysis, association rule mining, clustering, anomaly detection, etc. are some major areas in unsupervised learning as shown in Figure 1-3.

![Unsupervised Learning](image)

**Figure 1-3: Clustering process**

Among unsupervised algorithms, k-means clustering, principal component analysis, and autoencoders are widely used. There is no specific way to compare model performance for most of the unsupervised learning methods due to the absence of response variables.

1.2.4 Stationary vs. Non-Stationary Data Mining

Data Mining Tasks can be significantly different based on whether the dataset is being collected in stationary or non-stationary environment. A stationary dataset is one whose statistical properties such as the mean, variance and autocorrelation are all constant over time, whereas a non-stationary data is one whose statistical properties change over time.

Traditionally, non-stationary data, for example, a data stream, arrive at a rate that does not permit to store them permanently in memory which imposes three major challenges: 1) memory management, 2) time management, and 3) detection of concept
change. Efficient memory management deals with storing and computing a small portion of useful data and discarding the rest of the information since it is impossible to store all the data at a time, whereas time Management limits the time in which an instance or batch of instances can be processed.

Concept change occurs when the distribution of data shifts from time to time after a minimum stability period. When the training and test data follow different distributions, it is difficult to learn about the test distribution from the training samples. This problem of concept change needs to be addressed to maintain the model performance within acceptable level. Due to these three challenges, data mining objective and tasks may change significantly for stationary and non-stationary data. While higher model performance is the most important goal for stationary data, computational time and memory optimization are equally important as model performance in the case of non-stationary data. However, stationary data can also be considered as a form of non-stationary data when it is collected in batch mode, stored, processed, and analyzed offline with regular interval.

1.2.5 Feature Space Modeling and Dimensionality Reduction

In the modern era, datasets collected from both stationary and non-stationary environment may be exploded with hundreds and thousands of features. Examples of such datasets are text documents, gene expression array, data from image or social, and so on. Among these numerous amounts of features, some are redundant, whereas some are irrelevant which may lead to not only poor model performance but also huge computational expense both in terms of runtime and memory. The selection of most relevant features which are capable to make maximum contributions to generate a desired output is called Feature Selection. The approach and techniques for Feature Selection are completely
different for unsupervised and supervised learning. Most of the traditional Feature Selection methods are designed for supervised learning where a dataset has certain target or response variables. Supervised Feature Selection methods are categorized into wrappers, filters, and embedded methods. Wrappers search through the feature space to identify possible important features by using search algorithms and run a model on the subset to provide scores, and evaluate each subset based on computed scores. The wrapper methods have two major drawbacks, 1) they are computationally expensive and 2) model can be overfitted. Simulated annealing, Genetic algorithm, Greedy forward selection, Greedy backward elimination, Particle swarm optimization are some popular wrapper feature selection methods.

Filter methods follow a similar search approach like wrapper methods, but instead of evaluating against a model, a single statistical measure is chosen suitable for chosen data to identify insignificant features to be filtered out. Correlation and mutual information-based methods are widely used as the filter-based feature selection methods. Embedded algorithms perform own their feature selection process, for example, Lasso Logistic Regression or Neural Network algorithm performs feature selection and classification simultaneously. Feature Subset Selection is one approach which reduces the dimension of original feature space to a significant degree. Another approach is transforming higher dimensional space into lower dimensional feature space and projecting original data onto lower dimension such that a maximum amount of information is retained while removing the redundancy. Principle Component Analysis (PCA) is one of the most widely used dimensionality reduction technique that makes linear transformation of higher dimensional
feature space into lower dimensional feature space retaining maximum original variance while removing co-variance as much as possible.

1.3 Problem Statement, Objective and Solution Methods

A non-stationary dataset is characterized by its volatile statistical properties such as non-constant mean, variance, correlation, probability distribution, etc. over time. Moreover, non-stationary data undergoes the critical challenges of time and memory management due to the limitation of computational resources mostly caused by the recent advancements in data collection technologies which generate a variety of data at an alarming pace and volume. When collected data is complex, managing data complexity, emerging from its dimensionality and heterogeneity, can pose additional challenges for effective computational learning. Under such scenarios, the overarching problem is to enable accurate and efficient learning from non-stationary data in a continuous fashion over time while facing and managing the critical challenges of time, memory, concept change, and complexity simultaneously.

The unified objective of this dissertation is to build three different feature space models to address the major problems of non-stationary data. First, we collected stationary labeled data from ovarian cancer image exhibiting very high data complexity in terms of class overlapping, feature non-linearity, class inseparability which led to considerable performance degradation of classification algorithms in differentiating between benign and malignant target classes. We solved this data complexity problem by building a Relief based feature space model to reduce the complexity of data which ultimately improved the accuracy of classifiers. Next, we collected unlabeled multi-dimensional time series data in a streaming environment with periodic change of statistical properties. We built a change
detection framework using density ratio estimation method to compute the Pearson divergence for detecting possible occurrence of change points between two data segments, but high dimensionality of data affected the ability of change detection framework both in terms of detection rate and run time. We solved this problem by using feature space transformation using PCA, and thus improved both the change detection rate and reduced run time significantly. Finally, we collected unlabeled multi-dimensional network traffic data in a non-stationary environment infused with malicious traffic launched from botnets. We built a botnet detection framework in conjugation with an unsupervised feature space model built by matrix factorization and Lasso regression to improve the accuracy and efficiency of the proposed botnet detection framework.

1.4 Related Mathematical and statistical methods

1.4.1 Matrix Decomposition

Matrix decomposition is a technique of factorization of a matrix into a product of constituent matrices. By decomposing a matrix, we can simplify more complex matrix operations on the decomposed matrix, for example lower triangular matrix or upper triangular matrix, rather than on the original matrix itself.

In linear algebra, a $QR$ decomposition is a factorization of a matrix $A$ into a product $X = QR$. Given $A$ is a data matrix with dimension $n \times m$ and rank $r$, then the $QR$ decomposition of $X$ is defined as

$$A = QR.$$  \hspace{1cm} \text{Eq. 1-1}

where $R$ is an $r \times m$ upper triangular matrix and $Q$ is an $n \times r$ column wise orthonormal matrix. Eigen Value Decomposition (EVD) of a matrix $A$ be a square $n \times n$ matrix
with \( n \) linearly independent eigenvectors be factorized as (where \( i = 1, 2, \ldots, n \)) so the matrix \( A \) can be obtained as

\[
A = Q \Lambda Q^{-1}.
\]

Eq. 1-2

where \( A \) is the square \( n \times n \) matrix whose \( ith \) column is the eigenvector \( q_i \) of \( A \), and \( \Lambda \) is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, \( \Lambda_{ii} = \lambda_{ii} \).

Singular Value Decomposition (SVD) of any \( n \times d \) matrix \( A \) can be uniquely expressed as

\[
A = U D_s V^T.
\]

Eq. 1-3

where \( U \) is a column-orthonormal \( n \times r \) matrix. \( D_s \) is a diagonal \( r \times r \) matrix where the singular values \( s_i \) are sorted in descending order, and \( V \) is a column-orthonormal \( r \times d \) matrix.

QR decomposition, EVD, and SVD are very effective mathematical methods which are widely used to reduce the dimensionality of feature space. For example, both EVD and SVD are used for PCA technique which linearly transforms higher dimensional feature space into lower dimensional feature space without losing information as much as possible. Matrix decomposition has been extensively explored for both feature space transformation and feature subset selection in Section 3.4.3 and 4.4.1 respectively.

1.4.2 Pairwise t-Test

At first, we establish a null hypothesis and alternative hypothesis \( H_0 \) and \( H_1 \) as follows:

\[
H_0: \mu_d = 0
\]

Eq. 1-4

\[
H_1: \mu_d \neq 0
\]

Eq. 1-5
where $\mu_d$ refers to the means of the difference from two selected samples with size $n$ from a dataset at a time.

$$t_0 = \frac{\bar{d}}{s_d/\sqrt{n}} \quad \text{Eq. 1-6}$$

$$\bar{d} = \frac{1}{n} \sum_{j=1}^{n} d_j \quad \text{Eq. 1-7}$$

$$d_j = y_{1j} - y_{2j} \quad \text{Eq. 1-8}$$

$$S_d = \left[ \frac{\sum_{j=1}^{n} (d_j - \bar{d})^2}{n - 1} \right]^{1/2} \quad \text{Eq. 1-9}$$

$y_{ij} (i = 1, 2)$ is the mean in group one or two at the $j^{th}$ observation. Using pairwise t-test, we can decide if a method is significantly different than another method on a collection of datasets based on predefined significance level.

1.4.3 **Lasso Regularized Regression**

In statistics, linear regression is a linear approach to modeling the relationship between a response or dependent variable and one or more predictor or independent variables. The response variable is mostly scalar, but the predictor variables can be both scalar and categorical. Simple linear regression has only one predictor variable, whereas multiple linear regression has more than one predictor variable.

In statistics and machine learning, Least Absolute Shrinkage and Selection Operator (Lasso) is a regression analysis method that performs both feature selection, regularization, and prediction which can increase the prediction accuracy and interpretability of the statistical model.
1.4.4 Density Ratio Estimation

The density ratio of two data distribution from reference window $P_{\text{ref}}(x)$ and current window $P_{\text{cur}}(x)$ is defined as follows, where $x$ is a traffic instance:

$$r(x) = \frac{P_{\text{ref}}(x)}{P_{\text{cur}}(x)}$$  \hspace{1cm} \text{Eq. 1-10}

where $x$ refers to a single data instance. The mathematical formulation of the density-ratio estimator is given as

$$\hat{r}(x) = g(x; \hat{\theta}) = \sum_{l=1}^{n_{\text{ref}}} \hat{\theta}_l K(x, x_{l\text{ref}}).$$  \hspace{1cm} \text{Eq. 1-11}

1.5 Associated Validation and Metrics of Performance

1.5.1 k-Fold Cross Validation

At first, we partition the input dataset into $k$ subsets. The size of each subset is equal. In each iteration, a single subset is selected among these $k$ subsets as the test dataset. The remaining $k - 1$ subsets are combined and used as the training dataset. The process is then repeated $k$ times with selected algorithm which produces $k$ results. The $k$ results from the $k$ iterations are averaged to produce the result. The 10-fold cross validation is a widely used technique for model validation with $k$ equal to 10. Overfitting a common problem which can be solved with $k$-fold cross validation.

1.5.2 Metrics of Performance for Balanced Dataset

Given that True Positive, True Negative, False Positive, False Negative, True Positive Rate, True Negative Rate, and False Positive Rate are denoted by TP, TN, FP, FN, TPR, TNR, and FPR respectively, some widely used statistical performance measurement criteria are defined as shown in Eq. 1-12 to Eq. 1-16:
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.

\textbf{Eq. 1-12}

Sensitivity = Recall = TPR = \frac{TP}{TP + FN}.

\textbf{Eq. 1-13}

Specificity = TNR = \frac{TN}{TN + FP}.

\textbf{Eq. 1-14}

FPR = 1 - Specificity = \frac{FP}{TN + FP}.

\textbf{Eq. 1-15}

\[\text{Precision} = \frac{TP}{TP + FP}\]

\textbf{Eq. 1-16}

AUC - ROC curve is a performance measurement for classification problem at various threshold settings. ROC is a probability curve where TPR is plotted on y-axis and FPR is plotted on the x-axis as shown in Figure 1-4.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{auc_roc_curve.png}
\caption{AUC-ROC curve}
\end{figure}

ROC curve represents the tradeoff between TPR (or sensitivity) and FPR (or specificity) if any increase in sensitivity is producing any decrease in specificity. The closer the curve to the left-hand border and then the top border of the ROC space, the more accurate the test. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test. AUC, the measure of separability, explains how much the model
can separate between classes. The higher the value of AUC, the better the model is at predicting both negative and positive classes.

### 1.5.3 Metrics of Performance for Unbalanced Dataset

\[
F1 - score = \frac{2 \times Precision}{Precision + Recall}
\]

Eq. 1-17

Selection of the appropriate performance measure for system validation is crucial for an imbalanced dataset to identify anomaly or minority classes. For imbalanced or skewed datasets, performance measures such as accuracy, TNR, FPR, ROC-AUC might go high just because of high TN, value which may show a misleading performance. On the contrary, both precision and recall do not include TN, and thus F1-score and PR-AUC are appropriate performance measures for imbalanced datasets due to not considering TN in their formulations.

### 1.6 Organization of Dissertation

The remainder of the dissertation is divided into four chapters. In **CHAPTER 1**, we have introduced important background information, definitions and explanations from the areas of mathematics, statistics, and datamining to explain different components proposed frameworks presented in **CHAPTER 2, CHAPTER 3**, and **CHAPTER 4**. **CHAPTER 2** demonstrated how analysis of Data Complexity Matrix was applied to select Fuzzy Forest as an appropriate classifier for inherently complex dataset collected from ovarian cancer images. Singular Value Decomposition (SVD) and Direct Density Ratio Estimation has been explored in **CHAPTER 3** to detect change points in multi-dimensional health sensor data in a non-stationary environment. **CHAPTER 4** explored matrix decomposition and Lasso Regression to design an unsupervised feature selection approach in combination with Relative Density Ration Estimation method to detect
BASHLITE and Mirai botnets launched from nine commercial IoT devices. **CHAPTER 5** discusses the novel contributions and directions for future research.

In the next chapter, we explored feature space modeling using Relief based feature selection as a technique of feature space modeling to reduce the complexity of ovarian cancer dataset collected in a batch mode and the less complex dataset is subsequently used by Fuzzy Classification to classify malignant and benign records with better accuracy and efficiency.
CHAPTER 2
DISTANCE BASED FEATURE SPACE MODELING TO IMPROVE CLASSIFICATION PERFORMANCE

2.1 Chapter Overview

In this chapter, we have explored Relief, a distance-based feature sub-selection method, to reduce the inherent complexity of data from 469 ovarian cancer images collected in batch mode from non-stationary environment. We subsequently employed Fuzzy Classifier on this reduced and less complex dataset to build a classification model with better accuracy. Diagnosis of ovarian cancer using ultrasonography is tedious as ovarian tumors exhibit minute clinical and structural differences between the suspicious and non-suspicious classes. Early prediction of ovarian cancer will reduce its growth rate and may save many lives. Computer aided diagnosis (CAD) is a non-invasive method for finding ovarian cancer in its early stage which can avoid patient anxiety and unnecessary biopsy.

In this chapter, we proposed a novel CAD tool for the characterization of suspicious ovarian cancer using Relief-F based feature space modeling and Fuzzy Ensemble classifier. Data complexity analysis, both pre and post future selection, indicates that instances of the two classes significantly overlap each other, thereby affecting a classifier’s ability to differentiate between instances of the normal versus the target class. We reduced the complexity of the data by applying distanced based Relief feature selection method. In this
work, we have also investigated the use of Fuzzy Forest based ensemble classifier in contrast to known crisp rule-based classifiers. The proposed frameworks is evaluated using 469 (non-suspicious: 238, suspicious: 231) subjects and achieved a maximum accuracy of 80.60 ± 0.5% accuracy, 81.40% sensitivity, 76.30% specificity with Fuzzy Forest, an ensemble fuzzy classifier using thirty-nine features. The proposed method is robust and reproducible as it used a maximum number of subjects (469) as compared to state-of-the-art techniques. Hence, it can be used as an assisting tool by gynecologists during their routine screening.

2.2 Related Works

Ovarian tumor refers to any malignant development that happens in the ovary [3]. In most cases, ovarian cancer arises from the epithelium (outer lining) of the ovary. It brings about unusual cells that can attack or spread to different parts of the body. When this starts, there might be no or just obscure side effects. Side effects turn out to be more perceptible as the growth progresses. These indications may bring about bloating, pelvic agony, stomach swelling, and loss of hunger [3]. Figure 2-1 depicts a typical representation of a normal ovary and an ovary with cancer. It can be noted that the ovary with cancer is bloated due to the cancerous cell growth in the ovary [3].

Being the eighth most regular malignancy among ladies, ovarian disease is the fifth driving reason for death among ladies and is the deadliest of gynecologic tumors [3]. A woman’s lifetime risk of developing invasive ovarian cancer is 1 in 75, whereas a woman’s lifetime risk of dying from invasive ovarian cancer is 1 in 100. Ovarian cancer rates are highest in women aged 55-64 years and the survival rates are much lower than different malignancies that influence ladies. Five-year survival rates are commonly used to compare
different cancers and the relative five-year survival rate for ovarian malignancy is 46.25%. Survival rates change incredibly relying upon the phase of the finding. Ladies analyzed at an early stage before the tumor has spread have a substantially higher five-year survival rate than those analyzed at a later stage. Around 14.80% of ovarian growth patients are determined right on time to have early stage illness [4]. Figure 2-1 depicts a typical representation of a normal ovary and an ovary with cancer.

![Figure 2-1](image-url)

**Figure 2-1:** A typical illustration of a normal ovary and ovary with cancer.

Ultrasound based Computer Aided Diagnostic (CAD) techniques can prove to be excellent adjunct techniques, especially for mass screening, because of their speed, non-invasiveness, easy usability, cost-effectiveness, and reliability [5, 6]. Reference [7] summarizes the state-of-the-art CAD systems developed for ovarian cancer diagnosis. It can be observed that mass spectrometry (MS) based systems outperform all other techniques in terms of different performance parameters [7]. These methods are restricted due to its cost and data analysis equipments. Biagoilti et al. [8] and Tailor et al. [9] have
observed that subjective features such as operator suggested parameters can achieve comparable performance. Lucidarme et al. [10] and Zimmer et al. [11] developed a model and have achieved maximum performance of 91.73% and 70% accuracy, respectively. Acharya et al. [12] have used texture features to discriminate benign and malignant US images and achieved 99.9% accuracy.

In another study [13], the same group has used Gabor wavelets features, Hu’s moments, entropies, and achieved 99.80% accuracy for Probabilistic Neural Network (PNN) classifier. The same group extended their study using higher order spectra (HOS) and achieved 97% accuracy for decision tree classifier [14]. Recently, Acharya et al. [6] have achieved about 100% accuracy using first order statistical features, gray level co-occurrence matrix (GLCM) and run length matrix. They have used PNN and $k$-nearest neighbor classifiers during classification. In [15], it is concluded that 3D ultrasonography can capture the minute morphological structures as compared to 2D ultrasonography.

### 2.3 Dataset Information

We collected preprocessed and cleaned data from ultrasonography image of 469 non-consecutive women (238: Benign, 231: Malignant) with an age limit of 23 to 90 years. The dataset has 811 features and 469 records with a binary class label (0/1) as the target variable. Excluding the target variable, the other 810 features are numeric. Class label 0 represents non-suspicious cancer, whereas class label 1 refers to suspicious Cancer. The collected dataset is clean with no missing values, but it is not normalized. The dataset is balanced with where 238 records have class label 0 and 231 records have class label 1.
2.4 Algorithm and Methodology

2.4.1 Normalization

The filtering of useless features resulted in the reduction of features to 796 features. The reduced set of features was subject to the min-max normalization. Here, all the features were subject to fit to a predefined range of [0, 1]. The normalized values $z$ of a feature $x$ was computed using

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}.$$  \hspace{1cm} \text{Eq. 2-1}

where $\min ()$ and $\max ()$ represent functions that compute the minimum and maximum values of feature $x$ respectively.

2.4.2 Filtering of Useless Features

The nonlinear feature extraction generated 810 features per instance of the dataset. We applied a filter to remove from further analysis those features that exhibit a minimum and those features that exceed the maximum threshold of variance ($\sigma^2$). Variance ($\sigma^2$) of each feature $x$ where $1 \leq x \leq 810$ was calculated using

$$\text{variance } (\sigma^2) = \frac{\sum (x - \mu)^2}{N}.$$  \hspace{1cm} \text{Eq. 2-3}

where the maximum variance threshold is set at 0.99 and the minimum threshold at 0.

2.4.3 Relief Based Feature Space Model (Feature Ranking and Selection)

The Relief-F is one of the most widely used wrapper-based feature selection algorithm [16]. To evaluate the significance of a feature, the Relief-F algorithm repeatedly samples instances and assigns a weight based on Euclidean distance to the feature relative to the class distribution of instances in the nearest neighborhood. The algorithm then sorted
the features from higher worth value to lower worth. Relief-F computes two weights namely the near-hit score and the near-miss score based on nearest instances in the neighborhood. If an instance in the nearest neighborhood belongs to the same class, Relief-F considers the feature to be relevant and assigns a higher near hit score. The weights of each feature vector are calculated and updated in iterative fashion using Euclidean distance as follows:

\[ W_i = W_i - (x - \text{nearHit}_i)^2 + (x - \text{nearMiss}_i)^2. \]

where nearHit and nearMiss are defined as the closest same-class instance and the closest different-class instance, respectively. The feature vectors are normalized after a fixed number of iterations. Features are then ranked using the updated weights and selected using a threshold \( \tau \).

2.4.4 Data Complexity Analysis

We believe that the performance of classifiers is strongly sensitive to the complexity of the dataset. Hence, we performed data complexity analysis before and after feature selection. Considering the inherent overlap of target classes, the objective of this is twofold, namely (a) to estimate the importance of the features selected, and (b) to guide us in the choice of classifiers. To measure the complexity of our dataset, we focused on three types of measures of problem complexity.

a) Measures of overlap of individual feature values: We applied two complexity descriptors to measure class boundary overlap which are maximum Fisher's Discriminant Ratio (F1), maximum (individual) feature efficiency (F3).
b) Measures of inseparability of classes: We performed linear separability measure (L1, L2), fraction of points on boundary (MST method) (N1), ratio of average intra / inter class NN distance (N2) to measure separability of classes, and
c) Measures of non-linearity: we measured L3 and N4 to measure non-linearity of classes.

2.4.5 Measures of Overlap of Individual Feature Values (F1, F3)

As per T.K. Ho [17], and considering \( n \) dimensional feature space, we compute the Fisher’s discriminant ratio for each feature with respect to the target class as follows:

\[
 f_i = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}. \tag{2-5}
\]

\[
 F1 = \max(f_i). \tag{2-6}
\]

where \( f_i \) refers to an individual feature with \( 1 \leq i \leq n \) and \( \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \) are the respective means and variances feature \( f_i \) with respect to the two classes. The derived F1 measure is defined as the maximum \( f_i \) over all the features.

Furthermore, Tin Ko [17] defines the measure F3 to measure the overlap of class boundaries with respect to feature efficiency as follows:

\[
 f_{\text{efficiency}} = \frac{N_{\text{separate}}}{N} \tag{2-7}
\]

\[
 f_{\text{efficiency}} = \frac{N - N_{\text{overlap}}}{N} \tag{2-8}
\]

\[
 F3 = \max(f_{\text{efficiency}}). \tag{2-9}
\]

where \( N \) is the total number of data instances, \( N_{\text{overlap}} \) is the number of data instances in the overlap region. So, \( (N - N_{\text{overlap}}) \) are those data instances that contribute to feature
efficiency for a single feature. The maximum value of feature efficiency across all
dimensions is defined as feature efficiency \( F_3 \).

2.4.6 Measures of Inseparability of Classes ((L1, L2), (N1, N2))

Friedman and Rafsky [18] proposed a test to check if two data instances are from
the same distribution or different distributions. This test relies on the construction of a
minimum spanning tree that connects all the points to their nearest neighbors. We can
define N1 and N2 as follows:

\[
N_1 = \frac{N_{\text{boundary}}}{N}.
\]

where \( N \) is the total number of data points, \( N_{\text{boundary}} \) is the number of points connected to
the points lying next to the class boundary.

\[
N_2 = \frac{(D_{\text{avg}})_{\text{IntraClass}}}{(D_{\text{avg}})_{\text{InterClass}}}.
\]

To compute, \((D_{\text{avg}})_{\text{IntraClass}}\) is the average of all the distances from each point to
intra-class nearest neighbors and \((D_{\text{avg}})_{\text{InterClass}}\) is the average of all the distances from each
point to the inter class nearest neighbors.

2.4.7 Non-linearity (L3, N4)

Hoekstra and Duin [19] proposed a measure for the nonlinearity of a classifier for a given
dataset. For a training set, at first, we trained any linear classifier and then we create a test
set by linear interpolation with random coefficients between randomly drawn pairs of
points from the same class. We then define L3 as the error rate of the linear classifier on
the test set and N4 as the error rate of the nearest neighbor classifier on the test set.
2.4.8 **Single Classifier Vs. Ensemble Classifier**

Classification or predictive modeling is the task of approximating a mapping function ($f$) from input variables ($X$) to discrete output variables ($y$). To design a single classifier, different approaches can be pursued to achieve different goals. Tree-based methods recursively partition a data based on predefined parameters and stopping criteria to produce a set of rules. Statistical methods, for example, logistic regression estimates ($y|X$) directly, whereas discriminant analysis estimates class-conditional probabilities ($X|y$) which is converted into posterior probabilities using Bayes rule. Bagging, Random Forest, Fuzzy Forest are widely used ensemble classifier. The basic structure of an ensemble classifier is shown in Figure 2-2.

