Fall 2019

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

Ayesha Akter
Louisiana Tech University

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

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

SEPTEMBER 16, 2019
Date

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 College

Approved:

Dean of the Graduate School

GS Form 13
(8/10)

ABSTRACT

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

LIST OF FIGURES

LIST OF TABLES

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**.

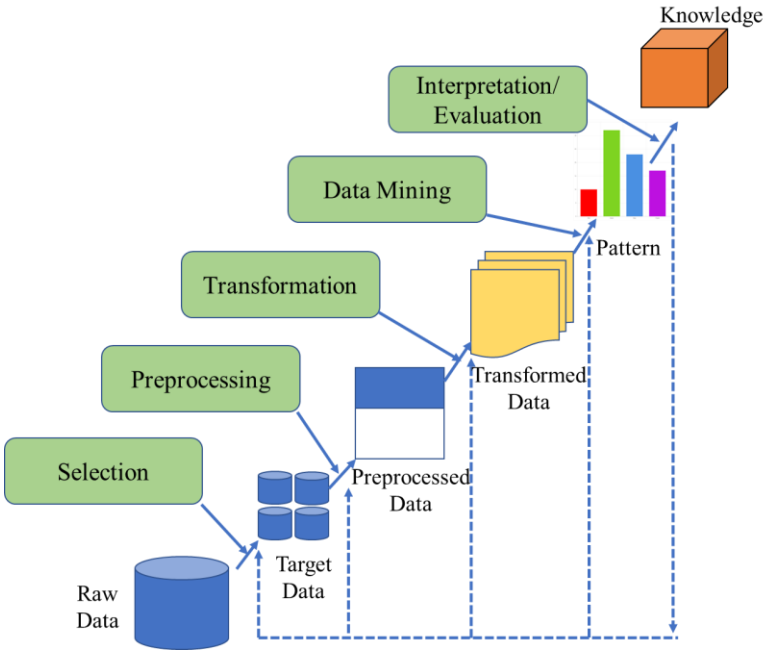

Figure 1-1: The step diagram of the KDD process.

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**.

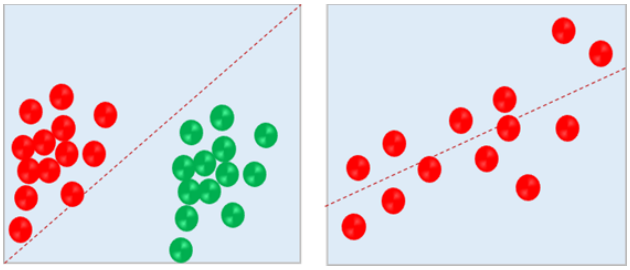

Figure 1-2: (left) classification plot, (right) regression plot

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**.

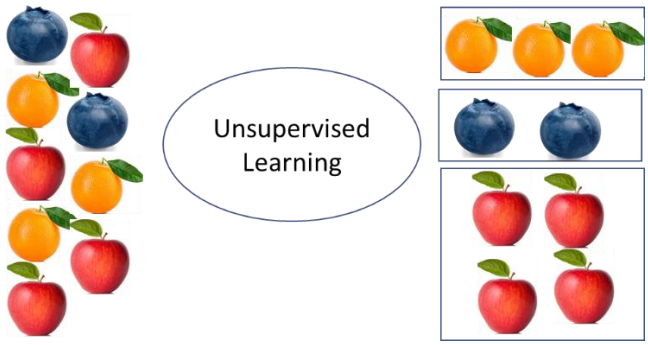

Figure 1-3: Clustering process

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

$$A = QR. \qquad \textbf{Eq. 1-1}$$

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

$$A = Q\Lambda Q^{-1}. \quad \textbf{Eq. 1-2}$$

where A is the square $n \times n$ matrix whose ith column is the eigenvector q_i of A, and Λ 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

$$A = UD_sV^T. \quad \textbf{Eq. 1-3}$$

where U is a column-orthonormal n × 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 × 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:

$$H_0: \mu_d = 0 \quad \textbf{Eq. 1-4}$$

$$H_1: \mu_d \neq 0 \quad \textbf{Eq. 1-5}$$

where μ_d refers to the means of the difference from two selected samples with size n from a dataset at a time.

$$t_{0=\frac{\bar{d}}{S_d/\sqrt{n}}} \quad \textbf{Eq. 1-6}$$

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

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

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

$y_{ij}(i = 1,2)$ 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 $P_{ref}(x)$ and current window $P_{cur}(x)$ is defined as follows, where x is a traffic instance:

$$r(x) = \frac{P_{ref}(x)}{P_{cur}(x)}. \quad \textbf{Eq. 1-10}$$

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

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

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 data set. 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**:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad \textbf{Eq. 1-12}$$

$$Sensitivity = Recall = TPR = \frac{TP}{TP + FN}. \quad \textbf{Eq. 1-13}$$

$$Specificity = TNR = \frac{TN}{TN + FP}. \quad \textbf{Eq. 1-14}$$

$$FPR = 1 - Specificity = \frac{FP}{TN + FP}. \quad \textbf{Eq. 1-15}$$

$$Precision = \frac{TP}{TP + FP} \quad \textbf{Eq. 1-16}$$

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**.