![Figure 2-2: Single vs. ensemble classifier](image)

On the contrary, ensemble classifiers use multiple base models to make final predictions. Traditionally ensemble classifiers are homogeneous, which means base models are developed using the same classification algorithm. Both theoretical and
empirical evidence demonstrated strong evidence that the combination of models increases predictive accuracy.

2.4.9 Non-Fuzzy vs. Fuzzy Classifier

Traditional classification models are crisp or hard classification where class membership is binary, which means a data instance can belong to only a class with class membership value equal to 1 or 0 for all other classes. In contrast, in fuzzy classification, a data instance can have membership in many different classes to different degrees such that the sum of membership values for that instance is equal to 1. Fuzzy classes are suitable for continuous data where class boundaries are unclear and overlapped than crisp or hard classification.

Fuzzy classification is very effective to deal with two types of problems present in data, 1) feature or attribute ambiguity, and 2) spatial vagueness, for example, remotely sensed data from aerial photography imposes attribute ambiguity problem, whereas spatial vagueness emerges due to inadequate sampling resolution to define clear boundary locations. In both cases, Fuzzy classifiers demonstrated their effectiveness in classifying data with the spatial and attribute uncertainty more accurately than crisp or hard classification.

2.4.10 Fuzzy Forests

In contrast to the random forest ensemble classifier, in this work we investigated the role of fuzzy forests ensemble classifier [20]. The fuzzy forest classifier used in this work is inspired based on the random forest classifier with the intent of reducing biases caused by the presence of correlated features. Figure 2-3(a) shows the flow diagram for Fuzzy Forest classifier we used for our experiment.
Proposed by Conn [20], fuzzy forest algorithm which reduces the feature space using a two-step iterative process, the screening step as shown in Figure 2-3(a), and the selection step as shown in Figure 2-3(b).

In screening step, unimportant features which are already assigned into partitions are removed in piecewise recursive manner. Inputs of the screening step are partitions of correlated features where the correlation within each partition is maximum but the correlation across the partitions is minimum. We consider the partitioning of the features by the set, \( P = \{ P_1, P_2, \ldots, P_m \} \) such that \( \sum P_i = P \) using the Weighted Correlation Network Analysis (WGCNA) algorithm [21] as shown in Figure 2-3(c).

**Figure 2-3:** Flow diagram for fuzzy forest classifier. (a) screening step (b) selection step (c) final step.

WGCNA is widely used to construct the correlation module or network using correlation as follows:
\[
    r = \frac{n \left( \sum_{i} xy \right) - \left( \sum_{i} x \sum_{i} y \right)}{\sqrt{\left[ n \sum_{i} x^2 - \left( \sum_{i} x \right)^2 \right] \left[ n \sum_{i} y^2 - \left( \sum_{i} y \right)^2 \right]}}. 
\]

Eq. 2-11

where \( x \) is a feature, and \( y \) is the associated class labels. After partitioning the feature space, the Recursive Feature Elimination - Random Forest (RFE-RF) algorithm [20] is used on each partition to remove less important features (as shown in Figure 2-3(a)). RFE-RF applies the Variable Importance Measure (VIM) to evaluate the importance of each feature within each partition. RFE-RF calculates VIM for \( n^{th} \) feature from \( k^{th} \) tree using the following relation:

\[
    VIM_k(n) = \frac{\sum_{i \in B_k} (Y_i - \tilde{f}_k(\tilde{X}_i))^2 - (Y_i - \hat{f}_k(X_i))^2}{|B_k|}. 
\]

Eq. 2-12

RFE-RF then combines VIM values for \( n^{th} \) feature from all trees in an entire random forest to calculate the final VIM value of \( n^{th} \) feature

\[
    VIM(n) = \frac{\sum_{k=1}^{ntree} VIM_k(n)}{ntree}. 
\]

Eq. 2-13

where \( Y_i \) is the target class for the \( i^{th} \) instance \( X_i \), \( \tilde{X}_i \) is the \( i^{th} \) instances in out of bag samples of the \( k^{th} \) tree. Similarly, \( \tilde{f}_k(X_i) \) is the conditional mean, and \( \hat{f}_k(\tilde{X}) \) is the conditional mean \( E[Y_i / \tilde{X}_i] \), and \( B_k \) is the indices for the out of bag samples from the \( k^{th} \) tree.

Starting with all features in each partition \( P_i \), RFE-RF is applied, and least important features produced by \( VIM \) function are then removed. We name the new partition with reduced features as \( P_i^{(1)} \). A second random forest is then applied on partition \( P_i^{(1)} \). The process of removing the features is continued until the predefined stopping criteria is
obtained. The selection step as shown in Figure 2-3(b) uses RFE-RF to allow for interaction among partitions. This RFE-RF is then applied to all the features from all the partitions that have been selected at screening steps to achieve the final set of reduced features and to build the final model using random forest classifier as shown in Figure 2-3(c).

The fuzzy forest algorithm uses parameters such as the Drop_fraction, keep_fraction, min_ntree, final_ntree, and module_number. Module_number refers to the number of modules or partitions created by WGCNA algorithm. For each partition, RFE-RF drops features according to drop_fraction in each step, whereas keep_fraction acts as stopping criteria. Both drop_fraction and keep_fraction lie between 0 and 1. Parameter min_ntree refers to the number of trees grown in each random forest by RFE-RF algorithm in screening step and final_ntree denotes to the number of trees grown in the final random forest after selection step.

The Fuzzy Forest algorithm aims at reducing dependency among features as much as possible while preserving maximum feature strength. Features are strongly dependent on each other when they are highly correlated. High feature correlation may produce bias to the modelling process. Since the WGCNA [22] algorithm creates partitions among features using the principle of minimizing correlation among partitions thereby alleviating biases between trees of the forest. Moreover, RFE-RF acts independently on each partition to eliminate unimportant features using VIM. During this elimination process, unimportant but highly correlated features within partitions are removed. Hence, features of the final partition produced by Fuzzy Forest turns out to be less dependent as well as more relevant.
We believe that this is the advantage of the Fuzzy Forest algorithm while dealing with dependent features.

The average runtime complexity of WGCNA algorithm is $\Theta(l^2 n)$ [22] for a single partition. The runtime complexity to build one unpruned decision tree is $O(mn \log(n))$ [21]. Since algorithm RFE-RF uses random forest for features elimination in recursive fashion, the average complexity of RFE-RF algorithm for $p$ number of partitions is $\Theta(p st m n \log(n))$. Complexity of final random forest algorithm is $\Theta(t f m n \log(n))$.

Hence, the average run time complexity of fuzzy forest algorithm is as follows:

$$\Theta(pl^2 n + pst mn \log(n) + tf mn \log(n)). \quad \text{Eq. 2-14}$$

$$\Rightarrow \Theta(p \left[ \frac{m^2}{p} \right] n + tm n \log(n)(ps + f)). \quad \text{Eq. 2-15}$$

$$\Rightarrow \Theta\left(\frac{1}{p} m^2 n + tm n \log(n)(ps + f)\right). \quad \text{Eq. 2-16}$$

$$\Rightarrow \Theta(m^2 n + tm n \log(n)). \quad \text{Eq. 2-17}$$

$$\Rightarrow \Theta(n(m^2 + mt \log(n))). \quad \text{Eq. 2-18}$$

To analyze the relationship between runtime and the number of instances $e$, we consider $m$ and $t$ constant. Hence, the average run complexity of Fuzzy Forest algorithm with respect to the number of instances is as follows:

$$\Rightarrow \Theta(n \log(n)). \quad \text{Eq. 2-19}$$

where $n$ is the total number of instances, $m$ is the total number of features, $p$ is the number of partitions created by WGCNA algorithm, $l$ is the average length of partition, $t$ is the number of trees in random forest created by the RFE-RF algorithm, $s$ is number of times RFE-RF runs for each partition, and $f$ is the fraction of features in the fuzzy forest classifier.
We compared the performance of our framework against k-Nearest Neighbor (k-NN) [23, 24], Fuzzy-Rough Nearest Neighbor (FRNN) [25], and Random Forest [26]. This section provides description of the non-linear classification approaches used.

2.5 Results

The following section provides an overview of the results obtained in establishing the relevance of the features extracted. We describe our results obtained using feature selection, and tests carried out to estimate the relevance of selected feature using data complexity analysis. Our objective is to utilize the insights obtained from data complexity analysis to dictate the choice of classification models and interpret the results obtained.

2.5.1 Feature Selection

Prior to applying feature selection, we filtered out 14 features that exhibited a minimum of zero variance and a maximum variance that exceeded a threshold of 0.99. To the remaining 796 features, we performed feature ranking using Relief-F followed by incremental feature selection using k-NN classifier. A subset of 39 features among 796 features was chosen for further analysis based on the reported highest accuracy, sensitivity, and specificity, respectively. It should be noted that the estimates of accuracy, sensitivity, and specificity was obtained using the instance-based k-NN classifier (refer to refer to Figure 2-4). Furthermore, we report an accuracy of 69.50%, sensitivity of 69.69%, and specificity or 69.33% using 39 features.
2.5.2 Linear Inseparability of Data

To establish and understand the characteristics of the data, we applied principle component analysis (PCA) to our data both pre and post feature selection process. Our data set prior to feature selection consisted of 796 original features. Similarly, our data set post feature selection consisted of 39 of the most significant features. We plotted our dataset using the two most prominent principal components for both pre and post processed datasets (refer Figure 2-5, Figure 2-6).
Figure 2-5: PC1 Vs. PC2 of 796 features prior to relief feature selection

Figure 2-6: PC1 Vs. PC2 of 39 features post relief feature selection
From Figure 2-5 and Figure 2-6, we observed that the feature selection using Relief-F indicate no reduction of overlap of the target classes of ovarian cancer dataset.

2.5.3 Data Complexity Analysis

To support the notion of class inseparability observed as shown in Figure 2-5 and Figure 2-6, we further quantified the complexity of the data by performing the data complexity analysis on both datasets pre and post feature selection using Relief-F. Our observations are tabulated in Table 2-2 and Table 2-2.

Table 2-1: Data complexity analysis pre-relief feature selection

<table>
<thead>
<tr>
<th>No of Fold</th>
<th>F1</th>
<th>F3</th>
<th>N1</th>
<th>N2</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>N4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0762</td>
<td>0.0403</td>
<td>0.5261</td>
<td>0.8885</td>
<td>0.8657</td>
<td>0.4028</td>
<td>0.3570</td>
<td>0.2360</td>
</tr>
<tr>
<td>2</td>
<td>0.0710</td>
<td>0.0427</td>
<td>0.5142</td>
<td>0.8845</td>
<td>0.8419</td>
<td>0.3768</td>
<td>0.3340</td>
<td>0.2330</td>
</tr>
<tr>
<td>3</td>
<td>0.0732</td>
<td>0.0427</td>
<td>0.5427</td>
<td>0.8945</td>
<td>0.8645</td>
<td>0.4052</td>
<td>0.3445</td>
<td>0.2185</td>
</tr>
<tr>
<td>4</td>
<td>0.0841</td>
<td>0.0403</td>
<td>0.5427</td>
<td>0.8897</td>
<td>0.8518</td>
<td>0.3957</td>
<td>0.3665</td>
<td>0.2180</td>
</tr>
<tr>
<td>5</td>
<td>0.0695</td>
<td>0.0403</td>
<td>0.5261</td>
<td>0.8860</td>
<td>0.8533</td>
<td>0.4076</td>
<td>0.3540</td>
<td>0.2350</td>
</tr>
<tr>
<td>6</td>
<td>0.0686</td>
<td>0.0427</td>
<td>0.5190</td>
<td>0.8816</td>
<td>0.8581</td>
<td>0.3815</td>
<td>0.3440</td>
<td>0.2350</td>
</tr>
<tr>
<td>7</td>
<td>0.0604</td>
<td>0.0450</td>
<td>0.5261</td>
<td>0.8859</td>
<td>0.8693</td>
<td>0.3957</td>
<td>0.3525</td>
<td>0.2150</td>
</tr>
<tr>
<td>8</td>
<td>0.0682</td>
<td>0.0332</td>
<td>0.5237</td>
<td>0.8756</td>
<td>0.8556</td>
<td>0.3957</td>
<td>0.3640</td>
<td>0.2060</td>
</tr>
<tr>
<td>9</td>
<td>0.0647</td>
<td>0.0403</td>
<td>0.5474</td>
<td>0.8876</td>
<td>0.8669</td>
<td>0.4028</td>
<td>0.3735</td>
<td>0.2420</td>
</tr>
<tr>
<td>10</td>
<td>0.0814</td>
<td>0.0426</td>
<td>0.5508</td>
<td>0.8923</td>
<td>0.8423</td>
<td>0.4043</td>
<td>0.3450</td>
<td>0.2425</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.0717</strong></td>
<td><strong>0.0410</strong></td>
<td><strong>0.5319</strong></td>
<td><strong>0.8866</strong></td>
<td><strong>0.8569</strong></td>
<td><strong>0.3968</strong></td>
<td><strong>0.3535</strong></td>
<td><strong>0.2281</strong></td>
</tr>
</tbody>
</table>

Table 2-2: Data complexity analysis post-relief feature selection.

<table>
<thead>
<tr>
<th>No of fold</th>
<th>F1</th>
<th>F3</th>
<th>N1</th>
<th>N2</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>N4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0814</td>
<td>0.0332</td>
<td>0.5569</td>
<td>0.9151</td>
<td>0.7649</td>
<td>0.3246</td>
<td>0.2720</td>
<td>0.1945</td>
</tr>
<tr>
<td>2</td>
<td>0.0946</td>
<td>0.0000</td>
<td>0.5664</td>
<td>0.9123</td>
<td>0.7454</td>
<td>0.2749</td>
<td>0.2395</td>
<td>0.1265</td>
</tr>
<tr>
<td>3</td>
<td>0.1095</td>
<td>0.0355</td>
<td>0.5190</td>
<td>0.9370</td>
<td>0.7684</td>
<td>0.3175</td>
<td>0.2540</td>
<td>0.1810</td>
</tr>
<tr>
<td>4</td>
<td>0.1173</td>
<td>0.0332</td>
<td>0.4834</td>
<td>0.9037</td>
<td>0.7348</td>
<td>0.2867</td>
<td>0.2440</td>
<td>0.1990</td>
</tr>
<tr>
<td>5</td>
<td>0.0945</td>
<td>0.0332</td>
<td>0.5450</td>
<td>0.9322</td>
<td>0.7254</td>
<td>0.2725</td>
<td>0.2585</td>
<td>0.1540</td>
</tr>
<tr>
<td>6</td>
<td>0.0814</td>
<td>0.0355</td>
<td>0.5592</td>
<td>0.9257</td>
<td>0.7631</td>
<td>0.3152</td>
<td>0.2500</td>
<td>0.1605</td>
</tr>
<tr>
<td>7</td>
<td>0.1046</td>
<td>0.0355</td>
<td>0.5237</td>
<td>0.9020</td>
<td>0.7439</td>
<td>0.2891</td>
<td>0.2525</td>
<td>0.1360</td>
</tr>
<tr>
<td>8</td>
<td>0.0797</td>
<td>0.0190</td>
<td>0.4929</td>
<td>0.9144</td>
<td>0.7393</td>
<td>0.2891</td>
<td>0.2240</td>
<td>0.1385</td>
</tr>
<tr>
<td>9</td>
<td>0.0747</td>
<td>0.0355</td>
<td>0.5308</td>
<td>0.9205</td>
<td>0.7262</td>
<td>0.2678</td>
<td>0.2535</td>
<td>0.1385</td>
</tr>
<tr>
<td>10</td>
<td>0.0916</td>
<td>0.0355</td>
<td>0.4634</td>
<td>0.9065</td>
<td>0.7387</td>
<td>0.2766</td>
<td>0.2175</td>
<td>0.1060</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.0942</strong></td>
<td><strong>0.0292</strong></td>
<td><strong>0.5204</strong></td>
<td><strong>0.9171</strong></td>
<td><strong>0.7428</strong></td>
<td><strong>0.2877</strong></td>
<td><strong>0.2437</strong></td>
<td><strong>0.1489</strong></td>
</tr>
</tbody>
</table>
The comparison in measure of eight data complexity descriptors are shown in Figure 2-7.

Figure 2-7: (a-h) Comparison of measures of eight data complexity descriptors pre and post feature selection process.
From the comparative analysis, we find that the Fisher discrimination ratio and individual feature efficiency remain unaltered in both the datasets (pre and post feature selection). Similarly, both (L1, L2) and (L3, N4) decrease and N2 increase after feature selection. Hence, through feature selection our dataset exhibits a higher linear separability between target classes. However, the non-linearity of classes has decreased.

While evaluating the goodness of features using complexity measurement criteria, it is desired that the measures of F1, F3, and N2 exhibit high values, implying a high discriminatory potential. Similarly, measures of L1, L2, L3, and N4 exhibit lower values implying lower degrees of class overlap. In our analysis, we observed that Fisher's Discriminant Ratio (F1) is considerably low for both datasets. Similarly, the linear inseparability analysis ((L1, L2), (N1, N2)) were relatively high. These observations indicate that the dataset post feature selection suffers from significant overlap and the target classes are inseparable despite the application of Relief-F feature selection.

Furthermore, we believe that classification using crisp rules or stringent hyper planes would not help in boosting accuracy. Therefore, we believe that a fuzzy based classifier would perform better. This is reinforced considering the degree of linear inseparability, our model choice would best suit an ensemble-based approach to enhance our sensitivity and specificity rates of classification.

2.5.4 Validation using Crisp and Fuzzy Classifiers

In this section, we adopted the 10-fold cross validation strategy to estimate model performance. We compared our performance using two broad classification approaches, namely, non-fuzzy classifiers and fuzzy classifier. Our non-fuzzy classifiers include the non-linear k-NN and non-linear ensemble Random Forest. Similarly, we compared the
results obtained with fuzzy classifiers FRNN and ensemble Fuzzy Forest. Among these chosen four classifiers, k-NN and FRNN are deterministic classifiers, whereas Random Forest and Fuzzy Forest are non-deterministic classifiers.

To benchmark our performance estimation, we generated models using all 796 features (refer Table 2-3) prior to using the Relief-F feature selection. It was observed that the fuzzy classifiers (FRNN and Fuzzy Forest) performed better than non-fuzzy classifiers (k-NN and Random Forest) with the Fuzzy Forest recording a maximum accuracy of 66%.

Table 2-3: Accuracy, sensitivity, and specificity obtained using 10-fold cross-validation.

<table>
<thead>
<tr>
<th>Type</th>
<th>Model</th>
<th>Benchmark Accuracy Using 796 Features (%)</th>
<th>Accuracy Using 39 Features (%)</th>
<th>Sensitivity Using 39 Features (%)</th>
<th>Specificity Using 39 Features (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Fuzzy</td>
<td>k-NN (k = 5)</td>
<td>63.35</td>
<td>67.16</td>
<td>68.39</td>
<td>65.97</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>63.10</td>
<td>63.75</td>
<td>68.39</td>
<td>67.65</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>FRNN</td>
<td>65.25</td>
<td>71.86</td>
<td>66.23</td>
<td>77.31</td>
</tr>
<tr>
<td></td>
<td>Fuzzy Forest</td>
<td>65.52</td>
<td>80.60</td>
<td>81.40</td>
<td>76.30</td>
</tr>
</tbody>
</table>

We then tested the performance of the classifiers using the selected 39 features. The observed accuracy, sensitivity, and specificity are reported in Table 2-3. In comparison with the benchmark results, it was observed that there is a negligible boost in overall accuracies using non-fuzzy classifiers. For example, the k-NN classifier observed a boost of 3.81% and Random Forest classifier observed a boost of 0.65%. On the contrary, the Fuzzy classifier experienced a significant boost – FRNN observed a boost of 6.61% and Fuzzy Forest observed a boost of 14.98%, respectively.

For Fuzzy Forest classifier, we set drop_fraction equal to 0.2, keep_fraction equal to 0.5, min_ntree equal to 500, final_ntree equal to 2000, module_number equal to 5. Figure 2-8 shows the results of screening steps of Fuzzy Forest algorithm. Each bar refers to one partition created by WGCNA algorithm. Each bar is divided into two parts. The part
of each bar colored in gray refers to unimportant features, whereas the part of each bar colored in red refers to important features in each partition, which had been selected by VIM measures in RFE-RF algorithm.

![Module Membership Distribution](image)

**Figure 2-8:** Results of module/partition membership distribution

**Figure 2-9** depicts that the Fuzzy Forest ensemble classification model obtained the highest reported sensitivity of 81.40% and specificity of 76.30% when compared to the other classifiers, respectively.

![Comparison of performance of non-fuzzy and fuzzy classifiers](image)

**Figure 2-9:** Comparison of performance of non-fuzzy and fuzzy classifiers
Furthermore, as shown in Figure 2-10, the ROC curve is used to illustrate the overall ability of our model to classify the tumor into a target class. The ROC curve in Figure 2-10 displays the True Positive Rate represented as the Sensitivity and False Positive Rate as the Specificity for the predicted suspicious and non-suspicious tumor class. An optimal threshold value of 0.5 results in a Sensitivity value of 81.40% and Specificity value of 76.30%. The Area Under the Curve (AUC) is estimated at 80.60% given a 95% confidence interval and a p-value of $2.33 \times 10^{-11}$.

![Figure 2-10: ROC curve (sensitivity vs specificity) for fuzzy forest](image)

2.5.5 Results of Runtime Complexity Analysis

To investigate the relationship between the number of instances and runtime, we created separate datasets using 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, and 469 instances using balanced sampling. Balanced sampling is used to retain the ratio of suspicious and non-suspicious instances to be consistent across every dataset. In these datasets, we maintain a constant feature set (39 features) and constant number of trees in RFE-RF across all datasets. Since the number of features and number of trees in each
random forest of RFE-RF are the same for all datasets, we observed that the runtime with respect to the number of records of Fuzzy Forest algorithm is logarithmic. Fuzzy Forest algorithm was then applied to each dataset and the runtime was computed and reported as shown in Table 2-4.

**Table 2-4: Actual runtime with respect to the number of instances**

<table>
<thead>
<tr>
<th>No of records (N)</th>
<th>Experimental Runtime Analysis</th>
<th>Theoretical Runtime Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Run time t (in sec)</td>
<td>t(normalized)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s(N) = N*log(N)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s(normalized)</td>
</tr>
<tr>
<td>200</td>
<td>3.15</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>460.21</td>
<td>0.58</td>
</tr>
<tr>
<td>225</td>
<td>4.27</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>529.24</td>
<td>0.67</td>
</tr>
<tr>
<td>250</td>
<td>4.96</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>599.49</td>
<td>0.76</td>
</tr>
<tr>
<td>275</td>
<td>5.8</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>670.82</td>
<td>0.85</td>
</tr>
<tr>
<td>300</td>
<td>6.38</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>743.14</td>
<td>0.94</td>
</tr>
<tr>
<td>325</td>
<td>6.62</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>816.36</td>
<td>1.00</td>
</tr>
<tr>
<td>350</td>
<td>7.15</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>890.42</td>
<td>1.12</td>
</tr>
<tr>
<td>375</td>
<td>7.75</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>965.26</td>
<td>1.22</td>
</tr>
<tr>
<td>400</td>
<td>8.12</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td>1040.82</td>
<td>1.31</td>
</tr>
<tr>
<td>425</td>
<td>9.14</td>
<td>1.18</td>
</tr>
<tr>
<td></td>
<td>1117.07</td>
<td>1.41</td>
</tr>
<tr>
<td>450</td>
<td>10.04</td>
<td>1.30</td>
</tr>
<tr>
<td></td>
<td>1193.95</td>
<td>1.51</td>
</tr>
<tr>
<td>469</td>
<td>10.9</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>1252.78</td>
<td>1.58</td>
</tr>
</tbody>
</table>

As the number of instances of the above datasets varied from 200 to 469 and we increased the size of each dataset by adding 25 instances each time, the logarithmic runtime is very close to the linear runtime as shown in Figure 2-11.

**Figure 2-11:** Comparison between actual versus theoretical runtime with respect to number of instances.
Table 2-5 provides a comparative analysis of the proposed approach with known state of the art CAD techniques for ovarian cancer classification.

Table 2-5: Summary of the state-of-the-art CAD techniques for ovarian classification

<table>
<thead>
<tr>
<th>Authors</th>
<th>No. of Subjects</th>
<th>Method</th>
<th>Classifier</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tang [27]</td>
<td>Normal: 95</td>
<td>Four statistical moments</td>
<td>Kernel partial least square classifier</td>
<td>Acc: 99.35%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 121</td>
<td></td>
<td></td>
<td>Sen: 99.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Spe: 99.16%</td>
</tr>
<tr>
<td>Petricoin [28]</td>
<td>Benign: 66</td>
<td>Proteomic spectra</td>
<td>Genetic algorithm + self-organizing cluster analysis</td>
<td>Sen: 100%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 50</td>
<td></td>
<td></td>
<td>Spe: 95%</td>
</tr>
<tr>
<td>Biagiotti [8]</td>
<td>Benign: 175</td>
<td>Age and parameters from TVUS images</td>
<td>Three-layer back propagation network</td>
<td>Sen: 96%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tailor [9]</td>
<td>Benign: 52</td>
<td>Clinical and ultrasound based variables from TVUS images</td>
<td>Back propagation neural network</td>
<td>Sen: 100%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 15</td>
<td></td>
<td></td>
<td>Spe: 98.1%</td>
</tr>
<tr>
<td>Lucidarme [10]</td>
<td>Benign: 234</td>
<td>Backscattered ultrasound (3D TVUS)</td>
<td>Ovarian HistoScanning (OHS) system</td>
<td>Sen: 98%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 141</td>
<td></td>
<td></td>
<td>Spe: 88%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acc: 91.73%</td>
</tr>
<tr>
<td>Acharya [12]</td>
<td>Benign: 10</td>
<td>Local Binary Pattern + Law’s Texture Energy</td>
<td>SVM</td>
<td>Sen: 100%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 10</td>
<td></td>
<td></td>
<td>Spe: 99.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acc: 99.9%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 10</td>
<td></td>
<td></td>
<td>Spe: 99.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acc: 99.8%</td>
</tr>
<tr>
<td>Acharya [14]</td>
<td>Benign: 10</td>
<td>Texture + higher order spectral features</td>
<td>Decision Tree</td>
<td>Sen: 94.3%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 10</td>
<td></td>
<td></td>
<td>Spe: 99.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acc: 97.0%</td>
</tr>
<tr>
<td>Acharya [6]</td>
<td>Benign: 10</td>
<td>First order statistics + GLCM + run length matrix</td>
<td>k-NN/PNN</td>
<td>Sen: 100%</td>
</tr>
<tr>
<td></td>
<td>Malignant: 10</td>
<td></td>
<td></td>
<td>Spe: 100%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acc: 100%</td>
</tr>
<tr>
<td><strong>Our Method</strong></td>
<td><strong>Non-suspicious: 238</strong></td>
<td><strong>Radon Transform + Non-linear Feature extraction</strong></td>
<td><strong>Fuzzy Forest</strong></td>
<td><strong>Sen: 81.40%</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Suspicious: 231</strong></td>
<td></td>
<td></td>
<td><strong>Spe: 76.30%</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Acc: 80.60%</strong></td>
</tr>
</tbody>
</table>
In our experiment, we have used a dataset of 469 instances (non-suspicious: 238, suspicious: 231) which is much larger compared to the datasets mentioned in Table 2-5. Moreover, the complexity of these dataset is unknown. Both principal component analysis and data complexity analysis support the notion that the dataset we used to conduct our experiment was complex. Furthermore, the target classes in the dataset were overlapped. Under this scenario, boosting of model performance was a computational challenge.

The proposed approach and observations justify the use of the Fuzzy Forest classifier as an effective technique in handling inherent class overlap that is common in most real-world dataset.

2.6 Findings and Discussion

In this chapter, we focused on reducing the inherent complexities posed by features by using Relief-based feature space modeling and building an appropriate Fuzzy classification to classify ovarian cancer data effectively. We report a highest classification accuracy of 80.6 ± 0.5% accuracy, 81.40% sensitivity, 76.30% specificity, respectively.

In the next chapter, we explored feature space transformation using PCA as a technique of feature space modeling to reduce the dimensionality of health sensor data collected in a streaming environment which subsequently used by Direct Density Ratio Estimation method to detect change points with better accuracy and efficiency.
CHAPTER 3

FEATURE SPACE TRANSFORMATION FOR CHANGE POINTS DETECTION FRAMEWORK

3.1 Chapter Overview

In this chapter, we explored feature space transformation using PCA as a technique of feature space modeling to reduce the dimension of health sensor data collected in a streaming environment. Next, we modeled the direct ratio of probability distributions of two data segments and explored this statistical measure to identify the change points with better accuracy and efficiency. We propose an unsupervised framework combining Principle Component Analysis (PCA) with Pearson divergence measured by a density ratio estimation method known as unconstrained least squares importance fitting to measure the divergence between data distributions of two retrospective multidimensional data segments embracing the concept that significant changes in data distribution lead to higher divergence, which may eventually indicate the occurrence of potential change points. To address the issue of higher dimensionality, we reduced the original feature space into lower dimensional subspace by affine transformation using PCA based on the hypothesis that Pearson divergence between PCA transformed data segments can detect change points with equal performance and less computational time compared to the performance from original data segments with full features. For PCA transformation, we used the matrix factorization method known as Singular Value Decomposition (SVD) instead of the traditional Eigen
Value Decomposition (EVD) approach since SVD is more stable and computationally less expensive compared to EVD due to performing low rank matrix approximation on data matrix. At first, we compared proposed framework with another density ratio estimation method known as Kullback–Leibler importance estimation procedure using artificial datasets and later, we validated our proposed framework with three baseline methods using real world datasets. In both setups, we observed better and more consistent performance by proposed framework in terms of higher change detection rate and reduced computational time which demonstrate its applicability and effectiveness.

3.2 Related Works

In this modern era, a wide range of applications, for example, fraud detection in financial systems [29], defect analysis in product line [30-32], intrusion or outlier detection in cyber systems [33-38], evolution of heterogeneous information in social media [38, 39], cloud-based health monitoring service [40, 41], and many more, depend on streaming data collection and analysis which can be subjected to frequent change of underlying data distribution. The problem of time points discovery when the data distributions over time span undergo significant changes is defined as change point detection, drift detection, and data evolution [42-44]. However, both streaming data collection and analysis in real time can be very challenging; hence detection of change points may experience unwanted delay. Based on this delay, there exists two categories of change-point detection methods known as real-time detection [45] and retrospective detection [46], respectively.

The major focus of real-time change-point detection is to respond immediately whenever a change occurs, whereas retrospective change-point detection emphasizes on accurate detection within predefined time delay. Applications like climate change or
intrusion detection may permit delays to a certain level, hence, fall in retrospective detection category. On the contrary, applications like control mechanism of robotics require instant detection and immediate response, hence opt for real time detection. Apart from delay criteria, change detection methods can be divided into two categories based on change measurement criteria. The first approach is the model-driven approach which is particularly suitable for supervised learning in streaming environment [47].

In the model-driven approach, the performance, as example accuracy, class precision, recall, F1-Score etc. are continuously monitored. Potential change points are predicted when model performance fall below a certain level [43, 47]. The second approach is the data-driven approach which focuses on change of certain data properties with an aim to identify potential change points. The most widely used data-driven approach for change point detection is to measure the dissimilarity or divergence between reference time segment and current time segment by comparing their probability distribution. Change points are reported when dissimilarity measures are observed above a predefined threshold [44, 48, 49]. Subspace identification is another major data-driven approach to analyze changes in time series data [50]. Among data-driven approaches, density ration estimation has caught much attention in recent years to detect change points in time series data. The basic notion of this approach is to estimate the ratio of probability densities instead of estimating the probability densities separately. In [49], Sugiyama has successfully explored density ratio estimation to detect change points in single dimensional time series data.

In this chapter, change point analysis focuses on the data-driven retrospective approach for multi-dimensional time-series data based on applying PCA on data segments from reference and test time interval for dimensionality reduction followed by a density
ratio estimation algorithms known as unconstrained least squares importance fitting (uLSIF) [49, 51] to approximate Pearson divergence (PE-divergence) with an aim to investigate the following problem:

"Is PE-divergence between two PCA transformed data segments with reduced feature space capable to detect the change point between two original data segments with full feature space? What are the consequences of PCA transformation on change detection rate and runtime?"

For PCA transformation, we applied SVD [52] instead of EVD [53] to reduce the original feature space into a new affine subspace. SVD uses low rank matrix approximation to find the optimal number of orthogonal principle vectors which makes the approach finite, hence SVD is very stable and less computationally expensive. On other hand, EVD involves covariance matrix calculation of data matrix followed by solving the characteristic polynomial equation of that covariance matrix to find eigenvalues and associated eigenvectors. However, solving the characteristic polynomial equation of a covariance matrix is an iterative approach. When the size of the data samples and the dimension of feature space are relatively high, which is a common scenario in streaming data environment, covariance matrix calculation may incur high computing errors and convergence of solving characteristic polynomial equation may become very slow. This is the reason EVD may produce unstable results compared to SVD when the problem high feature dimensionality is involved with datasets.

To conduct our experiments, we used both artificial and real datasets. At first, we compared our proposed framework with another density ratio estimation method known as Kullback–Leibler Importance Estimation Procedure (KLIEP) [51] using artificial datasets
and observed overall better performance by uLSIF over KLIEP. Finally, we validated our proposed framework with three baseline methods known as Change Detection by Log Likelihood (CD-LLH), Change Detection by Maximum KL Divergence (CD-MKL), Change Detection by Intersection Area between two data distribution (CD-Area) presented in [54] using the same real-world public datasets, and the experimental results show that the proposed technique performs better and in a more consistent manner which assures the usefulness of our proposed framework.

The remainder of this chapter is arranged as follows. In Section 3.4, we described algorithm and detail methodologies for our proposed technique. In Section 3.5, we reported experimental results on artificial and real datasets using necessary tables and graphs along with analysis of computational complexity. Finally, we drew conclusions by summarizing our contribution in Section 3.6.

3.3 Dataset Information

3.3.1 Artificial Dataset: Type # 01

We created four datasets with sample size of 5000 each and dimensions 25, 50, 75 and 100 respectively with mean and covariance matrix as follows:

Mean: $\mu_1 = \mu_2 = \mu_3 = \ldots \ldots = \mu_n = 1 \ (n = 25, 50, 75, 100)$

Covariance matrix: 

$$
\begin{bmatrix}
\frac{1}{2} & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & \frac{1}{2}
\end{bmatrix}
$$

In a covariance matrix, a diagonal element $m_{ii}$ refers to the variances of the $i^{th}$ variable and an off-diagonal element, $m_{ij} (i \neq j)$ refers to the covariance between $i^{th}$ and $j^{th}$ variables. Since we used $n \times n$ diagonal matrix as covariance matrix, for any $i \neq j, m_{ij} = 0$ and $i = j, m_{ii} = 1/2$, which means all the variables generated by multi
variate random generator have a variance equal to 0.5 but not correlated to each other. We introduced a change point every 100 samples by adding gaussian noise with mean $\mu_N$ and variance $\sigma_N$ at time $t$ as follows:

Mean: $\mu_N = \begin{cases} 
0, & N=1 \\
\mu_{N-1} + \frac{N}{16}, & N=2,3,4,\ldots,49 
\end{cases}$

Variance: $\sigma_N = \begin{cases} 
1, & N=1,3,5,\ldots,49 \\
\ln(e+\frac{N}{4}), & N=2,4,6,\ldots,48 
\end{cases}$

We added Gaussian noise to the variables of the original datasets in sequential order, which means we kept the first 100 samples unchanged, then applied noise to the first feature of subsequent 100 samples while keeping other features unchanged. Next, we added noise to the second feature of the next 100 samples keeping other features unchanged, and so on. In this way, we inserted one change point every 100 samples and we can view original datasets as a combination of fifty data segments where a change point exists between every two data segments.

3.3.2 Artificial Dataset: Type # 02

We created four datasets with sample size of 5000 each and dimensions 25, 50, 75 and 100, respectively with the same mean and different covariance matrix for every 100 samples. By magnifying and shrinking variance in a periodic fashion as follows:

Initialize Variances: $V_{\text{cur}}, V_{\text{min}}, V_{\text{max}}, V_{\text{inc}}, V_{\text{dec}}$

Magnify Variances: $V_{\text{cur}} = V_{\text{cur}} + V_{\text{inc}}$ until $V_{\text{cur}} < V_{\text{max}}$

$V_{\text{inc}} = V_{\text{inc}} + 0.5$ when $V_{\text{cur}} = V_{\text{max}}$

Shrink Variances: $V_{\text{cur}} = V_{\text{cur}} - V_{\text{dec}}$ until $V_{\text{cur}} > V_{\text{min}}$

$V_{\text{dec}} = V_{\text{dec}} + 1$ when $V_{\text{cur}} = V_{\text{min}}$

3.3.3 Artificial Dataset: Type # 03

We created four datasets with a sample size of 5000 each and dimensions 25, 50, 75, and 100, respectively with changing mean and the same covariance matrix for every 100 samples. We magnify and shrink mean in a periodic fashion as follows:
Initialize mean: \( m_{\text{cur}}, m_{\text{min}}, m_{\text{max}}, m_{\text{inc}}, m_{\text{dec}} \)
Magnify mean: \( m_{\text{cur}} = m_{\text{cur}} + m_{\text{inc}} \text{ until } m_{\text{cur}} < m_{\text{max}} \)
\( m_{\text{inc}} = m_{\text{inc}} + 2.5 \text{ when } m_{\text{cur}} = m_{\text{max}} \)
Shrink mean: \( m_{\text{cur}} = m_{\text{cur}} - m_{\text{dec}} \text{ until } m_{\text{cur}} > m_{\text{min}} \)
\( m_{\text{dec}} = m_{\text{dec}} + 1 \text{ when } m_{\text{cur}} = m_{\text{min}} \)

3.3.4 Public Dataset

Forest Cover Type and PAMAP2 are two composite datasets, publicly available in UCI machine learning repository, which offer a collection of quality datasets. Forest Cover Type datasets contain information about cover type of 581,012 cells (each 30 x 30 meter) from RIS (Resource Information System) data and USGS (US Geological Survey) collected by USFS (US Forest Service). PAMAP2 Physical Activity Monitoring dataset contains 3,850,505 data samples of 18 different physical activities collected from three body sensors and one heart rate monitor performed by 9 subjects. PAMAP2 dataset collection was donated by the department of Augmented Visions, a common research group of DFKI and the University of Kaiserslautern, Germany. For our experiments, we picked seven datasets from these two composite datasets. Reference [54] shows brief information about these chosen seven datasets as follows:

**Table 3-1:** Brief information about real datasets available in UCI machine repository.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source</th>
<th># of Records</th>
<th># of Dimensions Chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spruce</td>
<td>Forest Cover Type</td>
<td>211840</td>
<td>10</td>
</tr>
<tr>
<td>Lodgepole Pine</td>
<td>Forest Cover Type</td>
<td>283301</td>
<td>10</td>
</tr>
<tr>
<td>Ascending Stairs</td>
<td>PAMAP2</td>
<td>117216</td>
<td>30</td>
</tr>
<tr>
<td>Cycling</td>
<td>PAMAP2</td>
<td>164600</td>
<td>30</td>
</tr>
<tr>
<td>Descending stairs</td>
<td>PAMAP2</td>
<td>104944</td>
<td>30</td>
</tr>
<tr>
<td>Ironing</td>
<td>PAMAP2</td>
<td>238690</td>
<td>30</td>
</tr>
<tr>
<td>Vacuum cleaning</td>
<td>PAMAP2</td>
<td>175353</td>
<td>30</td>
</tr>
</tbody>
</table>
3.4 Algorithm and Methodology

In this section, we described our proposed framework Change Detection in multi-dimensional Feature Space with PCA and uLSIF (MCD-PuLSIF) (Algorithm 1), several methods and techniques in detail. Our major contribution is to combine all these separate methods into a unifying framework so that we can detect change points in a multi-dimensional time series stream with an improved change point detection rate and reduced computational time.

3.4.1 Framework MCD-PuLSIF

Change Detection in multi-dimensional Feature Space with PCA and uLSIF are presented in Table 3-2.

Table 3-2: Change detection in multi-dimensional feature space with PCA and uLSIF

<table>
<thead>
<tr>
<th>ALGORITHM 1: Framework MCD-PuLSIF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1:</strong> Procedure MCD-PuLSIF</td>
</tr>
<tr>
<td>Parameters: Sliding window size l, sliding step size s, method name mn, cumulative variance cv</td>
</tr>
<tr>
<td>Input: streaming data $S_1 \leftarrow {x_1, x_2, \ldots, x_t, \ldots}$</td>
</tr>
<tr>
<td>Output: time $t_{cp}$ when detecting a change</td>
</tr>
<tr>
<td>%set current time equal to window size</td>
</tr>
<tr>
<td><strong>2:</strong> Initialize current time $t_{cur} \leftarrow l$</td>
</tr>
<tr>
<td><strong>3:</strong> while a new sample $x_t$ arrives in the stream do</td>
</tr>
<tr>
<td><strong>4:</strong> Set sliding window $W_{cur} \leftarrow {x_{t'}, t \in [t_{cur} - 1 + 1: t_{cur}]}$</td>
</tr>
<tr>
<td><strong>5:</strong> Divide $W_{cur}$ into two data segments, reference data segment $DS_{ref}$ and test data segment $DS_{test}$</td>
</tr>
<tr>
<td><strong>6:</strong> Set $DS_{ref} \leftarrow {x_{t'}, t \in \left[t_{cur} - 1 + 1: \frac{t_{cur}}{2}\right]}$</td>
</tr>
<tr>
<td><strong>7:</strong> Set $DS_{test} \leftarrow {x_{t'}, t \in \left[\frac{t_{cur}}{2} + 1: t_{cur}\right]}$</td>
</tr>
<tr>
<td>%apply PCA on both data segments</td>
</tr>
<tr>
<td><strong>8:</strong> $PC_{ref} \leftarrow \text{CalculatePCA}(DS_{ref})$</td>
</tr>
<tr>
<td><strong>9:</strong> $PC_{test} \leftarrow \text{CalculatePCA}(DS_{test})$</td>
</tr>
<tr>
<td><strong>10:</strong> Set $n =$ number of first $n$ principle component which capture cumulative variance equal to $cv$</td>
</tr>
<tr>
<td><strong>11:</strong> $PC'<em>{ref} \leftarrow$ first $n$ components from $PC</em>{ref}$</td>
</tr>
<tr>
<td><strong>12:</strong> $PC'<em>{test} \leftarrow$ first $n$ componets from $PC</em>{test}$</td>
</tr>
<tr>
<td>% define forward divergence $f_r$ and backward divergence $f_b$</td>
</tr>
</tbody>
</table>
Table 3-2: Change detection in multi-dimensional feature space with PCA and uLSIF

**ALGORITHM 1**: Framework MCD-PuLSIF

13: \( f_r \leftarrow D(PC_{ref}' \parallel PC_{test}') \leftarrow \text{Calculate}(PC_{ref}', PC_{test}', mn) \)
14: \( f_b \leftarrow D(PC_{test}' \parallel PC_{ref}') \leftarrow \text{Calculate}(PC_{test}', PC_{ref}', mn) \)
15: \( f_{sym} \leftarrow f_r + f_b \)
   \% \( f_r \) is the cutoff range to detect change point
16: \( f_t \leftarrow \text{CalculateCutOffF}(f_{sym}, \text{listf}_{cp}) \)
17: if \( f_{sym} \in f_t \)
   Report a change point at time \( t_{cur} \) detected at \( t_{cur} \)
   Update listf_{cp} with \( f_{sym} \)
End if
18: end while
19: set current time \( t_{cur} \leftarrow t_{cur} + s \)
20: end procedure
21: Procedure Calculate\( f(X_{ref}, X_{test}, mn) \)
   \% \( mn \leftarrow \text{KLIEP or uLSIF} \)
22: find density ration estimator \( \hat{g}(X) \) using density ration estimation method
23: Estimate divergence \( f(X_{ref}\|X_{test}) \) using \( \hat{g}(X) \)
24: return \( f \)
25: end procedure
26: Procedure CalculateCutOffF\( (f_{cur}, \text{listf}_{cp}) \)
   \% listf_{cp} contains \( f \) values which detected change point at time \( t_{cp}, t_{cp+1}, t_{cp+2}, ..., t_{cp+n}, ... \)
   \% \( t_{cp+n} < t_{cur} \)
27: Calculate cumulative mean \( f_{mean} \) from listf_{cp}
28: Calculate standard deviation \( f_{std} \) from listf_{cp}
29: Calculate cutoff range \( f_{cutoff} \) such that \( f_{cutoff} \in [f_{mean} - f_{std} : f_{mean} + f_{std}] \)
30: return \( f_{cutoff} \)
31: end procedure
32: Procedure CalculatePCA\( (X) \)
   \% Right Singular Vectors from SVD factorization of centralized \( X \) represents the principle components
33: Centerize \( X \)
34: Apply SVD on \( X \)
35: \( u \leftarrow \) Right Singular Vectors of SVD factorization of \( X \) that captures 96% variance
36: return \( u \)
37: end procedure

The detail of SVD mentioned in Lines 34 - 36 are described in [52], and the pseudo-code of in density ratio estimation in line 21 for both KLIEP and uLSIF are available in [51].
3.4.2 Defining Temporal Intervals

According to [56], three types of temporal spans are defined to analyze the time series data collected from the streaming environment. Landmark window is the first type of temporal span where analysis is performed on the data segment between a specific time point called landmark and the current time point. Suppose \( t_{\text{landmark}} \) and \( t_{\text{current}} \) \((t_{\text{landmark}} > t_{\text{current}})\) to be the landmark time point and the current time point respectively. For Landmark Window, records only from the time interval defined by \([t_{\text{landmark}} : t_{\text{current}}]\) are considered. For the Sliding Window, the second kind of temporal span, primarily focuses on the length of the window size. Suppose \( l \) to be the size of the sliding window, for the sliding window only records from time interval defined by \([t_{\text{current}} - l + 1 : t_{\text{current}}]\) will be considered for analysis. The third approach to define temporal span is the damped window which considers recent data samples more important than the older samples.

In the damped window, the recent samples are given more weight than the older samples such that the older samples fade away from the damped window after a certain period when the weight falls below a certain level. Sliding window is known to be the most suitable windowing technique where streaming data analysis is concerned. In our research, we applied the sliding window technique on the time series data and divided each window into two data segments. We considered the first segment as the reference segment and the next segment as the test segment for further analysis. To conduct our experiments, we set the window size equal to \(1.5 \times 10^2\) and \(10^4\) for the synthetic and public datasets, respectively.
3.4.3 PCA (Principle Component Analysis)

We consider a dataset $X$ in matrix form with $n$ number of features where each row is referred to as $x_i$. Features in dataset $X$ may or may not be correlated. PCA [57] transforms these $n$ number of features into $p$ dimensional orthogonal space ($1 \leq p \leq n$) such that the transformed features are linearly uncorrelated. After transformation into the new feature space, the first principle component captures maximum variance, the second principle component, orthogonal to the first principle component, captures the second maximum variance. The third principle component which is orthogonal to the second one captures the third maximum variance. The subsequent principle components follow the same pattern as the preceding components. PCA is known to be sensitive to the normalization or scaling of the original features. In our experiment, we selected only those PCA components that captured 96% of cumulative variance of original data segments.

Principle components turn out to be the new basis vectors or axis of reduced subspace and original data is projected along these new orthogonal vectors. EVD is the traditional approach to find these principle axes of transformed subspace. In EVD, the covariance matrix $C = X^T X$ is calculated after centralizing dataset $X$. By solving the characteristic polynomial equation of $C$, we find the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and corresponding eigenvectors $v_1, v_2, ..., v_n$. The eigenvector $(v_{\text{max}})_1$ associated with the highest eigenvalue is the first principle component. Similarly, the eigenvector $(v_{\text{max}})_2$ associated with the second highest eigenvalue is the second principle component, and so on. Solving characteristic polynomial equation of $C$ is an iterative approach. When the number of data sample $n$ and the number of features $p$ are very high, covariance matrix calculation may experience computational error and the convergence of characteristic
polynomial solution becomes very slow which makes EVD less stable, and from the computation point of view, it is more expensive.

SVD factors the dataset $X$ as $X = UDV^T$ where $U$ is a $n \times r$ column-orthonormal matrix, $D$ is a diagonal $r \times r$ matrix with singular values $s_i$ in descending order, and $V$ is a $r \times p$ column-orthonormal matrix. Here, $r$ represents the rank of data matrix $X$. The singular values $s_i$ is the square root of eigenvalue of $\lambda_i$, and the right orthogonal vector $V$ of SVD is the same as eigenvector $V$ of EVD. Detail mathematical proof is available in [39]. Therefore, SVD is used as an alternative approach for PCA transformation. SVD uses row lank matrix approximation which makes, SVD finite. This is the reason SVD is more stable than EVD for larger $n$ and $p$.

Moreover, the time complexity of EVD is $O(n^2 p + p^2 n)$, whereas time complexity of SVD is $O(\min(n^2p, p^2n))$ for PCA. When $n$ and $p$ are very large, which is a common scenario in streaming environment, this small improvement in computational time may create significant impact on the overall performance. Since dealing change points in multi-dimensional data stream is our primary objective, we chose SVD for principle component over EVD.

3.4.4 Dissimilarity Measurement Based on Divergence

In general term, the dissimilarity measure between two data distribution is defined as f-divergence. In [49, 51], the statistical measure of f-divergence is formulated as follows:

$$f - \text{divergence} = D(P_{ref} || P_{test}). \quad \text{Eq. 3-1}$$

$$DD(P_{ref} || P_{test}) = \int P_{ref}(x)f\left(\frac{P_{ref}(x)}{P_{test}(x)}\right)dx. \quad \text{Eq. 3-2}$$
Here, $P_{\text{ref}}$ and $P_{\text{test}}$ are the probability distribution of data segments from reference time interval and test time interval, respectively, and $f$ in Eq. 3-1 is a generalized form of a convex function. Hence, $f(1) = 0$. Using f-divergence, we can formalize Kullback–Leibler divergence (KL-divergence) [58] between two data distribution as follows:

$$f = x \log(x).$$  \hspace{1cm} \text{Eq. 3-3}$$

$$KL(P_{\text{ref}} || P_{\text{test}}) = \int P_{\text{ref}}(x) \log \left( \frac{P_{\text{ref}}(x)}{P_{\text{test}}(x)} \right) dx.$$  \hspace{1cm} \text{Eq. 3-4}$$

Another widely used divergence measure called Pearson divergence (PE-divergence) [59] between two data distribution can be formulated as follows:

$$f = \frac{1}{2}(x - 1)^2.$$  \hspace{1cm} \text{Eq. 3-5}$$

$$PE(P_{\text{ref}} || P_{\text{test}}) = \int P_{\text{ref}}(x) \left( \frac{P_{\text{ref}}(x)}{P_{\text{test}}(x)} - 1 \right)^2 dx.$$  \hspace{1cm} \text{Eq. 3-6}$$

Since $\frac{P_{\text{ref}}(x)}{P_{\text{test}}(x)} \neq \frac{P_{\text{test}}(x)}{P_{\text{ref}}(x)}$, $D(P_{\text{ref}} || P_{\text{test}}) \neq D(P_{\text{test}} || P_{\text{ref}})$, is the reason f-divergence measure is asymmetric. According to [49], the performance of symmetric divergence is significantly better than asymmetric divergence; hence, we symmetrized both KL-divergence and PE-divergence as follows:

$$KL_{(\text{sym})} = KL(P_{\text{ref}} || P_{\text{test}}) + KL(P_{\text{test}} || P_{\text{ref}}).$$  \hspace{1cm} \text{Eq. 3-7}$$

$$PE_{(\text{sym})} = PE(P_{\text{ref}} || P_{\text{test}}) + PE(P_{\text{test}} || P_{\text{ref}}).$$  \hspace{1cm} \text{Eq. 3-8}$$

In reality, actual probability distributions, both $P_{\text{ref}}(x)$ and $P_{\text{test}}(x)$ from reference data segment and test data segment, respectively, are not known, and even estimation of probability distributions is known to be a hard problem [60]. Hence, in actual scenario, to compute KL-divergence directly from Eq. 3-7 or PE-divergence from Eq. 3-8 is very challenging. This problem is approached by approximating both KL-divergence and
PE-divergence instead of computing the actual divergences using a method called direct
density-ratio estimation. Instead of estimating $P_{\text{ref}}(x)$ and $P_{\text{test}}(x)$ separately, the
density-ratio estimation method focuses in learning the density-ratio function $\frac{P_{\text{ref}}(x)}{P_{\text{test}}(x)}$ which is significantly easier. In the following sections, we described two density ratio estimation methods, KLIEP and uLSIF from [49, 51].

3.4.5 Mathematical Formulation of KLIEP Method

KLIEP, one of the algorithms to estimate density-ratio described in [51] is well-suited to estimate KL-divergence between two data distributions. From Eq. 3-9, the formulation of actual KL-divergence with respect to actual density ratio $r(x)$ is as follows:

$$KL(r(x)) = \int P_{\text{ref}}(x) \log \left( \frac{P_{\text{ref}}(x)}{P_{\text{test}}(x)} \right) dx. \quad \text{Eq. 3-9}$$

$$r(x) = \frac{P_{\text{ref}}(x)}{P_{\text{test}}(x)}. \quad \text{Eq. 3-10}$$

Suppose $\hat{r}(x)$ is the estimated density ratio modelled from actual density ratio $r(x)$, then the $P_{\text{ref}}(x)$ can be estimated using $\hat{r}(x)$ as follows:

$$\hat{P}_{\text{ref}}(x) = \hat{r}(x) \cdot P_{\text{test}}(x). \quad \text{Eq. 3-11}$$

If we integrate any probability function, the result should be one as follows:

$$\int \hat{P}_{\text{ref}}(x) dx = \int \hat{r}(x) \cdot P_{\text{test}}(x) dx = 1. \quad \text{Eq. 3-12}$$

We consider another divergence, $KL_d$ between actual probability distribution $P_{\text{ref}}$ and estimated probability distribution $\hat{P}_{\text{ref}}(x)$ as follows:

$$KL_d = KL(P_{\text{ref}} \parallel \hat{P}_{\text{ref}}) = \int P_{\text{ref}}(x) \log \left( \frac{P_{\text{ref}}(x)}{\hat{P}_{\text{ref}}(x)} \right) dx. \quad \text{Eq. 3-13}$$

Now the approximation of KL-divergence between $P_{\text{ref}}$ and $P_{\text{test}}$ will be closer to the actual KL-divergence between $P_{\text{ref}}$ and $P_{\text{test}}$ when $KL_d$, the divergence between the
actual probability distribution $P_{ref}$ and estimated probability distribution $\hat{P}_{ref}(x)$ will be minimum. Combining Eq. 3-10 and Eq. 3-13, we have the following mathematical formulation:

$$KL(P_{ref} \parallel \hat{P}_{ref}) = \int P_{ref}(x) \log \left( \frac{P_{ref}(x)}{r(x) \ast p_{test}(x)} \right) dx.$$  \hspace{1cm} \text{Eq. 3-14}

$$KL(P_{ref} \parallel \hat{P}_{ref}) = \int P_{ref}(x) \log \left( \frac{P_{ref}(x)}{p_{test}(x)} \right) dx - \int P_{ref}(x) \log (r(x)) dx.$$  \hspace{1cm} \text{Eq. 3-15}

The second term of Eq. 3-13 does not depend on $r(x)$; therefore, it is a constant defined as $Const$.

$$KL(P_{ref} \parallel \hat{P}_{ref}) = Const - \int P_{ref}(x) \log (r(x)) dx.$$  \hspace{1cm} \text{Eq. 3-16}

$$KL_d(r(x)) = Const - \int P_{ref}(x) \log (r(x)) dx.$$  \hspace{1cm} \text{Eq. 3-17}

We define the 2$^\text{nd}$ part of equation (8) as follows:

$$KL_{2nd}(r(x)) = \int P_{ref}(x) \log (\hat{r}(x)) dx.$$  \hspace{1cm} \text{Eq. 3-18}

In [51], Sugiyama used two empirical approximations. The first approximation $KL_e(r(x))$ of $KL_d(r(x))$ from Eq. 3-13 is given as follows:

$$KL_e(r(x)) = \frac{1}{n_{ref}} \sum_{i=1}^{n_{ref}} \log \left( x_i^{n_{ref}} \right).$$  \hspace{1cm} \text{Eq. 3-19}

The second approximation from Eq. 3-12 is given as follows:

$$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \hat{r}(x_i^{n_{test}}) = 1.$$  \hspace{1cm} \text{Eq. 3-20}

The optimization problem was formulated using Eq. 3-15 - Eq. 3-20 as follows:
maximize \( \frac{1}{n_{\text{ref}}} \sum_{i=1}^{n_{\text{ref}}} \log \hat{r}(x_i^{n_{\text{ref}}}) \) such that \( \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \hat{r}(x_i^{n_{\text{test}}}) := 1. \) \( \text{Eq. 3-21} \)

Here, \( r(x) \geq 0 \) for all \( x \). This formulation is well known as KLIEP (KL importance estimation procedure). Given a density-ratio estimator \( \hat{G}(x) \), an approximator of actual KL divergence is given as follows:

\[
KL(\hat{r}(x)) = \frac{1}{n} \sum_{i=1}^{n} \log (\hat{G}(x_i)). \quad \text{Eq. 3-22}
\]

3.4.6 Mathematical Formulation of uLSIF Method

uLSIF, another well-known algorithm to estimate density-ratio described in [51] is used to estimate PE-divergence between two data distributions. The basic idea to compute Pearson Divergence is model \( \hat{r}(x) \) from actual density ratio \( r(x) \) (see Eq. 3-13) such that squared error \( SQ \) in the following equations is minimized:

\[
SQR' = \frac{1}{2} \int (\hat{r}(x) - r(x))^2 P_{\text{test}}(x) \, dx. \quad \text{Eq. 3-23}
\]

\[
SQR = \frac{1}{2} \int (\hat{r}(x))^2 P_{\text{test}}(x) \, dx - \int \hat{r}(x) \, r(x) P_{\text{test}}(x) \, dx + \frac{1}{2} \int r(x)^2 P_{\text{test}}(x) \, dx. \quad \text{Eq. 3-24}
\]

Replacing \( P_{\text{test}}(x) = \frac{p_{\text{ref}}(x)}{r(x)} \) (see Eq. 3-13) in the following equation:

\[
SQR' = \frac{1}{2} \int (\hat{r}(x))^2 P_{\text{test}}(x) \, dx - \int \hat{r}(x) \, p_{\text{ref}}(x) \, dx + \frac{1}{2} \int r(x) P_{\text{ref}}(x) \, dx. \quad \text{Eq. 3-25}
\]

The third term in Eq. 3-23 is not dependent on \( \hat{r}(x) \); therefore, it is a constant. We define another term \( SQR \) using first two terms in Eq. 3-24 as follows:

\[
SQR(\hat{r}(x)) = \frac{1}{2} \int (\hat{r}(x))^2 P_{\text{test}}(x) \, dx - \int \hat{r}(x) P_{\text{ref}}(x) \, dx. \quad \text{Eq. 3-26}
\]
Applying empirical averages, the approximation $SQR(\hat{r}(x))$ from $SQR(r(x))$ is achieved as follows:

$$SQR(\hat{r}(x)) = \frac{1}{2 \cdot n_{test}} \sum_{i=1}^{n_{test}} (\hat{\rho}(x_{i}^{n_{test}}))^2 - \frac{1}{n_{ref}} \sum_{i=1}^{n_{ref}} \hat{\rho}(x_{i}^{n_{ref}}). \tag{Eq. 3-27}$$

Hence, we can define the optimization problem to estimate density ratio using Pearson-divergence as follows: minimize $\hat{r}(x)$ such that $\frac{1}{2 \cdot n_{test}} \sum_{i=1}^{n_{test}} (\hat{\rho}(x_{i}^{n_{test}}))^2 - \frac{1}{n_{ref}} \sum_{i=1}^{n_{ref}} \hat{\rho}(x_{i}^{n_{ref}})$ is minimum. This formulation is well known as uLSIF (least-squares importance fitting). Given a density-ratio estimator $\hat{G}(x)$, an approximator of actual PE divergence is given as follows:

$$\hat{PE}(x) = -\frac{1}{2 \cdot n} \sum_{i=1}^{n} (\hat{G}(x_{i}))^2 + \frac{1}{n} \sum_{i=1}^{n} \hat{G}(x_{i}) + \frac{1}{2} \tag{Eq. 3-28}$$

### 3.4.7 Dynamic Cutoff Point

Static values have been widely used as cutoff points to decide the potential change points, as seen in [44, 49]. However, prior knowledge about the existing environment is required to set a cutoff value to be static or constant. To conduct our experiments, we adapted a dynamic approach to define the cutoff point at time $t_{cur}$ by using previously known divergence values which have already been used to detect changes at time $t_{i}$ where $t_{i} < t_{cur}$. Suppose the change points have been detected at time $t_{cp+1}, t_{cp+2}, t_{cp+3}, \ldots, t_{cp+n}$ with divergence $f_{cp}, f_{cp+1}, f_{cp+2}, \ldots, f_{cp+n}$ respectively within recent time interval $[t_{i}: t_{i+k}]$ where $t_{i+k} < t_{cur}$. We chose to use a recent time interval instead of a full time interval to lower the impact of older divergence values compared to recent divergence values. Next, we define range $f_{mean}$ and $f_{std}$ as follows:
\[ f_{\text{mean}} = \text{mean}(f_{cp}, f_{cp+1}, f_{cp+2}, \ldots, f_{cp+n}). \]  \hspace{1cm} \text{Eq. 3-29}

\[ f_{\text{std}} = \text{std}(f_{cp}, f_{cp+1}, f_{cp+2}, \ldots, f_{cp+n}). \]  \hspace{1cm} \text{Eq. 3-30}

Here, \( f_{cp+n} \) is the divergence value measured at \( t_{cp+n} \) where \( t_{cp+n} < t_{\text{cur}} \) and \( n = 1, 2, 3 \ldots \). Using \( f_{\text{mean}} \) and \( f_{\text{std}} \), we define the cutoff values with \( f_{\text{cutoff}} \) and change detection criteria as follows:

\[ f_{\text{cutoff}} = f_{\text{mean}} - f_{\text{std}}. \]  \hspace{1cm} \text{Eq. 3-31}

\[ cp = \{ 1, \quad f_{\text{cur}} \geq f_{\text{cutoff}} \}
\quad \quad \text{otherwise}. \]  \hspace{1cm} \text{Eq. 3-32}

### 3.4.8 Validation Criteria

We adapted the validation criteria described in [49]. Since our change point analysis follows a retrospective approach, certain delay to identify change point is allowed. Suppose the proposed method detects a change point at time \( t_{\text{detect}} \), and the closest actual change point exists at time \( t_{\text{actual}} \). The detected change point is considered to be correct if \( (t_{\text{actual}} - t_{\text{adjust}}) \leq t_{\text{detect}} \leq (t_{\text{actual}} + t_{\text{adjust}}) \) where \( t_{\text{adjust}} \) refers to time delay within an acceptable range. Since it is very difficult to distinguish the clear boundary where actual change of data distribution starts, duplicate alarms can be generated for several consecutive time points to detect the same change point. To avoid such duplication, we discarded the \( k^{th} \) alarm at step \( t_k \) if \( t_k - t_{k-1} \leq 2 * t_{\text{adjust}} \).

For artificial datasets with a sample size 5000, we set \( t_{\text{adjust}} \) equal to be 12.5, whereas for real datasets with a record size of \( 5 \times 10^5 \), we choose \( t_{\text{adjust}} \) equal to be 250. To set our validation criteria, we define \( n_{cr} \) as the number of times the change points are correctly detected, \( n_{cp} \) as the number of actual change points, and \( n_{al} \) as the number of detected change points, and \( n_{ncr} \) as the number of times change points are not correctly
detected. We can count \( n_{ncr} \) by observing if there exists a change point at step \( t_{actual} \) but no alarm in \( (t_{actual} - t_{adjust}) \leq t \leq (t_{actual} + t_{adjust}) \). We define \( TP \) (True positive), \( FP \) (False positive), \( FN \) (False negative), \( P \) (precision), \( R \) (Recall), \( F1 - Score \) as follows:

\[
TP = \frac{n_{cr}}{n_{cp}}, \quad \text{Eq. 3-33}
\]

\[
FP = \frac{n_{al} - n_{cr}}{n_{al}}, \quad \text{Eq. 3-34}
\]

\[
FN = \frac{n_{ncr}}{n_{cp}}, \quad \text{Eq. 3-35}
\]

\[
P = \frac{TP}{TP + FP}, \quad \text{Eq. 3-36}
\]

\[
R = \frac{TP}{TP + FN}, \quad \text{Eq. 3-37}
\]

\[
F1 - Score = \frac{2 \times P \times R}{P + R}. \quad \text{Eq. 3-38}
\]

### 3.5 Results

In this section, we investigated the performance of the proposed change point detection technique using both artificial (Section 3.3.1 – 3.3.3) and real datasets (Section 3.3.4). At first, we synthesized three types of datasets using multi variate random generators (`mvtnorm` package in R [57]). For each type, we created four datasets with a sample size of 5000 each and dimensions of 25, 50, 75 and 100, respectively. Hence, we created a total of twelve artificial datasets to conduct our experiments. We introduced a change point every 100 samples by adding Gaussian random variable or by varying the combination of the mean and covariance matrix. We added timestamp to each sample to make the datasets compatible for the sliding window technique. For simplicity, we considered only one sample collected at each time point. We divided the current window
into two data segments, reference data segment and test data segment. For comparison, we applied two different density ratio estimation technique, KLIEP and uLSIF using densratio package in R [61], on original and PCA transformed dataset separately.

At first, we applied the KLIEP method on multidimensional data segments with and without PCA transformation to measure respective KL-divergence and compared their performance in terms of TPR (true positive rate), FPR (false positive rate), and FNR (false negative rate) and computational time to analyze the contribution of PCA on KLIEP performance. Next, we applied a similar approach by using the uLSIF method on the same data segments with and without PCA transformation to measure respective PE-divergence and compared their performances to analyze the contribution of PCA on uLSIF performance. After analyzing the contribution of PCA transformation on individual methods, we shifted our focus to compare the performance between KLIEP and uLSIF on PCA transformed data segments. Using twelve artificial datasets, we ran a total of 48 experiments and used the results of these experiments to demonstrate the superiority of the proposed framework. In a later section, we further reinforced our claim by comparing the proposed framework with three base line methods using seven public datasets to demonstrate its effectiveness.

3.5.1 Results: Type # 01 Dataset

Figure 3-1 depicts the scenario of detecting potential change points for type #01 datasets where KL-divergence or PE-divergence undergoes noticeable changes. Table 3-3 shows the results of the KLIEP method on PCA reduced features on these four-time series datasets, whereas Table 3-4 presents the results of the uLSIF method on PCA reduced features.
Figure 3-1: Potential change points detection for type 01 datasets where KL-divergence and PE-divergence undergoes noticeable changes. (a) & (b) time vs. KL-divergence between PCA transformed data segments (original feature dimension equals to 25, and 75, respectively). (c) & (d) time vs. PE-divergence between PCA transformed data segments with original feature dimension equals to 75, and 100, respectively.

Table 3-3: Comparison of performance of KLIEP on full features and PCA reduced features for four type #01 synthetic datasets.

<table>
<thead>
<tr>
<th># of features</th>
<th>Only KLIEP</th>
<th>PCA + KLIEP</th>
<th>Avg runtime per window (in sec)</th>
<th>Avg runtime per window (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP (%)</td>
<td>FP (%)</td>
<td>FN (%)</td>
<td>TP (%)</td>
</tr>
<tr>
<td>50</td>
<td>77.551</td>
<td>18.502</td>
<td>22.449</td>
<td>83.673</td>
</tr>
<tr>
<td>75</td>
<td>75.510</td>
<td>18.943</td>
<td>24.900</td>
<td>81.633</td>
</tr>
<tr>
<td>100</td>
<td>69.388</td>
<td>20.264</td>
<td>30.612</td>
<td>75.510</td>
</tr>
</tbody>
</table>
Table 3-4: Comparison of performance of uLSIF on full features and PCA reduced features for four type #01 synthetic datasets

<table>
<thead>
<tr>
<th># of features</th>
<th>Only uLSIF</th>
<th>PCA + uLSIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP (%)</td>
<td>FP (%)</td>
</tr>
<tr>
<td>50</td>
<td>81.633</td>
<td>17.621</td>
</tr>
<tr>
<td>75</td>
<td>79.592</td>
<td>18.062</td>
</tr>
<tr>
<td>100</td>
<td>71.429</td>
<td>19.824</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>79.592</td>
<td>18.062</td>
</tr>
</tbody>
</table>

3.5.2 Results: Type #02 Dataset

Figure 3-2 illustrates the scenario of detecting potential change points for type #02 datasets where KL-divergence or PE-divergence undergoes noticeable changes.

Figure 3-2: Potential change points detection for type 02 datasets where KL-divergence or PE-divergence undergoes noticeable changes. (a) & (b) Time vs. KL-divergence between PCA transformed data segments with original feature dimension equals to 50 and 100 respectively. (c) & (d) Time vs. PE-divergence between PCA transformed data segments with original feature dimension equals to 25 and 75 respectively.
Table 3-5 shows the results of the KLIEP method on full features and PCA reduced features on these four-time series datasets, whereas Table 3-6 presents the results of the uLSIF method on full features and PCA reduced features.

### Table 3-5: Comparison of performance of KLIEP on full features and PCA reduced features on four type #02 datasets

<table>
<thead>
<tr>
<th># of features</th>
<th>TP (%)</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Avg runtime per window (in sec)</th>
<th>TP (%)</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Avg runtime per window (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>83.673</td>
<td>15.315</td>
<td>16.327</td>
<td>13.74</td>
<td>89.796</td>
<td>15.111</td>
<td>10.204</td>
<td>9.99</td>
</tr>
<tr>
<td>100</td>
<td>77.551</td>
<td>17.040</td>
<td>22.449</td>
<td>39.66</td>
<td>79.592</td>
<td>16.964</td>
<td>20.408</td>
<td>13.36</td>
</tr>
<tr>
<td>Average</td>
<td>82.143</td>
<td>16.123</td>
<td>17.857</td>
<td>21.75</td>
<td>86.224</td>
<td>15.985</td>
<td>13.776</td>
<td>11.06</td>
</tr>
</tbody>
</table>

### Table 3-6: Comparison of performance of uLSIF on full features and PCA reduced features on four type #02 datasets

<table>
<thead>
<tr>
<th># of features</th>
<th>TP (%)</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Avg runtime per window (in sec)</th>
<th>TP (%)</th>
<th>FP (%)</th>
<th>FN (%)</th>
<th>Avg runtime per window (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>83.673</td>
<td>16.444</td>
<td>16.327</td>
<td>7.62</td>
<td>91.837</td>
<td>15.044</td>
<td>8.163</td>
<td>4.33</td>
</tr>
<tr>
<td>75</td>
<td>77.551</td>
<td>18.142</td>
<td>22.449</td>
<td>9.71</td>
<td>87.755</td>
<td>15.929</td>
<td>12.245</td>
<td>4.92</td>
</tr>
<tr>
<td>100</td>
<td>73.469</td>
<td>19.027</td>
<td>26.531</td>
<td>11.65</td>
<td>83.673</td>
<td>16.444</td>
<td>16.327</td>
<td>5.55</td>
</tr>
<tr>
<td>Average</td>
<td>80.612</td>
<td>17.478</td>
<td>19.388</td>
<td>9.07</td>
<td>90.306</td>
<td>15.284</td>
<td>9.694</td>
<td>4.70</td>
</tr>
</tbody>
</table>

### 3.5.3 Results: Type #03 Dataset

Figure 3-3 exemplifies the situation of detecting potential change points for type #03 datasets where KL-divergence or PE-divergence experience visible changes after PCA transformation. Table 3-7 shows the results of the KLIEP method on full features and PCA reduced features on these four-time series datasets whereas Table 3-8 presents the results of the uLSIF method on full features and PCA reduced features.
Figure 3-3: Potential change points detection for type 03 datasets where KL-divergence or PE-divergence undergoes noticeable changes. (a) & (b) Time vs. KL-divergence between PCA transformed data segments with original feature dimension equals to 25 and 100 respectively. (c) & (d) Time vs. PE-divergence between PCA transformed data segments with original feature dimension equals to 50 and 75 respectively.

Table 3-7: Comparison of performance of KLIEP on full features and PCA reduced features on four type #03 datasets.

<table>
<thead>
<tr>
<th># of features</th>
<th>Only KLIEP</th>
<th>PCA + KLIEP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP (%)</td>
<td>FP (%)</td>
</tr>
<tr>
<td>50</td>
<td>83.673</td>
<td>16.071</td>
</tr>
<tr>
<td>75</td>
<td>75.510</td>
<td>18.943</td>
</tr>
<tr>
<td>100</td>
<td>69.388</td>
<td>19.912</td>
</tr>
<tr>
<td>Average</td>
<td>75.510</td>
<td>18.577</td>
</tr>
</tbody>
</table>
Table 3-8: Comparison of performance of uLSIF on full features and PCA reduced features on four type #03 datasets.

<table>
<thead>
<tr>
<th># of features</th>
<th>Only uLSIF</th>
<th>PCA + uLSIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP (%)</td>
<td>FP (%)</td>
</tr>
<tr>
<td>25</td>
<td>83.673</td>
<td>16.814</td>
</tr>
<tr>
<td>75</td>
<td>77.551</td>
<td>18.142</td>
</tr>
<tr>
<td>100</td>
<td>75.510</td>
<td>17.117</td>
</tr>
<tr>
<td>Average</td>
<td>79.082</td>
<td>17.259</td>
</tr>
</tbody>
</table>

We measured the improvement of performance of both KLIEP and uLSIF due to dimensionality reduction by the PCA as follows:

\[
\text{Improvement}_X = \frac{X_{\text{average}}^{\text{Full+method}} - X_{\text{average}}^{\text{PCA+method}}}{X_{\text{average}}^{\text{Full+method}}} \times 100. \quad \text{Eq. 3-39}
\]

Here, \( X = TP, FP, FN, \) execution time and \( \text{method} = \text{KLIEP}, \text{uLSIF} \) for average values. Using equation (24), Table 3-9 shows all the results of performance improvement for TP, FP, FN, and execution time due to PCA transformation.

Table 3-9: Performance improvement of KLIEP and uLSIF methods in terms of TP, FP, FN and execution time due to PCA.

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>(Original Features + KLIEP) Vs. (PCA + KLIEP)</th>
<th>(Original Features + uLSIF) Vs. (PCA + uLSIF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Change in TP (%)</td>
<td>Change in FP (%)</td>
</tr>
<tr>
<td>Type 01</td>
<td>(+)10.34</td>
<td>(-)10.05</td>
</tr>
<tr>
<td>Type 02</td>
<td>(+)4.97</td>
<td>(-)0.86</td>
</tr>
<tr>
<td>Type 03</td>
<td>(+)6.76</td>
<td>(-)0.60</td>
</tr>
<tr>
<td>Average</td>
<td>(+)7.36</td>
<td>(-)3.83</td>
</tr>
</tbody>
</table>
We observed that for both KLIEP and uLSIF, the values of TP increased whereas values of FP, FN and execution time decreased when PCA was used. We also observed significant reduction in runtime due to dimensionality reduction. All results justified that KLIEP and uLSIF, both density ratio estimation methods, perform better when PCA is used to reduce the dimensionality of the feature space in the streaming environment. We illustrated this observation visually in Figure 3-4.

**Figure 3-4**: The results of performance improvement of KLIEP method in terms of TP, FP, FN, and execution time for PCA

All results justified that KLIEP and uLSIF, both density ratio estimation methods, perform better when PCA is used to reduce the dimensionality of the feature space in the streaming environment. We illustrated this observation visually in Figure 3-5.
Figure 3-5: The results of performance improvement of uLSIF method in terms of TP, FP, FN, and execution time for PCA.

The results demonstrated that KLIEP and uLSIF, both density ratio estimation methods, performed better when dimensionality of the features space was reduced by PCA which also justified our hypothesis. However, when dimensionality has already been reduced by the PCA, the overall performance of the uLSIF method is better than the KLIEP in terms of $TP, FP, FN$, and execution time which is shown in Table 3-10 and Figure 3-6.

Table 3-10: Overall comparison of performance improvement between KLIEP and uLSIF on PCA transformed data.

<table>
<thead>
<tr>
<th>dataset Type</th>
<th>PCA + KLIEP</th>
<th></th>
<th></th>
<th>PCA + uLSIF</th>
<th></th>
<th></th>
<th></th>
<th>Avg runtime per window (in sec)</th>
<th>Avg runtime per window (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP (%)</td>
<td>FP (%)</td>
<td>FN (%)</td>
<td>Avg runtime per window (in sec)</td>
<td>TP (%)</td>
<td>FP (%)</td>
<td>FN (%)</td>
<td>Avg runtime per window (in sec)</td>
<td></td>
</tr>
<tr>
<td>Type 01</td>
<td>81.633</td>
<td>17.257</td>
<td>18.367</td>
<td>11.30</td>
<td>86.224</td>
<td>16.448</td>
<td>13.776</td>
<td>5.35</td>
<td></td>
</tr>
<tr>
<td>Type 02</td>
<td>86.224</td>
<td>15.985</td>
<td>13.776</td>
<td>11.06</td>
<td>90.306</td>
<td>15.284</td>
<td>9.694</td>
<td>4.70</td>
<td></td>
</tr>
<tr>
<td>Type 03</td>
<td>80.612</td>
<td>18.466</td>
<td>19.388</td>
<td>11.33</td>
<td>84.184</td>
<td>16.792</td>
<td>15.816</td>
<td>4.87</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>82.823</td>
<td>17.236</td>
<td>17.177</td>
<td>11.230</td>
<td>86.905</td>
<td>16.175</td>
<td>13.095</td>
<td>4.97</td>
<td></td>
</tr>
</tbody>
</table>
3.5.4 Results: Public Datasets

We also compared our proposed framework MCD-PuLSIF with three baseline methods known as CD-LLH (Change Detection by Log Likelihood), CD-MKL (Change Detection by Maximum KL Divergence), CD-Area (Change Detection by Intersection Area between two data distribution) presented in [54] so that the final datasets turn out to be similar and comparison of the experimental results between the proposed framework and the three base line methods, CD-LLH, CD-MKL, and CD-Area, are justified. At first, we resized each dataset up to $5 \times 10^5$. Suppose $x$ is a randomly chosen data sample for dataset $D_i$ with record size $n_{D_i}$. Finally, we applied the proposed MCD-PuLSIF framework on these fourteen datasets and the experimental results using F1-score shown in the rightmost column of Table 3-11.
Table 3-11: Comparison of experimental results of MCD-PuLSIF method with CD-LLH, CD-MKL, and CD-Area using real world datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>kdt-tree</th>
<th>PCA-SPLL</th>
<th>CD-LLH</th>
<th>CD-MKL</th>
<th>CD-Area</th>
<th>MCD-PuLSIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spruce(G1D)</td>
<td>0.676</td>
<td>0.493</td>
<td>0.913</td>
<td>0.846</td>
<td>0.925</td>
<td>0.917</td>
</tr>
<tr>
<td>Spruce(S1D)</td>
<td>0.676</td>
<td>0.346</td>
<td>0.778</td>
<td>0.772</td>
<td>0.884</td>
<td>0.895</td>
</tr>
<tr>
<td>Lodgepole Pine(G1D)</td>
<td>0.492</td>
<td>0.446</td>
<td>0.872</td>
<td>0.856</td>
<td>0.904</td>
<td>0.920</td>
</tr>
<tr>
<td>Lodgepole Pine(S1D)</td>
<td>0.346</td>
<td>0.735</td>
<td>0.794</td>
<td>0.741</td>
<td>0.839</td>
<td>0.914</td>
</tr>
<tr>
<td>Ascending Stairs(G1D)</td>
<td>0.585</td>
<td>0.504</td>
<td>0.838</td>
<td>0.785</td>
<td>0.848</td>
<td>0.891</td>
</tr>
<tr>
<td>Ascending Stairs(S1D)</td>
<td>0.592</td>
<td>0.772</td>
<td>0.995</td>
<td>0.761</td>
<td>0.821</td>
<td>0.843</td>
</tr>
<tr>
<td>Cycling(G1D)</td>
<td>0.614</td>
<td>0.352</td>
<td>0.814</td>
<td>0.832</td>
<td>0.849</td>
<td>0.870</td>
</tr>
<tr>
<td>Cycling(S1D)</td>
<td>0.658</td>
<td>0.387</td>
<td>1.000</td>
<td>0.805</td>
<td>0.833</td>
<td>0.895</td>
</tr>
<tr>
<td>Descending Stairs(G1D)</td>
<td>0.563</td>
<td>0.535</td>
<td>0.809</td>
<td>0.828</td>
<td>0.811</td>
<td>0.901</td>
</tr>
<tr>
<td>Descending Stairs(S1D)</td>
<td>0.746</td>
<td>0.406</td>
<td>0.989</td>
<td>0.878</td>
<td>0.872</td>
<td>0.866</td>
</tr>
<tr>
<td>Ironing(G1D)</td>
<td>0.599</td>
<td>0.559</td>
<td>0.862</td>
<td>0.753</td>
<td>0.772</td>
<td>0.894</td>
</tr>
<tr>
<td>Ironing(S1D)</td>
<td>0.649</td>
<td>0.615</td>
<td>0.995</td>
<td>0.794</td>
<td>0.828</td>
<td>0.869</td>
</tr>
<tr>
<td>Vacuum cleaning(G1D)</td>
<td>0.473</td>
<td>0.628</td>
<td>0.809</td>
<td>0.831</td>
<td>0.822</td>
<td>0.892</td>
</tr>
<tr>
<td>Vacuum cleaning(S1D)</td>
<td>0.691</td>
<td>0.691</td>
<td>0.985</td>
<td>0.782</td>
<td>0.814</td>
<td>0.920</td>
</tr>
<tr>
<td>Average (F1-Score)</td>
<td>0.597</td>
<td>0.534</td>
<td>0.890</td>
<td>0.805</td>
<td>0.844</td>
<td>0.892</td>
</tr>
</tbody>
</table>

At first, we searched five nearest neighbors \( x_1, x_2, \ldots, x_5 \) of \( x \) (FNN R package [62]) followed by drawing \( \left\lceil \frac{5 \times 10^5 - n_{D_i}}{n_{D_i}} \right\rceil \) number of random samples from these five nearest neighbor samples with a replacement and creating a new sample by taking their average. This technique was first used in [63] to boost the dataset up to a certain size while retaining the same distributional properties. After resizing each dataset up to \( 5 \times 10^5 \), we infused two types of changes every \( 2 \times 10^4 \) data samples. The first type of change was incorporated by adding one dimensional Gaussian random variable (G1D) to a randomly selected feature every \( 2 \times 10^4 \) data samples and the second type of change was fused by
scaling a randomly selected feature by two (S1D) every $2 \times 10^4$ data samples. We used these two types of changes because they affect only a single dimension of the dataset and they are harder to detect.

The results of three baseline methods known as CD-LLH, CD-MKL, and CD-Area along with kdt-tree and PCA-SPILL are taken directly from [54] where CD-LLH, CD-MKL and CD-Area methods were compared with two other methods, kdt-tree [64] and PCA-SPILL [65]. We used bold font with red color where baseline methods performed better than the proposed method.

From the experimental results, we observed the overall performance of the proposed MCD-PuLSIF framework is significantly better than CD-MKL (F1-Score 0.892 compared to 0.805) and CD-Area (F1-Score 0.892 compared to 0.844). However, the average performance of MCD-PuLSIF and CD-LLH are almost the same (F1-score 0.892 compared to 0.890), but when we observed carefully, we found that the performance of CD-LLH method outperformed MCD-PuLSIF only for activity datasets with S1D.

Through further investigation, we observed that not only the performance of CD-LLH method experienced larger variations in F1-score ranging from 0.778 to 1.00, but also CD-LLH underperformed more than MCD-PuLSIF for other datasets. CD-LLH showed lower performance than CD-MKL and CD-Area except for activity datasets with S1D. Therefore, the overall performance of MCD-PuLSIF is more consistent than CD-LLH when all datasets are considered. Figure 3-7 depicts the compared performance of MCD-PuLSIF with CD-LLH, CD-MKL, CD-Area using different types of datasets separately.
Figure 3-7: Comparison of experimental results of MCD-PuLSIF method with CD-LLH, CD-MKL, and CD-Area using cover type and activity datasets separately.

3.5.5 Computation Complexity Analysis

For our experiment, we applied SVD for PCA transformation. Time complexity of SVD is $O(\min(n^2p, p^2n))$ [52], where $n$ and $p$ refer to the sample size per window and feature dimension, respectively. For the worst-case scenario, when $n$ and $p$ are equal, the worst cast time complexity of PCA using SVD is $O(n^3)$. Table 3-12 shows runtime considering several combinations of window size and feature dimension.
Table 3-12: Comparison of runtime against window size and feature dimension for MCD-PuLSIF.

<table>
<thead>
<tr>
<th>Window Size</th>
<th>No of PCAs</th>
<th>Run time(sec)</th>
<th>Window Size</th>
<th>No of PCAs</th>
<th>Run time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>10</td>
<td>46.12</td>
<td>15000</td>
<td>30</td>
<td>110.49</td>
</tr>
<tr>
<td>10000</td>
<td>20</td>
<td>44.99</td>
<td>20000</td>
<td>10</td>
<td>185.89</td>
</tr>
<tr>
<td>10000</td>
<td>30</td>
<td>45.42</td>
<td>20000</td>
<td>20</td>
<td>186.20</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>111.17</td>
<td>20000</td>
<td>30</td>
<td>185.67</td>
</tr>
<tr>
<td>15000</td>
<td>20</td>
<td>111.00</td>
<td>----</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

**Figure 3-8(a-c)** depicts that given a fixed window size, runtime remains almost constant even when the number of PCAs are varied from 10 to 30. On the other hand, **Figure 3-8(e-g)** shows that runtime increases with an increase of window size from 1000 to 20000. Data from **Table 3-12** and graphs from **Figure 3-8** depict that the window size plays a major role for runtime, since the impact of feature dimension on runtime is very little.

**Figure 3-8**: Comparison of runtime against window size and feature dimension for MCD-PuLSIF.
On the other hand, uLSIF is a convex quadratic problem [51], which can be computed by solving a set of linear equations. For a data matrix $n \times p$, $n$ number of linear equations with $p$ number of coefficients are required to be solved. Time complexity of uLSIF is $O(n^2p)$. When $n$ and $p$ are equal, so the worst cast time complexity of uLSIF is $O(n^3)$. Hence, for worst case scenario, the total time complexity of proposed framework is $O(n^3)$. The runtime complexity of CD-LLH, CD-MKL and CD-Area is $O(n^3)$ mentioned in [54], which is theoretically the same as the worst case runtime complexity of MCD-PuLSIF, but we could not make practical comparison on runtime since detailed data on runtime is unavailable. However, some figures in [54] show that runtime of CD-LLH, CD-MKL and CD-Area vary from 100 sec to 700 sec based on several combinations of window size and feature dimension. However, the maximum run time of MCD-PuLSIF is 186 sec for window size varying from 10000 to 20000 and feature dimensions varying from 10 to 30. Therefore, we can draw the conclusion that proposed framework is a better technique compared to these three-baseline methods in terms of reducing the run time along with improving the change detection rate.

### 3.6 Findings and Discussion

In this chapter, we proposed a useful unsupervised framework combining feature space model using PCA computed by SVD and density ratio method known as uLSIF to detect change point in time series data streams with multidimensional feature space with better accuracy and efficiency. We applied a sliding window technique to limit the amount of streaming data processing to manage memory and time. To make decisions about plausible occurrence of change points, we used dynamic thresholding instead of static thresholding. We compared our framework with another density ratio estimation method
known as KLIEP and three other baseline methods using artificial and real datasets which demonstrated the effectiveness of our technique using the experimental results. Some potential application areas are also identified where the proposed technique can be explored. For our experiments, the proposed framework focused only on unlabeled numerical data for both artificial and real datasets. We can further investigate how the proposed technique performs in the presence of mixed or labeled variables in datasets. However, we left all these issues open as potential future works.

In the next chapter, we applied unsupervised feature space modeling using matrix factorization and Lasso Regression to reduce the dimensionality of botnet affected network traffic data collected in a non-stationary environment which subsequently used by Relative Density Ratio Estimation method to detect malicious attacks launched from botnets with better accuracy and efficiency.
CHAPTER 4
UNSUPERVISED FEATURE SPACE MODELING FOR BOTNETS DETECTION FRAMEWORK

4.1 Chapter Overview

In this chapter, we built an unsupervised feature space model using matrix factorization, Lasso Regression to select the most relevant features as well as to reduce the dimension of network traffic data infused with malignant data generated from botnets in a non-stationary environment. Next, we modeled the relative ratio of probability distributions of two data segments and explore this statistical measure to identify attacks launched from compromised IoT devices, collectively known as botnets with better accuracy and efficiency. We proposed a semi-supervised network based IoT botnet detection framework consisting of two major components using network traffic statistics.

The first component, an unsupervised feature selection method, selected useful features from several network traffic statistics using QR factorization, Singular Value Decomposition, and Lasso Regularized Regression. The second component comprised of relative density ratio estimation method, known as Relative Unconstrained Least Square Importance Fitting, estimated the ratio of probability density of traffic instances in the current window with respect to the reference window. It was subsequently employed to prove the hypothesis that the data distributions of benign and malicious network traffic are different, and thus the traffic instances with low probability densities can be regarded as
anomalies or malicious generated from the plausible botnets. To conduct our experiments, we used real datasets on IoT devices infected by BASHLITE and Mirai botnets. We validated the proposed framework by comparing its performance with four baseline methods. We observed better and more consistent performance by the proposed framework in terms of higher botnet detection rate and lower computational time which demonstrated its applicability and effectiveness.

4.2 Related Works

Botnet is a robot or zombie network consisting of infected or compromised network devices known as bots which are capable of running malicious software scripts under the command and control (C&C) server with a wide range of purposes, for example, scanning and infecting other vulnerable devices, launching distributed denial-of-service (DDoS) attacks, cracking default or weak passwords, sending spam email, logging public and private keys, and mining cryptocurrency, etc. [66, 67]. Compared to conventional computing systems, IoT infrastructure is much more robust and convenient for creating large-scale botnets with significant network bandwidth and computational power since attacks from malicious agents performed from distributed infrastructures are more effective since they can exploit more resources to destabilize the target network devices which may demand more advanced tracking, detecting and mitigating mechanisms [68]. Among the widely used malicious software to launch DDoS attacks, BASHLITE and Mirai are very popular among attackers. BASHLITE is one of the most widely used malware programs. It launched over 1 million DDoS attacks to infect primarily Linux based IoT devices. It targeted mostly cameras and DVRs connected with the network [69].
Most BASHLITE attacks are simple UDP, TCP floods and HTTP requests which are capable of launching attacks of up to 400 Gbps [68, 69]. Mirai, another prevalent malware, targets mostly internet enabled CCTV cameras, DVRs, and home routers [69-71]. Mirai generates floods of GRE IP, GRE ETH, SYN and ACK, STOMP, DNS, UDP, or HTTP traffic against a target network device during a DDoS attack with strengths ranging from 200 Gbps to 1.2 Tbps [70, 71]. Since the release of the Mirai source code, the number of IoT infected devices has increased from 213,000 to 483,000 in just two weeks [69]. IoT devices cover a wide range of services, for example, smart home, healthcare, and transport automation, smart energy solutions, and extremely complicated industrial control systems, and thus the number of IoT devices is enormous. Therefore, data produced from these massive number of devices can range up to Tbps levels, and their value is worth billions of dollars. According to [66], in September 2016, French web host OVH was attacked by a Mirai botnet which has been recorded as the most massive DDoS attack hacking as large as 1.5 Tbps of data; another Mirai based DDoS attack was launched in the same month against Brian Krebs’s security blog with 600 Gbps. DDoS attacks led to financial losses on the order of 2 billion dollars per year [68].

Botnet attacks such as BASHLITE or Mirai impose huge threat both in terms of network safety and financial loss. For many countries, this is an emergency issue concerning national security. Early detection of botnet attacks can accelerate the process of alert mechanisms and disconnection of compromised IoT devices from the network which may further help in stopping the botnets from propagating and infecting new devices [66, 67], and thus IoT based botnet related research has garnered immense attention for researchers in recent time. Among other network traffic characteristics, network statistics
and behavioral features have been used for years by researchers in investigating, designing, and implementing botnets detection mechanism [72, 73].

DDoS attacks such as BASHLITE and Mirai, particularly in the IoT domain, result in a significant impact to both entrepreneurs and end users in terms of security risk and financial loss [66-68, 73]. A substantial amount of research has been done in analyzing the evolution of DDoS attacks, categorizing and creating taxonomies, illustrating the effect of botnet attacks, characterizing the frequently used BASHLITE and Mirai botnets, and exploring the DDoS attack underground market [66-75]. For years, researchers have relied on several network traffic characteristics in investigating, designing, and implementing botnets detection techniques, for example, use of suspicious traffic behavior [74, 76], actions in honeypots [77-79], communication protocols [76], networks statistics and behavioral features [67], graphical illustrations of network behaviors [73, 80], collaborative feedback collected from large networks [81], etc.

Among botnet detection mechanisms, honeypots are widely studied and explored. A honeynet is a trap network-attached system for luring hackers to engage and deceive hackers, tracking unconventional or new hacking attempts and identifying malicious activities performed over IoT devices [79]. Reference [73] presents a detailed taxonomy to classify the detection mechanisms by separating the detection sources into honeynets and intrusion detection system (IDS). Anomaly-based systems and signature-based systems are two major sub-categories of IDS, whereas the anomaly-based system is further grouped into network-based systems and host-based systems. Two more sub-categories are found under network-based systems which are known as active monitoring and passive monitoring. Botnet detection mechanisms, for example, host-based [82, 83] and network-
based [74, 76, 77, 84] techniques search discriminatory or anomalous pattern based on several network traffic characteristics, for example, malicious DNS traffic launched from C&C servers, signatures recognized by honeypots [78, 79], anomalous network traffic [67, 74], data mining and machine learning techniques [85], and multilayer hybrid detection approaches [75, 86, 87]. A different software-based edge-oriented mitigation solution using Software Defined Network (SDN) and Fog computing to detect Mirai botnet are presented in [88, 89].

Recently, Neural Network (NN) particularly Deep Neural Network (DNN) are explored in this area of botnet detection. For example, Recurrent Neural Network (RNN) has been explored in [90], whereas Auto Encoder has been used in [67] as a threat detection mechanism although a deep learning algorithm is computationally very expensive due to multiple hidden layers, and thus the deployment of these mechanisms are particularly convenient for the high-performance cloud computing environment. Moreover, DNN methods are highly parametric which require a lot of parameter tuning and manual inspection.

With the continuous change of heterogeneous IoT devices, the botnet attack or hacking networks are also evolving with more sophistication, and thus it is recommended to rely on multilayer security mechanism using a hybrid approach rather than a single layer security mechanism [72, 86, 87]. Hybrid approaches may combine both host-based and network-based mechanisms to prevent attacks before a launch as well as to detect attacks after the launch. For example, antivirus software, farewells, content filtering, or inspection technologies can be installed as preventative measures, whereas network traffic can be continuously monitored from network gateway using network-based detection methods to
find anomalous events or trend indicating potential risk after attacks have been launched. With this context of hybrid security approach to ensure an end to end security, our proposed data-driven framework deployed in a network router or gateway can add an extra level of network security to detect anomalous network traffic.

According to [72], IoT botnet detection mechanisms encounter four common problems: 1) lack of real datasets to conduct research, 2) lack of benign or standard traffic models, and 3) identifying essential and non-redundant features to mitigate overfitting problem and 4) reliability problems in the validation criteria. We addressed the first two issues by using real datasets publicly available in UCI machine repository [67] instead of using simulated datasets. The training window of the proposed framework consists of benign or normal traffic instances since the donated botnet attack related dataset is segmented into benign traffic data as well as malicious data from BASHLITE and Mirai botnets.

To address the third issue, we built an unsupervised feature selection method using QRcp, SVD and Lasso regularized regression. We applied Lasso regularization to avoid the problem of overfitting. To address the fourth and final issue, we have used F1-score as the performance validation criteria while comparing the performance with other anomaly detection systems. Other validation criteria such as accuracy, ROC-AUC, sensitivity, specificity, and TPR-FPR may give misleading performance measures.

The first component is an unsupervised feature selection method based on matrix factorization to select useful features (i.e., network statistics) as well as to reduce the dimensionality of the feature space. We adopted the concept of unsupervised feature selection method described in [91, 92] which used one-step matrix sketching by SVD.
without considering the actual rank of the matrix. We modified the approach by using QRcp and SVD in two consecutive steps of matrix sketching.

First, we applied QR factorization [93] to sketch or shrink the data matrix in terms of feature space by measuring the actual rank of data matrix using a column-wise pivotal element calculation. Next, we applied Singular Value Decomposition (SVD) [92, 94] on the shrunken data matrix to efficiently maintain a low-rank approximation of the observed features retaining 95% of total variances of the original dataset. Moreover, finally, we used Lasso regularized regression on this approximation to measure feature score to identify the critical features regarding the right singular vectors found from SVD decomposition as the regression target [91, 92]. The second component of the proposed framework comprises of relative density ratio estimation method known as Relative Unconstrained Least Square Importance Fitting (RuLSIF) [95]. RuLSIF estimates the ratio of the probability density of each data instance in the current window with respect to the training window based on the hypothesis that the data distribution of benign network traffic is different from the data distribution of malicious network traffic. Thus, traffic data instances with low probability densities can be regarded as an anomaly generated from plausible botnets imposing potential security threats.

The proposed system can exhibit multiple advantages. First, the use of a training window filling with benign traffic instances as baseline actions to discriminate between benign and malicious traffic makes the proposed framework reasonably independent of using a dedicated class label for malicious traffic provided by the experts to make necessary decisions. Therefore, the proposed framework can detect potential future unseen and unknown abnormal patterns from botnets apart from BASHLITE and Mirai. Second, the
The proposed system is highly tolerant of diversity and heterogeneity exhibited by IoT devices. As our approach to detect botnet attacks is solely data-driven using only network traffic statistics, other network related issues such as communication medium and protocols, platforms, and other third-party devices, C&C encryption impose a little impact on it.

Finally, the proposed network-based mechanism does not consume dedicated resources such as computational memory, network bandwidth, or energy from IoT devices to endanger their normal operation, and thus this data-driven network-based system can easily be installed in a network gateway to ensure final stage security while monitoring network traffic. The significant contribution of our chapter is to explore RuLSIF method by combining it with unsupervised feature selection in IoT security domain. RuLSIF, a relative density ratio estimation method between two data distributions, has previously been used in detecting change points in single dimensional data stream [49], land cover change detection in non-Gaussian time-series data [96], estimating feature importance [97], categorizing glucose level for type 2 diabetic patients [98], etc. RuLSIF is known for its numerical stability and faster computation even when the feature dimension is very high [49, 95]. To the best of our knowledge, we are the first to use the RuLSIF method in building a data-driven network-based IoT security mechanism using network traffic statistics to detect malicious attacks launched from compromised IoT devices with better performance in terms of improving the botnet detection rate and reducing computational time.

The remainder of this chapter is arranged as follows. In Section 4.4, we described the algorithm and detail methodologies for our proposed technique. In Section 4.5, we
reported experimental results using real-world datasets with necessary tables and graphs. Finally, we conclude by summarizing our contribution in Section 4.6.

4.3 Data Set Information

The dataset IoT_botnet_attacks_N_BaIoT, publicly available on UCI machine repository [67, 99], addresses the lack of public botnet datasets, especially for the IoT devices. The dataset provides 7,062,6069 (7M approx.) network traffic data instances with 115 variables or features in the form of network traffic statistics from 9 IoT devices which are Danmini Doorbell (D1), Ecobee Thermostat (D2), Ennio Doorbell (D3), Philips B120N10 Baby Monitor (D4), Provision PT 737E Security Camera (D5), Provision PT 838 Security Camera (D6), Samsung SNH 1011 N Webcam (D7), SimpleHome XCS7 1002 WHT Security Camera (D8), and SimpleHome XCS7 1003 WHT Security Camera (D9). Summary statistics of the recent traffic from the packet's host IP and MAC addresses, the weight of the network traffic stream, the root squared sum of the two streams' mean, variances, approximated covariance, approximated correlation coefficient, etc. are some examples of 115 features. This dataset enables empirical evaluation with real traffic data collected from nine commercial IoT devices compromised by two of the most common and harmful botnets, Mirai and BASHLITE in an isolated network. The dataset was donated by the Department of Software and Information Systems Engineering, at the Ben-Gurion University of the Negev in Israel.

4.4 Algorithm and Methodology

In this section, we described our proposed IBDS framework using several methods, techniques, mathematical formulation, and validation criteria in detail. At the end of this section, we have also presented the algorithm (Algorithm 1) of the proposed framework.
4.4.1 Unsupervised Feature Selection Using Matrix Factorization

Frobenius norm: The Frobenius norm of a matrix $X$ with dimension $n \times m$ is defined as follows:

$$\|X\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j}|^2}.$$  \hspace{1cm} \text{Eq. 4-1}

Here, $n$ refers to the number of records and $m$ refers to the number of features or columns. The Frobenius norm is also known as the $L_2$-norm or Euclidean norm.

QR decomposition with Column Pivoting (QRcp): Given $X$ is a data matrix with dimension $n \times m$ and rank $r$, then the QR decomposition of $X$ is defined as

$$X = QR.$$  \hspace{1cm} \text{Eq. 4-2}

Here, $R$ is an $r \times m$ upper triangular matrix and $Q$ is an $n \times r$ column wise orthonormal matrix. The purpose of QRcp decomposition is to pivot the feature vectors in successive orthogonal directions based on the decreasing order of the maximum Euclidean norm. The feature vector $f_1$ of $X$ is repositioned with feature $f_i$ which has maximum $f_i^T f_i$.

The unit vector $u_1$ in the direction of $f_1$ is defined as follows:

$$u_1 = \frac{f_1}{\|f_1\|_F}.$$  \hspace{1cm} \text{Eq. 4-3}

Next, $f_2$ is swapped with another $f_j$ which maximizes $(f_2 - u_1^T f_2 u_1)^T (f_2 - u_1^T f_2 u_1)$. The unit vector $u_2$ in the direction of $f_2$ is computed as follows:

$$u_2 = \frac{f_2 - (u_1^T f_1) u_1}{\|f_2 - (u_1^T f_1) u_1\|_F}.$$  \hspace{1cm} \text{Eq. 4-4}

On the $i^{th}$ successive selection, the rotated feature vector $f_j^*$ can be defined as

$$f_j^* = f_j - (u_1^T f_1 u_1 + \cdots + u_{i-1}^T f_{i-1} u_{i-1}).$$  \hspace{1cm} \text{Eq. 4-5}
where $i = 2, \ldots, n$ and $j = i, \ldots, n$ and the $i^{th}$ selected vector is the one maximizing $(f_j^*)^T f_j^*$. The successive selection is continued until the factorization reached following stopping criteria:

$$\text{cond}(R_{11}^i) \leq \frac{1}{\epsilon} \leq \text{cond}(R_{11}^{i+1}). \quad \text{Eq. 4-6}$$

Here, $\text{cond}(R_{11}^i)$ refers to the condition number of the upper-triangular matrix with positive diagonal entries. We computed the condition number of the data matrix of the network using Incremental Condition Estimator (ICE) [100]. The resultant permutation matrix $P$ registering the order of successive selections is given by

$$Q^T XP = R. \quad \text{Eq. 4-7}$$

where $R$ is an upper triangular matrix and the value of $i$ at which the factorization stops reveals the rank $r$ of matrix $X$. We performed first level matrix sketching using QR decomposition where we found the actual rank $r$ of matrix $X$ and list of features which are independent with each other as

$$QR(X_m) \rightarrow X_r. \quad \text{Eq. 4-8}$$

Singular Value Decomposition (SVD): Any $n \times m$ matrix $X$ can be uniquely expressed as:

$$X^T = U\Sigma V^T. \quad \text{Eq. 4-9}$$

where $U$ is a column-orthonormal $r \times r$ matrix, $\Sigma$ is a diagonal $r \times n$ matrix with the singular values $\sigma_i$ are sorted in descending order, $r$ is the actual rank of the matrix $X$, which refers to the number of linearly independent columns or features and $V$ is a column-orthonormal $n \times n$ matrix. For our experiment, we calculated the value of the actual rank, $r$, from QR decomposition rather than set it as an input parameter.
Regularized Regression to Select Important Features: The objective of the proposed unsupervised feature selection is to capture the essential characteristics of the dataset without losing too much information. We assume that the network traffic, denoted by \( X_t \in \mathbb{R}^{n \times m}, t = 1, 2, \ldots \) with sliding window size, \( n \), and feature dimension, \( m \), arrive at the timestamp, \( t \), in streams. We performed first level matrix sketching by applying QRcp on \( X_t \) to find the rank, \( r \). Therefore, the shrunken data matrix in the current network window is defined by \( \{ X_{t(r)} \in \mathbb{R}^{n \times r}, t = 1, 2, \ldots \} \). We normalize each column or feature vector of \( \{ X_{t(r)} \} \) using Frobenius norm. Next, we applied SVD on normalized \( X_t \) according to “(9)” as follows:

\[
X_t^T = U_t \Sigma_t V_t^T, \quad \text{Eq. 4-10}
\]

where \( X_{t(r)} \) forms the cosine affinity matrix of \( X_{t(r)} \) due to column-wise Frobenius-normalization. We adapted the concept mentioned in [92] by using \( V_{t(r)}^T \) as the target variable in the regression. We formulate the resulting regression problem as

\[
\min_A \| X_{t(r)} A - V_{t(r)} \|_F^2. \quad \text{Eq. 4-11}
\]

where each column in \( A \) refers to the combination coefficient for different features in approximating the right singular vectors \( V_{t(r)} \) of \( U_t \Sigma_t V_t^T \). Next, we defined another parameter, \( k \), which can be used to set the number of target right singular vectors of reduced feature space such that maximum information will be retained keeping the information loss as minimal as possible. To calculate \( k \), we adopted the criterion called the percentage of energy explained by singular values defined as:
\[ P_{ex} = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}. \]  

Eq. 4-12

We set \( P_{ex} = 95\% \) to calculate the value of \( k \) dynamically instead of using a static value of \( k \), as seen in [91, 92]. By considering a rank-\( k \) approximation of \( X_t \):

\[ X_{t(k)} = U_{t(k)} \sum_{i=1}^{k} \sigma_i^2 V_{t(k)}^T. \]  

Eq. 4-13

Here, \( k \) refers to the low rank of matrix \( X_t \), considering the low-rank approximation of \( X_t \), the least square regression problem can be redefined using \( V_{t(k)} \) as the regression target as follows:

\[ \min_A \| X_t A - V_{t(k)} \|_F^2. \]  

Eq. 4-14

However, the problem of applying a plain Eq. 4-14, particularly for the case of an ill-conditioned matrix, may result in an unstable solution or an overfit solution. To overcome this problem, a regularization term has been added. We formed feature importance coefficient matrix \( A_t = (a_{t,i,j}) \), where \( 1 \leq i \leq r \) and \( 1 \leq j \leq n \); however, only the first \( r \) columns in \( A \) are of importance:

\[ \min_{A_t} \| X_t A_t - V_{t(k)} \|_F^2 + \beta \| a_{t,i,j} \|_p^p. \]  

Eq. 4-15

where \( \beta \) is the regularization parameter. The purpose of using \( \beta \) is to control the magnitude and sparsity of the coefficients with larger values of \( \beta \) being both smaller and sparser. \( \beta \) also determines how much regularization or loss is proportioned.

Generally, a regression formulation with \( L_1 \)-norm \( (p = 1) \) and \( L_2 \)-norm \( (p = 2) \) regularization is referred to as Lasso and ridge regression, respectively. For our experiments, we used Lasso regularization because it produces sparse coefficients which can minimize the effect of overfitting problem, as well as non-zero coefficients, can be
useful to select important features. The feature importance vector \( w_t = \{w_{1(t)}, w_{2(t)}, ..., w_{r(t)}\} \in \mathbb{R}^r \) is defined by

\[
i \in \mathbb{Z}, 0 < i \leq r, w_{i(t)} = \max_j |a_{t,i,j}|.
\]

Next, we sorted \( w_t \) in decreasing order, interpreting that the feature with higher score carried more importance. Finally, we chose the top \( h \) ranked features which captured the cumulative feature importance as follows:

\[
W_{ex} = \frac{\sum_{i=1}^{h} w_{t,i}^2}{\sum_{i=1}^{m} w_{t,i}^2}.
\]

We set \( W_{ex} = 95\% \) to calculate the value of \( h \) dynamically instead of using a static value, as seen in [91, 92]. We applied this feature selection process both on reference window and current window, but the presence of malicious traffic instances in the current window might lead to a different subset of features than the reference window, and thus we took the superset of selected features from both windows for the final feature selection.

4.4.2 Relative Density Ratio Estimation and Defining Safety Score

In [51], the direct density ratio of two data distribution from reference window \( P_{ref}(x) \) and current window \( P_{cur}(x) \) is defined as follows, where \( x \) is a traffic instance:

\[
r(x) = \frac{P_{ref}(x)}{P_{cur}(x)}.
\]

where, \( x \) refers to a single data instance. The fundamental problem of Eq. 4-18 occurs when the denominator density, \( P_{cur} \), takes small values concerning numerator density, \( P_{ref} \), the density-ratio \( r(x) \) tends to take large values and therefore the overall convergence speed becomes slow. To mitigate this problem, an alternative approach, known as \( \alpha-\)
relative density-ratio of $P_{test}$ and $P_{train}$ has been proposed in [95] to compare the probability density of two data distribution as follows:

$$r_\alpha(x) = \frac{P_{ref}(x)}{\alpha P_{ref}(x) + (1 - \alpha)P_{cur}(x)}, 0 \leq \alpha \leq 1.$$ \hspace{1cm} \text{Eq. 4-19}

Here, $\alpha$ is a tuning parameter for controlling the adaptiveness to the current distribution. To simplify the denominator term in \textbf{Eq. 4-19}, another term $q_\alpha(x)$, $\alpha$-mixture density is given by

$$q_\alpha(x) = \alpha P_{ref}(x) + (1 - \alpha)P_{cur}(x), 0 \leq \alpha \leq 1.$$ \hspace{1cm} \text{Eq. 4-20}

Hence, we can rewrite \textbf{Eq. 4-19} as follows:

$$r_\alpha(x) = \frac{P_{ref}(x)}{q_\alpha(x)}, 0 \leq \alpha \leq 1.$$ \hspace{1cm} \text{Eq. 4-21}

We used $r_\alpha(x)$ as the safety score, to detect the botnet attack as:

$$\begin{cases} r_\alpha(x) \geq \tau, & x \text{ is benign or normal} \\ r_\alpha(x) < \tau, & x \text{ is malicious} \end{cases}.$$ \hspace{1cm} \text{Eq. 4-22}

where $\tau$ denotes the safety threshold for each network traffic instance. In [95], a Gaussian kernel model, using $K(x, x_l)$ as a kernel basis function, for the true $\alpha$-relative density-ratio, $r_\alpha(x)$, has been given by

$$g(x; \theta) = \sum_{l=1}^{n_{ref}} \theta_l K(x, x_{ref}).$$ \hspace{1cm} \text{Eq. 4-23}

The parameter $\theta = (\theta_1, \theta_2, ..., \theta_n)^T$ in \textbf{Eq. 4-21} in the model, $g(x; \theta)$, can be learned from network traffic samples by minimizing the following expected squared loss between true relative density ratio and estimated relative density ratio, $J(\theta)$:

$$J(\theta) = \frac{1}{2} \mathbb{E}[\{q_\alpha(x)\} \{g(x; \theta) - r_\alpha(x)\}^2].$$ \hspace{1cm} \text{Eq. 4-24}
\[ J(\theta) = \frac{1}{2} E[(q_\alpha(x))[g(x; \theta)]^2 - 2g(x; \theta)r_\alpha(x) + (r_\alpha(x))^2]. \]  \hspace{1cm} \text{Eq. 4-25} \\
\[ J(\theta) = \frac{1}{2} E[(q_\alpha(x))(g(x; \theta))^2 - 2g(x; \theta)r_\alpha(x)] + \text{const}. \]  \hspace{1cm} \text{Eq. 4-26} \\
\[ J(\theta) = \frac{1}{2} E[(\alpha P_{\text{ref}}(x) + (1 - \alpha)P_{\text{cur}})(g(x; \theta))^2 - E[(q_\alpha(x))]g(x; \theta)r_\alpha(x) + \text{const}]. \]  \hspace{1cm} \text{Eq. 4-27} \\
\[ J(\theta) = \frac{\alpha}{2} E[P_{\text{ref}}(x)](g(x; \theta))^2 + -E[(q_\alpha(x))]g(x; \theta) \frac{P_{\text{ref}}(x)}{q_\alpha(x)} + \text{const}. \]  \hspace{1cm} \text{Eq. 4-28} \\
\[ J(\theta) = \frac{\alpha}{2} E[P_{\text{ref}}(x)](g(x; \theta))^2 - E \left( q_\alpha(x) \frac{P_{\text{ref}}(x)}{q_\alpha(x)} \right) g(x; \theta) + \text{const}. \]  \hspace{1cm} \text{Eq. 4-29} \\
\[ J(\theta) = \frac{\alpha}{2} E[P_{\text{ref}}(x)](g(x; \theta))^2 - E[(P_{\text{ref}}(x))]g(x; \theta) + \text{const}. \]  \hspace{1cm} \text{Eq. 4-30} \\

The optimization problem by using the empirical averages to approximate the expectations in \textbf{Eq. 4-22} is given by

\[ \hat{\theta} = \arg\min_{\theta \in \mathbb{R}^n} \left[ \frac{1}{2} \theta^T \tilde{H} \theta - \tilde{h}^T \theta + \frac{\lambda}{2} \theta^T \theta \right]. \]  \hspace{1cm} \text{Eq. 4-31} \\

where \( \lambda (\lambda \geq 0) \) is the regularization parameter and \( \frac{\lambda}{2} \theta^T \theta \) is a penalty term for regularization purposes. \( \tilde{H} \) a \( n \times n \) matrix and \( \tilde{h} \) an \( n \) column vector (where \( n \) is the number of data points in the reference window) are defined as

\[ \tilde{H}_{\text{ref},\text{ref}} = \frac{\alpha}{n_{\text{ref}}} \sum_{i=1}^{n_{\text{ref}}} K(x_{i_{\text{ref}}, x_{i_{\text{ref}}}})K(x_{i_{\text{ref}}, x_{i_{\text{ref}}}}) + \frac{1-\alpha}{n_{\text{cur}}} \sum_{j=1}^{n_{\text{cur}}} K(x_{j_{\text{cur}}, x_{j_{\text{ref}}}})K(x_{j_{\text{cur}}, x_{j_{\text{ref}}}}). \]  \hspace{1cm} \text{Eq. 4-32} \\
\[ \tilde{h}_{i_{\text{ref}}} = \frac{1}{n_{\text{ref}}} \sum_{i=1}^{n_{\text{ref}}} K(x_{i_{\text{ref}}, x_{i_{\text{ref}}}}). \]  \hspace{1cm} \text{Eq. 4-33} \\

In [95], the following analytical solution for “(23)” was proposed and proved:

\[ \hat{\theta} = (\tilde{H} + \lambda I_n)^{-1} \tilde{h}. \]  \hspace{1cm} \text{Eq. 4-34} \\

where \( I_n \) denotes the \( n \)-dimensional identity matrix. For our experiment, we used the following two sets of candidate parameters to determine the best values for \( \sigma \) and \( \lambda \) as proposed in [49]:
\[ \hat{\theta} = (\hat{H} + \lambda I_n)^{-1} \hat{h}. \]  \textbf{Eq. 4-35}

where \( \sigma = 0.6d_{med}, 0.8d_{med}, d_{med}, 1.2d_{med}, \) and \( 1.4d_{med}. \) Here, \( d_{med} \) presents the median distance between samples of \( W_{ref} \) and \( W_{cur}. \) In kernel methods, one of the popular heuristics is to use the median value of the distances between samples for kernel width \( \sigma. \)

The best combination of \( \sigma \) and \( \lambda \) is chosen by grid search through 5-fold cross-validation.

Finally, a density-ratio estimator is given as

\[ \hat{r}_\alpha(x) = g(x; \hat{\theta}) = \sum_{l=1}^{n_{ref}} \hat{\theta}_l K(x, x_{t_{ref}}). \]  \textbf{Eq. 4-36}

This mathematical formulation is known as Relative Unconstrained Least Square Importance Fitting (RuLSIF). True \( \alpha \)-relative density-ratio \( r_\alpha(x) \) described in \textbf{Eq. 4-36} is modeled by \( \hat{r}_\alpha(x) \) in \textbf{Eq. 4-35} which has been used to compute the safety score to decide if a network traffic instance in the current window is meant to be categorized into a benign or malignant group.

\subsection*{4.4.3 \textit{k}-Fold Dynamic Threshold}

We computed the value of the threshold, \( \tau, \) using the instances of training or reference window \( W_{ref} \) dynamically instead of using any static value. Given \( n \) number of benign instances belong to \( W_{ref}, \) we created \( k \) number of subsets \( s_1, s_2, \ldots, s_k \) with a sample size \( \frac{2}{3}n \) for each subset. At first, we measure the safety score (SSC) of all benign instances of each subset \( s_k \) using the RuLSIF method according to \textbf{Eq. 4-21} by considering one single subset as the current window and the rest \( (k - 1) \) subsets as a reference window. For subset \( s_k, \) the median safety score (MDSC), mean safety score (MNSC), \( \tau_{min} \) are defined as follows:
\[
SSC(s_k(x_i)) = (\mathbf{r}_\alpha(x_i))_{s_k}, i = 1, 2, \ldots, \frac{2}{3}n. \quad \text{Eq. 4-37}
\]

\[
MDSC(s_k) = \text{median}((\mathbf{r}_\alpha(x_i))_{s_k}, i = 1, 2, \ldots, \frac{2}{3}n. \quad \text{Eq. 4-38}
\]

\[
MNSC(s_k) = \text{mean}((\mathbf{r}_\alpha(x_i))_{s_k}, i = 1, 2, \ldots, \frac{2}{3}n. \quad \text{Eq. 4-39}
\]

\[
\tau_{\text{min}}(s_k) = \min(MDSC(s_k), MNSC(s_k)). \quad \text{Eq. 4-40}
\]

Next, we created a sequence of 2k evenly spaced numbers \(\tau_1, \tau_2, \ldots, \tau_{2k}, 0 < \tau \leq \tau_{\text{min}}(s_k)\). We categorized each instance of subset \(s_k\) as a benign or malignant category using \(\tau_i\) and then measured the performance of categorizing traffic instances as benign or malignant in terms of True Positive Rate (TPR). We sorted \(\tau_1, \tau_2, \ldots, \tau_{2k}\) in decreasing order and selected the first quartile value from the ordered \(\tau\)-list as threshold \(\tau_{s_k}\) for subset \(s_k\). We followed the same steps for the remaining subsets. For our experiments, we set \(k\) equal to 5. Finally, we calculated \(\tau\) for \(W_{\text{ref}}\) with size \(n\) as follows:

\[
\tau_{W_{\text{ref}}} = \frac{1}{k} \sum_{i=1}^{k} \tau_{s_k}. \quad \text{Eq. 4-41}
\]

### 4.4.4 Algorithm of Proposed Framework

The prototype algorithm for the proposed framework is in Algorithm 1.

**Table 4-1: ALGORITHM 1: framework IBDS.**

<table>
<thead>
<tr>
<th>ALGORITHM 1: framework IBDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1:</strong> Procedure IBDS</td>
</tr>
<tr>
<td>Parameters: Value of (\alpha) and fold (k)</td>
</tr>
<tr>
<td><strong>Input:</strong> Reference or training window (W_{\text{ref}}), current window (W_{\text{cur}}), and (\alpha)</td>
</tr>
</tbody>
</table>
**Table 4-1: ALGORITHM 1: framework IBDS.**

<table>
<thead>
<tr>
<th>ALGORITHM 1: framework IBDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output:</strong> Safety score and response category (benign or malignant) of all traffic instances in the current window</td>
</tr>
<tr>
<td>2: ( \tau \leftarrow \text{CalculateThreshold}(W_{\text{ref}}, k) )</td>
</tr>
<tr>
<td>3: ( f_{\text{ref}} \leftarrow \text{CalculateImportantFeature}(W_{\text{ref}}, 1) )</td>
</tr>
<tr>
<td>4: ( f_{\text{cur}} \leftarrow \text{CalculateImportantFeature}(W_{\text{cur}}, 1) )</td>
</tr>
<tr>
<td>5: ( f \leftarrow \text{superset of } f_{\text{ref}} \text{ and } f_{\text{cur}} % \text{Safety Score } \text{SSC of } x = r_{a}(x) )</td>
</tr>
<tr>
<td>6: ( \text{SSC}(W_{\text{cur}}(x)) \leftarrow \text{RuLSIF}(W_{\text{ref}}(f), W_{\text{cur}}(f)) )</td>
</tr>
<tr>
<td>7: <strong>If</strong> ( \text{SSC}(W_{\text{cur}}(x)) ) is greater than or equal to ( \tau ) <strong>then</strong></td>
</tr>
<tr>
<td>8: traffic instance ( x ) is benign or normal</td>
</tr>
<tr>
<td>9: <strong>Else</strong></td>
</tr>
<tr>
<td>10: traffic instance ( x ) is malignant and is launched from a botnet</td>
</tr>
<tr>
<td>11: <strong>End If</strong></td>
</tr>
<tr>
<td>12: <strong>end procedure</strong></td>
</tr>
<tr>
<td>13: <strong>Procedure CalculateImportantFeature</strong> ( (W, p) )</td>
</tr>
<tr>
<td><strong>Input:</strong> Reference or training window ( W ) and ( p = 1 ) for Lasso regularization or ( p = 2 ) for ridge regularization</td>
</tr>
<tr>
<td><strong>Output:</strong> feature list ( f ) with higher feature importance coefficient</td>
</tr>
<tr>
<td>14: Apply Frobenius normalization on ( W )</td>
</tr>
<tr>
<td>15: ( W' \leftarrow \text{Apply QRcp on } W ) to find actual rank ( r ) and list of independent features of ( W )</td>
</tr>
<tr>
<td>16: ( U_{t(r)}\Sigma_{t(r)} V_{t(r)}^{T} \leftarrow \text{SVD } (W') )</td>
</tr>
</tbody>
</table>
Table 4-1: Algorithm 1: framework IBDS.

Algorithm 1: framework IBDS

17: Find the value of $k$ using 95% cumulative variance

18: Solve for $A_t$ the feature importance coefficient matrix

19: $j \in 1, 2, 3, ..., r - 1, r$; $w_{i(t)} = \max_j |a_{tij}|$

20: Find feature list $f$ with top $h$ features using 95% cumulative feature importance or feature weight

21: return $f$

22: end procedure

23: Procedure CalculateThreshold($W_{ref}, k$)

Input: Reference or training window $W_{ref}$ with size $n$, the number of folds or subsets $k$

Output: threshold $\tau$ for window $W$

24: Create $k$ number of subsets $s_1, s_2, ..., s_k$ with sample size $\frac{2}{3}n$ from $W_{ref}$.

25: For each $s_k$ do

26: $SSC(s_k(x)) \leftarrow$ RuLSIF($W_{ref} - s_k, s_k$)

27: $MDSC(s_k) \leftarrow median(\{\hat{r}_n(x_i)\}_{s_k})$

28: $MNSC(s_k) \leftarrow mean(\{\hat{r}_n(x_i)\}_{s_k})$

29: $\tau_{\text{min}}(s_k) = \min(\text{MDSC}(s_k), \text{MNSC}(s_k))$

30: Create a sequence of $2k$ number $\tau_1, \tau_2, ..., \tau_{2k}, 0 < \tau \leq \tau_{\text{min}}(s_k)$
**Table 4-1:** ALGORITHM 1: framework IBDS.

<table>
<thead>
<tr>
<th>ALGORITHM 1: framework IBDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>31: <strong>For</strong> each $\tau_i$ <strong>do</strong></td>
</tr>
<tr>
<td>32: Calculate TPR $s_k$ using $\tau_i$</td>
</tr>
<tr>
<td>33: Sort $\tau_1, \tau_2, \ldots, \tau_{2k}$, based on TPR</td>
</tr>
<tr>
<td>34: Select first quartile value of sorted $\tau$ list as threshold $\tau_{s_k}$</td>
</tr>
<tr>
<td>35: <strong>End For</strong></td>
</tr>
<tr>
<td>36: $\tau_{w_{ref}} \leftarrow average(\tau_{s_k}), for all k$</td>
</tr>
<tr>
<td>37: return $\tau_{w_{ref}}$</td>
</tr>
<tr>
<td>38: <strong>end procedure</strong></td>
</tr>
</tbody>
</table>

The pseudo-code of RuLSIF are available in [95].

### 4.5 Results

In this section, we experimentally evaluated our proposed IBDS framework by using benchmark datasets. The dataset IoT_botnet_attacks_N_BaIoT, publicly available on UCI machine repository [67, 99], addresses the lack of public botnet datasets, especially for the IoT devices. The dataset provides 7,062,6069 (7M approx.) network traffic data instances with 115 variables or features in the form of network traffic statistics from 9 IoT devices which are Danmini Doorbell (D1), Ecobee Thermostat (D2), Ennio Doorbell (D3), Philips B120N10 Baby Monitor (D4), Provision PT 737E Security Camera (D5), Provision PT 838 Security Camera (D6), Samsung SNH 1011 N Webcam (D7), SimpleHome XCS7 1002 WHT Security Camera (D8), and SimpleHome XCS7 1003 WHT Security Camera (D9). Summary statistics of the recent traffic from the packet's host IP and MAC addresses, the weight of the network traffic stream, the root squared sum of the two streams' mean,
variances, approximated covariance, approximated correlation coefficient, etc. are some examples of the 115 features. This dataset enables empirical evaluation with real traffic data, collected from nine commercial IoT devices compromised by two of the most common and harmful botnets, Mirai and BASHLITE in an isolated network. The dataset was donated by the Department of Software and Information Systems Engineering, at the Ben-Gurion University of the Negev in Israel.

For our experiments, we considered the task of finding malicious network traffic generated from compromised IoT in a sliding network data window based on a reference or training window which only included benign or normal traffic instances. In real life, a relatively small number of malicious network traffic activities are scattered with a huge number of normal traffic activities which makes the network data in the current window highly imbalanced. To test the correctness, consistency and scalability of proposed system, we used three different window size which are 5,000, 15,000 and 25,000 for all devices except D2 and D9 and infused different fraction rate, an example, 2%, 5%, 10% of randomly chosen malignant data with benign data in current window such that the malignant data instances acted like minority classes in imbalanced or skewed datasets. Since the number of records is much smaller for D2 and D9, we set the window size equal to 2,000, 5,000, 10,000 for D2 and 5,000, 10,000, 15,000 for D9.

To test each scenario (for example, device D1, window size 5000, anomaly rate 2%), we ran 10 trials with different datasets. Therefore, for nine IoT devices, we prepared a total of 810 datasets to conduct 5,670 number of experiments. We compared our proposed method with four other baseline methods, OSVM, LOF, ISF, and Deep Autoencoder which are widely used for the anomaly detection system.
Table 4-2 to Table 4-10 presented the comparison of performance in terms of F1-score and runtime (in the sec) of proposed IBDS framework with OSVM, LOF, ISF for nine IoT, whereas Table 4-12 to Table 4-14 demonstrated the performance of Deep Autoencoder. Hyperparameters used for Deep Autoencoder are presented in Table 4-11. We highlighted the cases with red font where baseline methods performed better than IBDS. For our experiments, Keras has been used to implement Autoencoder with same hyperparameters and thresholds (see Table 4-11) used in [67] to compare our results, but fixed thresholds (\(tr\)) showed lower performance; hence, we applied grid search on reconstruction error to find optimal threshold \(k\) to improve the performance of Deep Autoencoder. We used F1-score using both \(tr\) [67] and \(k\) in Table 4-12 to Table 4-14.

We also conducted pairwise t-test on proposed IBDS framework with OSVM, LOF, ISF, and Deep Autoencoder using average F1 score and average runtime based on anomaly ratio at 95% confidence level to compare the overall performance. The pairwise t-test results shown in Table 4-15 to Table 4-16 also demonstrated that IBDS framework performed much better than baseline methods for most of IoT devices.

Figure 4-1 illustrated the comparison of performance in terms of average F1-score of proposed IBDS framework with OSVM, LOF, ISF, and Deep Autoencoder for device 9 IoT devices. In

Figure 4-2, we compared the runtime performance in terms of runtime ratio of proposed IBDS framework with OSVM, LOF, ISF, and Deep Autoencoder for device 9 IoT devices. Run time ratio is defined as

\[
\text{Run time ratio, } RTR(x) = \frac{\text{Average run time of method } x}{\text{Average runtime of proposed IBDS}}. \quad \text{Eq. 4-42}
\]
where \(x = \text{Deep Autoencoder, OSVM, LOF, ISF, IBDS}\). Using Eq. 4-42, we obtained the RTR of proposed IBDS framework equal to be 1. Therefore, we could compare the relative runtime of baseline methods concerning the runtime of the proposed method.

**Figure 4-3(a),** we illustrated the run time ratio of proposed IBDS with OSVM, LOF, ISF, Deep Autoencoder with respect to window size. In **Figure 4-3(b),** we compared the average runtime (in a sec) of proposed IBDS with OSVM, and Deep Autoencoder concerning window size (2000, 5000, 10000, 150000, 25000).

**Table 4-2:** Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D1.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>Proposed Framework (Unsupervised Feature Selection + RuLSIF)</th>
<th>OSVM</th>
<th>LOF</th>
<th>ISF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>alpha=0</td>
<td>alpha=0.5</td>
<td>alpha=0.95</td>
<td>F1</td>
<td>Run time (sec)</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>87.9</td>
<td>8.4</td>
<td>88.2</td>
<td>8.6</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>83.9</td>
<td>8.4</td>
<td>85.3</td>
<td>8.7</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>84.1</td>
<td>8.4</td>
<td>84.4</td>
<td>8.4</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>91.3</td>
<td>26.2</td>
<td>92.7</td>
<td>26.1</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>86.8</td>
<td>26.3</td>
<td>81.5</td>
<td>26.2</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>81.2</td>
<td>26.4</td>
<td>76.4</td>
<td>26.3</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>89.2</td>
<td>45.6</td>
<td>87.2</td>
<td>46.4</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>87.6</td>
<td>44.2</td>
<td>83.1</td>
<td>44.5</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>78.1</td>
<td>44.4</td>
<td>80.3</td>
<td>44.4</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>85.6</td>
<td>26.5</td>
<td>84.3</td>
<td>26.6</td>
</tr>
</tbody>
</table>
Table 4-3: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D2.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>Proposed Framework (Unsupervised Feature Selection + RuLSIF)</th>
<th>OSVM</th>
<th>LOF</th>
<th>ISF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alpha=0</td>
<td>alpha=0.5</td>
<td>alpha=0.95</td>
<td>Kernel Width=0.1</td>
</tr>
<tr>
<td>2000</td>
<td>2</td>
<td>59.2</td>
<td>3.2</td>
<td>62.0</td>
<td>3.3</td>
</tr>
<tr>
<td>2000</td>
<td>5</td>
<td>54.1</td>
<td>3.3</td>
<td>54.3</td>
<td>3.3</td>
</tr>
<tr>
<td>2000</td>
<td>10</td>
<td>54.8</td>
<td>3.3</td>
<td>49.6</td>
<td>3.2</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>67.8</td>
<td>10.0</td>
<td>71.1</td>
<td>10.0</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>58.5</td>
<td>9.8</td>
<td>61.5</td>
<td>9.8</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>52.5</td>
<td>10.1</td>
<td>55.0</td>
<td>10.4</td>
</tr>
<tr>
<td>10000</td>
<td>2</td>
<td>76.9</td>
<td>26.0</td>
<td>78.3</td>
<td>25.6</td>
</tr>
<tr>
<td>10000</td>
<td>5</td>
<td>76.3</td>
<td>24.0</td>
<td>75.9</td>
<td>23.3</td>
</tr>
<tr>
<td>10000</td>
<td>10</td>
<td>68.8</td>
<td>25.1</td>
<td>66.3</td>
<td>24.6</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>63.2</td>
<td>12.7</td>
<td>63.8</td>
<td>12.6</td>
</tr>
</tbody>
</table>

Table 4-4: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D3.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>Proposed Framework (Unsupervised Feature Selection + RuLSIF)</th>
<th>OSVM</th>
<th>LOF</th>
<th>ISF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alpha=0</td>
<td>alpha=0.5</td>
<td>alpha=0.95</td>
<td>Kernel Width=0.1</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>84.1</td>
<td>8.7</td>
<td>83.0</td>
<td>8.4</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>80.1</td>
<td>8.2</td>
<td>78.8</td>
<td>8.7</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>76.1</td>
<td>8.2</td>
<td>79.8</td>
<td>8.2</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>88.2</td>
<td>25.5</td>
<td>82.7</td>
<td>25.4</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>82.1</td>
<td>25.5</td>
<td>82.4</td>
<td>25.7</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>80.0</td>
<td>25.4</td>
<td>80.1</td>
<td>26.1</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>87.5</td>
<td>45.4</td>
<td>86.2</td>
<td>45.2</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>85.1</td>
<td>46.2</td>
<td>84.5</td>
<td>45.9</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>80.0</td>
<td>45.4</td>
<td>81.8</td>
<td>45.1</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>82.6</td>
<td>26.5</td>
<td>82.1</td>
<td>26.5</td>
</tr>
</tbody>
</table>
Table 4-5: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D4.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alpha=0</td>
<td></td>
<td>alpha=0.5</td>
<td></td>
<td>alpha=0.95</td>
<td></td>
<td>k=5</td>
<td></td>
<td># of tree = window size/10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>86.3</td>
<td>8.3</td>
<td>87.8</td>
<td>8.3</td>
<td>85.7</td>
<td>8.9</td>
<td>62.5</td>
<td>6.0</td>
<td>49.9</td>
<td>59.8</td>
<td>54.8</td>
<td>48.4</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>84.2</td>
<td>8.7</td>
<td>84.8</td>
<td>8.5</td>
<td>84.2</td>
<td>9.6</td>
<td>73.3</td>
<td>6.0</td>
<td>49.8</td>
<td>59.1</td>
<td>60.3</td>
<td>47.9</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>83.5</td>
<td>8.5</td>
<td>83.6</td>
<td>8.5</td>
<td>83.4</td>
<td>9.0</td>
<td>82.0</td>
<td>5.7</td>
<td>48.9</td>
<td>59.4</td>
<td>70.1</td>
<td>52.5</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>85.5</td>
<td>27.1</td>
<td>88.3</td>
<td>26.9</td>
<td>81.9</td>
<td>28.3</td>
<td>62.7</td>
<td>72.9</td>
<td>50.0</td>
<td>1315.3</td>
<td>54.8</td>
<td>224.1</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>85.6</td>
<td>27.0</td>
<td>83.7</td>
<td>27.0</td>
<td>85.5</td>
<td>29.3</td>
<td>73.2</td>
<td>70.8</td>
<td>50.1</td>
<td>1255.5</td>
<td>60.3</td>
<td>250.2</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>82.9</td>
<td>27.5</td>
<td>83.0</td>
<td>27.7</td>
<td>84.0</td>
<td>29.1</td>
<td>81.8</td>
<td>67.8</td>
<td>48.0</td>
<td>1201.5</td>
<td>70.1</td>
<td>284.8</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>87.1</td>
<td>47.2</td>
<td>86.4</td>
<td>47.4</td>
<td>86.5</td>
<td>48.4</td>
<td>62.6</td>
<td>206.6</td>
<td>50.4</td>
<td>3585.2</td>
<td>54.8</td>
<td>576.6</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>83.8</td>
<td>48.1</td>
<td>87.4</td>
<td>47.7</td>
<td>85.9</td>
<td>48.6</td>
<td>72.9</td>
<td>201.7</td>
<td>48.8</td>
<td>3643.9</td>
<td>60.3</td>
<td>646.7</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>83.2</td>
<td>47.4</td>
<td>83.6</td>
<td>47.3</td>
<td>75.8</td>
<td>47.9</td>
<td>81.9</td>
<td>192.5</td>
<td>47.5</td>
<td>3503.2</td>
<td>70.1</td>
<td>724.1</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>84.7</td>
<td>27.7</td>
<td>85.4</td>
<td>27.7</td>
<td>83.7</td>
<td>28.8</td>
<td>72.5</td>
<td>92.2</td>
<td>49.3</td>
<td>1631.4</td>
<td>61.8</td>
<td>317.2</td>
</tr>
</tbody>
</table>

Table 4-6: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D5.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alpha=0</td>
<td></td>
<td>alpha=0.5</td>
<td></td>
<td>alpha=0.95</td>
<td></td>
<td>k=5</td>
<td></td>
<td># of tree = window size/10</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>79.3</td>
<td>8.6</td>
<td>79.6</td>
<td>8.4</td>
<td>80.0</td>
<td>8.9</td>
<td>62.4</td>
<td>5.9</td>
<td>50.3</td>
<td>59.6</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>78.3</td>
<td>8.9</td>
<td>75.1</td>
<td>8.7</td>
<td>76.9</td>
<td>9.1</td>
<td>72.7</td>
<td>5.8</td>
<td>49.5</td>
<td>59.3</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>71.5</td>
<td>8.9</td>
<td>72.8</td>
<td>8.8</td>
<td>72.1</td>
<td>9.2</td>
<td>81.4</td>
<td>5.5</td>
<td>50.7</td>
<td>62.9</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>83.3</td>
<td>27.4</td>
<td>83.3</td>
<td>27.3</td>
<td>83.0</td>
<td>29.0</td>
<td>62.4</td>
<td>70.5</td>
<td>50.0</td>
<td>1050.9</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>80.0</td>
<td>27.4</td>
<td>80.8</td>
<td>27.5</td>
<td>80.6</td>
<td>28.5</td>
<td>72.8</td>
<td>68.6</td>
<td>50.3</td>
<td>1083.9</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>72.9</td>
<td>27.1</td>
<td>74.0</td>
<td>27.6</td>
<td>73.3</td>
<td>28.7</td>
<td>81.6</td>
<td>65.4</td>
<td>48.4</td>
<td>1076.1</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>84.5</td>
<td>50.9</td>
<td>80.5</td>
<td>52.3</td>
<td>84.2</td>
<td>54.3</td>
<td>62.5</td>
<td>201.1</td>
<td>50.3</td>
<td>3187.0</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>79.8</td>
<td>49.9</td>
<td>82.5</td>
<td>50.6</td>
<td>83.3</td>
<td>52.7</td>
<td>73.1</td>
<td>196.1</td>
<td>49.8</td>
<td>3476.0</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>76.7</td>
<td>54.5</td>
<td>77.4</td>
<td>54.7</td>
<td>77.9</td>
<td>54.9</td>
<td>81.7</td>
<td>186.6</td>
<td>48.1</td>
<td>3561.1</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>78.5</td>
<td>29.3</td>
<td>78.4</td>
<td>29.6</td>
<td>79.0</td>
<td>30.6</td>
<td>72.3</td>
<td>89.5</td>
<td>49.7</td>
<td>1513.0</td>
</tr>
</tbody>
</table>
Table 4-7: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D6.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>2</td>
<td>66.2</td>
<td>8.6</td>
<td>72.2</td>
<td>9.0</td>
<td>69.4</td>
<td>9.1</td>
<td>62.2</td>
<td>5.9</td>
<td>49.7</td>
<td>59.2</td>
<td>53.0</td>
<td>85.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>68.3</td>
<td>8.5</td>
<td>65.7</td>
<td>8.5</td>
<td>65.0</td>
<td>9.1</td>
<td>72.7</td>
<td>5.7</td>
<td>50.0</td>
<td>66.7</td>
<td>55.4</td>
<td>82.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>59.8</td>
<td>8.5</td>
<td>59.8</td>
<td>8.6</td>
<td>61.3</td>
<td>8.7</td>
<td>81.5</td>
<td>5.5</td>
<td>49.9</td>
<td>67.0</td>
<td>58.2</td>
<td>83.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>79.1</td>
<td>28.0</td>
<td>80.7</td>
<td>28.0</td>
<td>76.0</td>
<td>29.1</td>
<td>62.6</td>
<td>70.4</td>
<td>49.7</td>
<td>800.2</td>
<td>53.1</td>
<td>480.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>71.0</td>
<td>28.0</td>
<td>68.6</td>
<td>27.7</td>
<td>73.8</td>
<td>29.0</td>
<td>73.2</td>
<td>68.5</td>
<td>49.5</td>
<td>803.8</td>
<td>55.5</td>
<td>484.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>63.9</td>
<td>27.6</td>
<td>67.5</td>
<td>27.8</td>
<td>68.7</td>
<td>28.9</td>
<td>81.8</td>
<td>65.2</td>
<td>47.1</td>
<td>916.3</td>
<td>57.8</td>
<td>510.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>81.5</td>
<td>53.4</td>
<td>76.9</td>
<td>52.5</td>
<td>77.6</td>
<td>52.2</td>
<td>62.6</td>
<td>199.7</td>
<td>50.2</td>
<td>3242.8</td>
<td>53.0</td>
<td>1216.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>73.5</td>
<td>52.5</td>
<td>69.7</td>
<td>50.0</td>
<td>75.3</td>
<td>53.4</td>
<td>73.1</td>
<td>195.0</td>
<td>49.1</td>
<td>3498.9</td>
<td>55.8</td>
<td>1251.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>67.0</td>
<td>50.5</td>
<td>67.7</td>
<td>52.2</td>
<td>70.7</td>
<td>52.6</td>
<td>81.9</td>
<td>185.5</td>
<td>47.9</td>
<td>3328.7</td>
<td>58.0</td>
<td>1317.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>70.0</td>
<td>29.5</td>
<td>69.9</td>
<td>29.4</td>
<td>70.9</td>
<td>30.2</td>
<td>72.4</td>
<td>89.0</td>
<td>49.2</td>
<td>1420.4</td>
<td>55.5</td>
<td>612.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-8: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D7.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
<th>F1</th>
<th>Run time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>2</td>
<td>84.7</td>
<td>9.4</td>
<td>85.6</td>
<td>9.5</td>
<td>84.7</td>
<td>9.5</td>
<td>62.4</td>
<td>5.9</td>
<td>51.1</td>
<td>59.3</td>
<td>54.8</td>
<td>83.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>75.5</td>
<td>9.4</td>
<td>73.1</td>
<td>9.6</td>
<td>74.4</td>
<td>9.8</td>
<td>72.9</td>
<td>5.7</td>
<td>50.3</td>
<td>59.8</td>
<td>60.3</td>
<td>84.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>75.6</td>
<td>8.9</td>
<td>73.8</td>
<td>8.9</td>
<td>76.7</td>
<td>9.2</td>
<td>81.8</td>
<td>5.5</td>
<td>48.9</td>
<td>59.7</td>
<td>70.2</td>
<td>89.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>86.4</td>
<td>26.4</td>
<td>89.0</td>
<td>26.6</td>
<td>86.5</td>
<td>27.3</td>
<td>62.6</td>
<td>70.3</td>
<td>50.5</td>
<td>1052.9</td>
<td>54.8</td>
<td>377.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>80.5</td>
<td>26.4</td>
<td>82.1</td>
<td>26.3</td>
<td>80.2</td>
<td>27.9</td>
<td>73.3</td>
<td>68.7</td>
<td>49.2</td>
<td>1007.9</td>
<td>60.3</td>
<td>397.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>79.4</td>
<td>26.4</td>
<td>79.9</td>
<td>26.3</td>
<td>78.4</td>
<td>27.3</td>
<td>81.6</td>
<td>65.7</td>
<td>48.6</td>
<td>935.7</td>
<td>70.1</td>
<td>426.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>84.3</td>
<td>45.8</td>
<td>85.1</td>
<td>45.9</td>
<td>85.4</td>
<td>47.0</td>
<td>62.7</td>
<td>201.3</td>
<td>50.1</td>
<td>3706.2</td>
<td>54.8</td>
<td>884.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>75.9</td>
<td>46.1</td>
<td>80.2</td>
<td>45.8</td>
<td>78.8</td>
<td>46.9</td>
<td>73.1</td>
<td>196.3</td>
<td>49.3</td>
<td>3453.4</td>
<td>60.3</td>
<td>947.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>75.6</td>
<td>46.2</td>
<td>74.7</td>
<td>46.0</td>
<td>75.5</td>
<td>47.8</td>
<td>81.7</td>
<td>187.1</td>
<td>48.9</td>
<td>3482.5</td>
<td>70.1</td>
<td>1018.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>79.8</td>
<td>27.2</td>
<td>80.4</td>
<td>27.2</td>
<td>80.1</td>
<td>28.1</td>
<td>72.4</td>
<td>89.6</td>
<td>49.7</td>
<td>1535.3</td>
<td>61.8</td>
<td>478.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4-9: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D8.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>Proposed Framework (Unsupervised Feature Selection + RuLSIF)</th>
<th>OSVM</th>
<th>LOF</th>
<th>ISF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alpha=0</td>
<td>alpha=0.5</td>
<td>alpha=0.95</td>
<td>Kernel Width=0.1</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>88.1</td>
<td>8.6</td>
<td>88.5</td>
<td>8.7</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>85.9</td>
<td>8.6</td>
<td>85.1</td>
<td>8.5</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>83.5</td>
<td>8.6</td>
<td>84.5</td>
<td>8.5</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>87.7</td>
<td>30.2</td>
<td>87.5</td>
<td>29.4</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>85.6</td>
<td>29.6</td>
<td>85.6</td>
<td>29.7</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>83.6</td>
<td>29.5</td>
<td>84.7</td>
<td>30.9</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>88.6</td>
<td>51.0</td>
<td>90.6</td>
<td>52.1</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>86.4</td>
<td>52.3</td>
<td>85.4</td>
<td>50.9</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>84.5</td>
<td>51.4</td>
<td>84.9</td>
<td>52.1</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>86.0</td>
<td>30.0</td>
<td>86.3</td>
<td>30.1</td>
</tr>
</tbody>
</table>

Table 4-10: Comparison of performance in terms of F1-score and runtime of proposed framework IBDS with OSVM, LOF, and ISF for device D9.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>Proposed Framework (Unsupervised Feature Selection + RuLSIF)</th>
<th>OSVM</th>
<th>LOF</th>
<th>ISF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alpha=0</td>
<td>alpha=0.5</td>
<td>alpha=0.95</td>
<td>Kernel Width=0.1</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>92.6</td>
<td>8.3</td>
<td>91.9</td>
<td>8.2</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>84.9</td>
<td>8.4</td>
<td>83.8</td>
<td>8.2</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>81.1</td>
<td>8.2</td>
<td>81.1</td>
<td>8.2</td>
</tr>
<tr>
<td>10000</td>
<td>2</td>
<td>81.5</td>
<td>17.4</td>
<td>90.5</td>
<td>17.4</td>
</tr>
<tr>
<td>10000</td>
<td>5</td>
<td>85.3</td>
<td>17.7</td>
<td>85.3</td>
<td>17.3</td>
</tr>
<tr>
<td>10000</td>
<td>10</td>
<td>82.0</td>
<td>17.7</td>
<td>81.5</td>
<td>18.0</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>89.0</td>
<td>27.8</td>
<td>86.9</td>
<td>27.9</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>77.9</td>
<td>27.2</td>
<td>81.3</td>
<td>27.7</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>79.4</td>
<td>27.8</td>
<td>82.0</td>
<td>27.3</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>83.7</td>
<td>17.8</td>
<td>84.9</td>
<td>17.8</td>
</tr>
</tbody>
</table>
Table 4-11: Hyperparameter used for deep autoencoder for 9 IoT devices.

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>D8</th>
<th>D9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>0.012</td>
<td>0.003</td>
<td>0.028</td>
<td>0.016</td>
<td>0.026</td>
<td>0.008</td>
<td>0.017</td>
<td>0.006</td>
<td>0.013</td>
</tr>
<tr>
<td>Number of Epochs (epochs)</td>
<td>800</td>
<td>350</td>
<td>250</td>
<td>100</td>
<td>300</td>
<td>450</td>
<td>230</td>
<td>500</td>
<td>150</td>
</tr>
<tr>
<td>Anomaly Threshold (tr)</td>
<td>0.042</td>
<td>0.011</td>
<td>0.011</td>
<td>0.030</td>
<td>0.035</td>
<td>0.038</td>
<td>0.056</td>
<td>0.004</td>
<td>0.074</td>
</tr>
</tbody>
</table>

Table 4-12: Performance of deep autoencoder in terms of F1-score for D1, D3, D4, and D5 IoT devices.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>2</td>
<td>42.4</td>
<td>83.1</td>
<td>33.5</td>
<td>2.1</td>
<td>39.8</td>
<td>44.6</td>
<td>9.8</td>
<td>47.3</td>
<td>53.3</td>
<td>4.0</td>
<td>47.8</td>
<td>37.7</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>65.0</td>
<td>89.8</td>
<td>35.8</td>
<td>4.2</td>
<td>57.8</td>
<td>47.4</td>
<td>18.5</td>
<td>64.3</td>
<td>46.8</td>
<td>9.3</td>
<td>72.6</td>
<td>42.0</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>72.3</td>
<td>95.2</td>
<td>29.6</td>
<td>9.8</td>
<td>79.7</td>
<td>47.7</td>
<td>35.2</td>
<td>80.9</td>
<td>41.3</td>
<td>18.2</td>
<td>83.2</td>
<td>40.1</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>47.1</td>
<td>80.6</td>
<td>59.2</td>
<td>2.0</td>
<td>30.5</td>
<td>80.7</td>
<td>9.9</td>
<td>45.5</td>
<td>71.4</td>
<td>4.0</td>
<td>50.5</td>
<td>51.6</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>72.3</td>
<td>93.0</td>
<td>59.9</td>
<td>5.1</td>
<td>45.9</td>
<td>78.1</td>
<td>25.9</td>
<td>70.0</td>
<td>86.4</td>
<td>10.4</td>
<td>70.8</td>
<td>48.4</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>83.4</td>
<td>95.6</td>
<td>58.1</td>
<td>9.9</td>
<td>55.1</td>
<td>73.4</td>
<td>34.5</td>
<td>82.0</td>
<td>91.0</td>
<td>15.6</td>
<td>79.7</td>
<td>49.0</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>49.6</td>
<td>83.3</td>
<td>109.0</td>
<td>2.1</td>
<td>18.1</td>
<td>115.2</td>
<td>11.0</td>
<td>47.1</td>
<td>139.3</td>
<td>3.3</td>
<td>42.0</td>
<td>68.2</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>76.8</td>
<td>83.4</td>
<td>110.2</td>
<td>5.0</td>
<td>36.9</td>
<td>109.1</td>
<td>23.0</td>
<td>70.4</td>
<td>125.5</td>
<td>8.4</td>
<td>65.6</td>
<td>76.0</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>88.9</td>
<td>96.7</td>
<td>106.0</td>
<td>9.6</td>
<td>51.3</td>
<td>97.2</td>
<td>36.2</td>
<td>82.5</td>
<td>122.3</td>
<td>16.9</td>
<td>78.1</td>
<td>87.8</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>66.4</td>
<td>89.0</td>
<td>66.8</td>
<td>5.5</td>
<td>46.1</td>
<td>77.0</td>
<td>22.7</td>
<td>65.6</td>
<td>86.4</td>
<td>10.0</td>
<td>65.6</td>
<td>55.6</td>
</tr>
</tbody>
</table>

Table 4-13: Performance of Deep Autoencoder in terms of F1-score for D6, D7, and D8 IoT devices.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1 (tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>F1 (tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>2</td>
<td>4.3</td>
<td>70.3</td>
<td>22.4</td>
<td>16.0</td>
<td>62.7</td>
<td>23.2</td>
<td>2.2</td>
<td>47.8</td>
<td>31.9</td>
<td>4.3</td>
<td>68.1</td>
<td>29.3</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>8.6</td>
<td>84.5</td>
<td>22.0</td>
<td>30.9</td>
<td>78.1</td>
<td>25.2</td>
<td>4.3</td>
<td>82.3</td>
<td>24.0</td>
<td>9.6</td>
<td>82.3</td>
<td>24.0</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>18.4</td>
<td>92.3</td>
<td>19.0</td>
<td>48.5</td>
<td>88.9</td>
<td>27.4</td>
<td>9.6</td>
<td>82.3</td>
<td>24.0</td>
<td>2.2</td>
<td>55.3</td>
<td>43.0</td>
</tr>
<tr>
<td>15000</td>
<td>2</td>
<td>4.6</td>
<td>78.3</td>
<td>53.1</td>
<td>14.8</td>
<td>63.7</td>
<td>46.7</td>
<td>2.2</td>
<td>55.3</td>
<td>43.0</td>
<td>5.7</td>
<td>73.4</td>
<td>60.2</td>
</tr>
<tr>
<td>15000</td>
<td>5</td>
<td>12.1</td>
<td>90.4</td>
<td>45.0</td>
<td>39.8</td>
<td>83.1</td>
<td>48.6</td>
<td>5.7</td>
<td>73.4</td>
<td>60.2</td>
<td>5.7</td>
<td>73.4</td>
<td>60.2</td>
</tr>
<tr>
<td>15000</td>
<td>10</td>
<td>22.1</td>
<td>95.3</td>
<td>50.6</td>
<td>60.6</td>
<td>91.8</td>
<td>43.6</td>
<td>10.5</td>
<td>82.3</td>
<td>58.5</td>
<td>10.5</td>
<td>82.3</td>
<td>58.5</td>
</tr>
<tr>
<td>25000</td>
<td>2</td>
<td>4.6</td>
<td>76.9</td>
<td>70.4</td>
<td>13.4</td>
<td>65.5</td>
<td>67.6</td>
<td>2.2</td>
<td>57.1</td>
<td>81.5</td>
<td>2.2</td>
<td>57.1</td>
<td>81.5</td>
</tr>
<tr>
<td>25000</td>
<td>5</td>
<td>12.9</td>
<td>91.2</td>
<td>74.6</td>
<td>30.8</td>
<td>75.0</td>
<td>90.8</td>
<td>5.9</td>
<td>74.3</td>
<td>94.0</td>
<td>5.9</td>
<td>69.1</td>
<td>56.0</td>
</tr>
<tr>
<td>25000</td>
<td>10</td>
<td>22.3</td>
<td>95.8</td>
<td>69.4</td>
<td>58.7</td>
<td>92.2</td>
<td>61.0</td>
<td>10.8</td>
<td>81.1</td>
<td>81.6</td>
<td>10.8</td>
<td>81.1</td>
<td>81.6</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>12.2</td>
<td>86.1</td>
<td>47.4</td>
<td>34.8</td>
<td>77.9</td>
<td>48.2</td>
<td>5.9</td>
<td>69.1</td>
<td>56.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-14: Performance of Deep Autoencoder of F1-score for D2, and D9 IoT devices.

<table>
<thead>
<tr>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
<th>Win Size</th>
<th>Frac (%)</th>
<th>F1 (Fixed tr)</th>
<th>F1 (k)</th>
<th>Run Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>2</td>
<td>2.0</td>
<td>45.5</td>
<td>13.9</td>
<td>5000</td>
<td>2</td>
<td>24.9</td>
<td>49.3</td>
<td>44.6</td>
</tr>
<tr>
<td>2000</td>
<td>5</td>
<td>6.3</td>
<td>75.5</td>
<td>18.2</td>
<td>5000</td>
<td>5</td>
<td>41.9</td>
<td>64.8</td>
<td>47.4</td>
</tr>
<tr>
<td>2000</td>
<td>10</td>
<td>11.6</td>
<td>85.2</td>
<td>16.3</td>
<td>5000</td>
<td>10</td>
<td>62.3</td>
<td>82.0</td>
<td>47.7</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>3.2</td>
<td>58.1</td>
<td>27.8</td>
<td>10000</td>
<td>2</td>
<td>25.2</td>
<td>46.3</td>
<td>80.7</td>
</tr>
<tr>
<td>5000</td>
<td>5</td>
<td>6.5</td>
<td>73.9</td>
<td>24.6</td>
<td>10000</td>
<td>5</td>
<td>48.8</td>
<td>73.2</td>
<td>78.1</td>
</tr>
<tr>
<td>5000</td>
<td>10</td>
<td>12.9</td>
<td>84.6</td>
<td>25.2</td>
<td>10000</td>
<td>10</td>
<td>64.1</td>
<td>85.6</td>
<td>73.4</td>
</tr>
<tr>
<td>10000</td>
<td>2</td>
<td>2.9</td>
<td>59.5</td>
<td>44.4</td>
<td>15000</td>
<td>2</td>
<td>27.3</td>
<td>53.1</td>
<td>115.2</td>
</tr>
<tr>
<td>10000</td>
<td>5</td>
<td>9.1</td>
<td>74.6</td>
<td>52.2</td>
<td>15000</td>
<td>5</td>
<td>48.5</td>
<td>68.1</td>
<td>109.1</td>
</tr>
<tr>
<td>10000</td>
<td>10</td>
<td>16.4</td>
<td>83.2</td>
<td>64.0</td>
<td>15000</td>
<td>10</td>
<td>66.0</td>
<td>84.1</td>
<td>97.2</td>
</tr>
</tbody>
</table>

Average 7.9 71.1 31.9

Average 45.4 67.4 77.0

### Table 4-15: Pairwise t-test results at 95% confidence level using average F1-score of proposed IBDS with OSVM, LOF, ISF, and Autoencoder for 9 IoT devices.

<table>
<thead>
<tr>
<th>Frac</th>
<th>Method 1 (Proposed)</th>
<th>Method 2 (Baseline)</th>
<th>t-value</th>
<th>p-value</th>
<th>Difference of means</th>
<th>Is Significant?</th>
<th>Conclusion (At 95% confidence interval, mean detection rate of IBDS is greater/less than the mean detection rate of method 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>IBDS</td>
<td>OSVM</td>
<td>9.082</td>
<td>1.73E-05</td>
<td>20.988</td>
<td>yes</td>
<td>20.988% greater</td>
</tr>
<tr>
<td>2</td>
<td>IBDS</td>
<td>LOF</td>
<td>14.391</td>
<td>4.84E-07</td>
<td>33.331</td>
<td>yes</td>
<td>33.331% greater</td>
</tr>
<tr>
<td>2</td>
<td>IBDS</td>
<td>ISF</td>
<td>12.527</td>
<td>1.26E-06</td>
<td>29.113</td>
<td>yes</td>
<td>29.113% greater</td>
</tr>
<tr>
<td>2</td>
<td>IBDS</td>
<td>Autoencoder</td>
<td>4.779</td>
<td>0.0005876</td>
<td>27.793</td>
<td>yes</td>
<td>27.793% greater</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>OSVM</td>
<td>2.509</td>
<td>0.03529</td>
<td>6.359</td>
<td>yes</td>
<td>6.359% greater</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>LOF</td>
<td>11.763</td>
<td>2.11E-06</td>
<td>29.648</td>
<td>yes</td>
<td>29.648% greater</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>ISF</td>
<td>7.588</td>
<td>3.06E-05</td>
<td>19.707</td>
<td>yes</td>
<td>19.707% greater</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>Autoencoder</td>
<td>3.110</td>
<td>0.01016</td>
<td>20.047</td>
<td>yes</td>
<td>20.047% greater</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>OSVM</td>
<td>-1.888</td>
<td>0.09546</td>
<td>-5.626</td>
<td>no</td>
<td>There is not enough evidence that the mean detection rates of IBDS and OSVM are different</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>LOF</td>
<td>9.378</td>
<td>9.57E-06</td>
<td>28.268</td>
<td>yes</td>
<td>28.27% greater</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>ISF</td>
<td>2.816</td>
<td>0.01481</td>
<td>9.654</td>
<td>yes</td>
<td>9.65% greater</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>Autoencoder</td>
<td>-1.175</td>
<td>0.2587</td>
<td>-5.856</td>
<td>no</td>
<td>There is not enough evidence that the mean detection rates of IBDS and Autoencoder are different</td>
</tr>
</tbody>
</table>
Table 4-16: Pairwise t-test results 95% confidence level using average run time (in sec) of proposed IBDS with OSVM, LOF, ISF, and Autoencoder for 9 IoT devices.

<table>
<thead>
<tr>
<th>Frac</th>
<th>Method 1 (Proposed)</th>
<th>Method 2 (Baseline)</th>
<th>t-value</th>
<th>p-value</th>
<th>Difference of means</th>
<th>Is Significant?</th>
<th>Conclusion (At 95% confidence interval, mean run time of IBDS is greater/less than the mean run time of method 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>IBDS</td>
<td>OSVM</td>
<td>4.504</td>
<td>0.001705</td>
<td>50.601</td>
<td>yes</td>
<td>50.601 seconds less</td>
</tr>
<tr>
<td>2</td>
<td>IBDS</td>
<td>LOF</td>
<td>6.660</td>
<td>0.0001591</td>
<td>1255.878</td>
<td>yes</td>
<td>1255.878 seconds less</td>
</tr>
<tr>
<td>2</td>
<td>IBDS</td>
<td>ISF</td>
<td>5.794</td>
<td>5.7936</td>
<td>322.351</td>
<td>yes</td>
<td>322.351 seconds</td>
</tr>
<tr>
<td>2</td>
<td>IBDS</td>
<td>Autoencoder</td>
<td>6.326</td>
<td>7.21E-05</td>
<td>34.268</td>
<td>yes</td>
<td>34.268 seconds less</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>OSVM</td>
<td>4.442</td>
<td>0.00183</td>
<td>48.762</td>
<td>yes</td>
<td>48.762 seconds less</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>LOF</td>
<td>6.721</td>
<td>0.0001493</td>
<td>1240.941</td>
<td>yes</td>
<td>1240.941 seconds less</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>ISF</td>
<td>6.121</td>
<td>0.0002799</td>
<td>344.039</td>
<td>yes</td>
<td>344.039 seconds less</td>
</tr>
<tr>
<td>5</td>
<td>IBDS</td>
<td>Autoencoder</td>
<td>6.142</td>
<td>0.0001033</td>
<td>35.927</td>
<td>yes</td>
<td>35.927 seconds less</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>OSVM</td>
<td>4.304</td>
<td>0.002187</td>
<td>45.162</td>
<td>yes</td>
<td>45.162 seconds less</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>LOF</td>
<td>6.724</td>
<td>0.0001489</td>
<td>1216.653</td>
<td>yes</td>
<td>1216.653 seconds less</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>ISF</td>
<td>6.466</td>
<td>0.000193</td>
<td>375.586</td>
<td>yes</td>
<td>375.586 seconds less</td>
</tr>
<tr>
<td>10</td>
<td>IBDS</td>
<td>Autoencoder</td>
<td>5.662</td>
<td>0.000217</td>
<td>35.024</td>
<td>yes</td>
<td>35.024 seconds less</td>
</tr>
</tbody>
</table>

Figure 4-1: Comparison of performance in terms of F1-score of proposed IBDS framework with OSVM, LOF, and ISF for device 9 IoT devices.

Figure 4-2: Comparison of run time ration of proposed IBDS framework with OSVM, LOF, and ISF for device 9 IoT devices.
We particularly focused on these two methods to compare its performance with our proposed IBDS framework which has been illustrated in Figure 4-3(a) and Figure 4-3(b). From experimental results, we observed that the overall performance of the proposed IBDS framework is significantly better than ISF and LOF for all IoT devices concerning both F1-score and runtime.

![Figure 4-3: (a) Comparison of runtime ratio of proposed IBDS with OSVM, LOF, ISF, and Deep Autoencoder concerning window size (b) Comparison of average runtime in seconds of proposed IBDS with OSVM, and deep autoencoder concerning window size.](image)

Figure 4-3: (a) Comparison of runtime ratio of proposed IBDS with OSVM, LOF, ISF, and Deep Autoencoder concerning window size (b) Comparison of average runtime in seconds of proposed IBDS with OSVM, and deep autoencoder concerning window size.

We observed that Deep Autoencoder outperformed IBDS for D1, D2 and D6 whereas OSVM showed better performance for device D2 and slightly better performance for device D6 in terms of F1-score; however, the t-test results in Table 4-16 demonstrated that the performance differences in those cases are insignificant at 95% confidence level. For the other six devices, the proposed IBDS framework performed much better than OSVM and Deep Autoencoder. From Figure 4-3(a), we observed almost same F1-score for 2%, 5% or 10% malignant network traffic by proposed IBDS whereas OSVM and Deep Autoencoder needed a higher ratio to show better performance. Therefore, the proposed framework is better for early detection of botnets when botnets start launching an attack by propagating malicious traffic. Figure 4-3(b) exhibited that runtime of OSVM, Deep
Autoencoder and proposed IBDS are almost the same for a smaller window size (window size equal to 2000, 5000), but with the increase of window size, IBDS performed much faster than OSVM and Autoencoder. Overall, we observed better performance for most of the IoT devices by our proposed technique than OSVM, ISF, LOF, and Deep Autoencoder both in terms of botnet detection rate and computational time.

4.6 Findings and Discussions

In this chapter, we investigated the problem of detecting IoT based botnets using an unsupervised feature model by two-steps matrix sketching followed by a relative density-ratio function with $\alpha$-mixture density to measure the plausibility of a data instance with better performance and efficiency. For performance validation, we compared the proposed IBDS framework with four other baseline methods, OSVM, ISF, LOF, and Deep Autoencoder using F1-score as the performance measure. Overall, we observed better and more consistent performance by the proposed framework compared to OSVM, ISF, LOF, and Deep Autoencoder in terms of improving botnet detection rate and reducing computational time concerning malignant traffic rate, window size and, type of IoT devices which demonstrated its usability and effectiveness. For our experiments, we explored network traffic data from nine IoT devices infected with BASHLITE and Mirai botnets. Our future work includes more improvement of the proposed framework and to apply it on additional IoT devices with new and unknown botnets to explore if we can achieve similar or better results.

In the next chapter, we conclude by summarizing our novel contributions and directions for future research.
CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

Our goal in this work is to develop novel solutions by exploring a variety of feature space modeling in non-stationary data to improve model accuracy and efficiency. We have tested each approach on multiple datasets using appropriate measures of performance and validated them against several base line methods to prove their applicability and effectiveness, and therefore we believe each solution has the potential to be extended to new promising areas in solving problems. Some of the specific contributions and results are as follows:

5.1.1 Relief based feature space modeling to reduce data complexity to improve classification performance

Early prediction of ovarian cancer will reduce its growth rate and may save many lives. Computer aided diagnosis (CAD) is a non-invasive method for finding ovarian cancer in its early stage which can avoid patient anxiety and unnecessary biopsy, but due to the quality of ultrasound image or unexpected error in data preprocessing step, converted data from the images can be very complex, which may lead to the problem of higher class inseparability or overlap resulting into poor classification performance. To our best knowledge, we are the first one to explore Data Complexity Analysis in the area of ovarian
image classification to investigate the inherent complexity of the data, and used Relief
based feature selection method to reduce its complexity, and finally applied Fuzzy
Ensemble classifiers to differentiate between instances of the normal versus the target
classes successfully to improve accuracy and efficiency.

5.1.2 SVD based feature space transformation to build a novel change detection
system with better accuracy and efficiency

In a non-stationary environment, identification of distributional change points over
time, known as concept drift, is very crucial for some applications when input data is
expected to follow the same distribution. Continuous monitoring of different health
activities, for example, patient’s heart rate monitoring, is one of the major application areas.
To the best of our knowledge, we are the first to use the Direct Density Ratio Estimation
in combination with SVD in building an unsupervised data-driven Active Health
Monitoring system to detect significant changes in data distribution of human health
activity data collected from body sensors with higher change detection rate and reduced
computational time.

5.1.3 Unsupervised feature space modeling to build a novel IoT botnet detection
system with better accuracy and efficiency

Botnets have become one of the overarching problems in the domain of IoT security
due to their unparallel popularity among cybercriminals comes from their ability to
infiltrate almost any internet-connected device. To the best of our knowledge, we are the
first to use the Relative Density Ratio Estimation in combination with unsupervised feature
space model in building a data-driven network-based IoT security mechanism using
network traffic statistics to detect malicious attacks launched from compromised IoT
devices with better performance in terms of improving the botnet detection rate and reducing computational time.

5.2 Future Work

We can further explore our proposed Fuzzy Framework on new datasets collected from ovarian cancer images or extend our studies in classifying cancer images of another category like thyroid or breast cancer. The novel solution we proposed to identify change points in health activity sensor data that can be further investigated with other health data, for example, heart rate monitoring data. For our botnet detection experiments, we explored network traffic data from nine IoT devices infected with BASHLITE and Mirai botnets. One possible future avenue is to apply it on additional IoT devices with new and unknown botnets if we can achieve similar or better results.
REFERENCES


[37] M. Rowe, "Mining user development signals for online community churner detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 3, p. 21, 2016.


[59] K. Pearson, "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," The London, Edinburgh, and