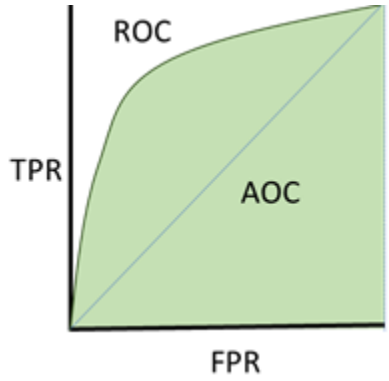

Figure 1-4: AUC-ROC curve

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

$$F1 - score = \frac{2 * Precision}{Precision + Recall}. \quad \textbf{Eq. 1-17}$$

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 $80.60 \pm 0.5\%$ 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.

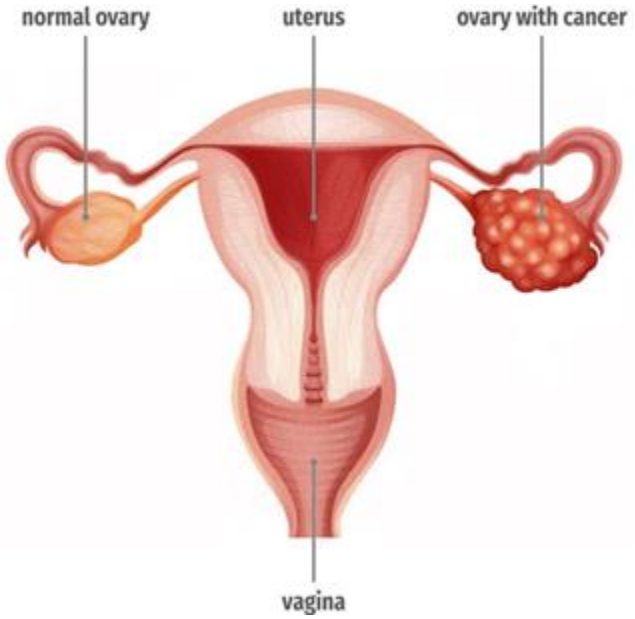

Figure 2-1: A typical illustration of a normal ovary and ovary with cancer.

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

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}. \quad \textbf{Eq. 2-1}$$

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 (σ^2). Variance (σ^2) of each feature x where $1 \leq x \leq 810$ was calculated using

$$mean\ (\mu) = \frac{\sum x}{N}. \quad \textbf{Eq. 2-2}$$

$$variance\ (\sigma^2) = \frac{\sum (x - \mu)^2}{N}. \quad \textbf{Eq. 2-3}$$

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:

$$W_i = W_i - (x - nearHit_i)^2 + (x - nearMiss_i)^2. \quad \textbf{Eq. 2-4}$$

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 τ.

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:

$$f_i = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^{\,2} + \sigma_2^{\,2}}. \qquad \textbf{Eq. 2-5}$$

$$F1 = max(f_i). \qquad \textbf{Eq. 2-6}$$

where f_i refers to an individual feature with $1 \leq i \leq n$ and μ_1, μ_2, σ_1^2, σ_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:

$$f_{efficiency} = \frac{N_{Seperate}}{N}$$
$$f_{efficiency} = \frac{N - N_{overlap}}{N} \qquad \textbf{Eq. 2-7}$$

$$F3 = \max(f_{efficiency}). \qquad \textbf{Eq. 2-8}$$

where *N* is the total number of data instances, $N_{overlap}$ is the number of data instances in the overlap region. So, $(N - N_{overlap})$ 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 (F3).

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:

$$N1 = \frac{N_{boundary}}{N} . \qquad \textbf{Eq. 2-9}$$

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

$$N2 = \frac{(D_{avg})_{IntraClass}}{(D_{avg})_{InterClass}} . \qquad \textbf{Eq. 2-10}$$

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

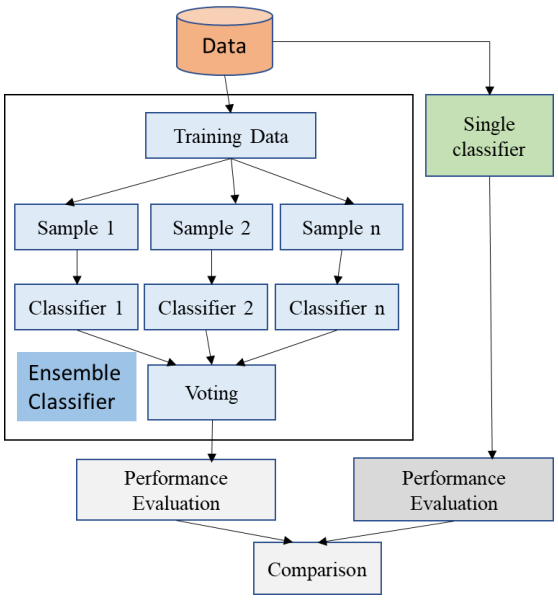

Figure 2-2: Single vs. ensemble classifier

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, $P = \{P_1, P_2, ..., P_m\}$ such that $\sum P_i = P$ using the Weighted Correlation Network Analysis (WGCNA) algorithm [21] as shown in **Figure 2-3(c)**.

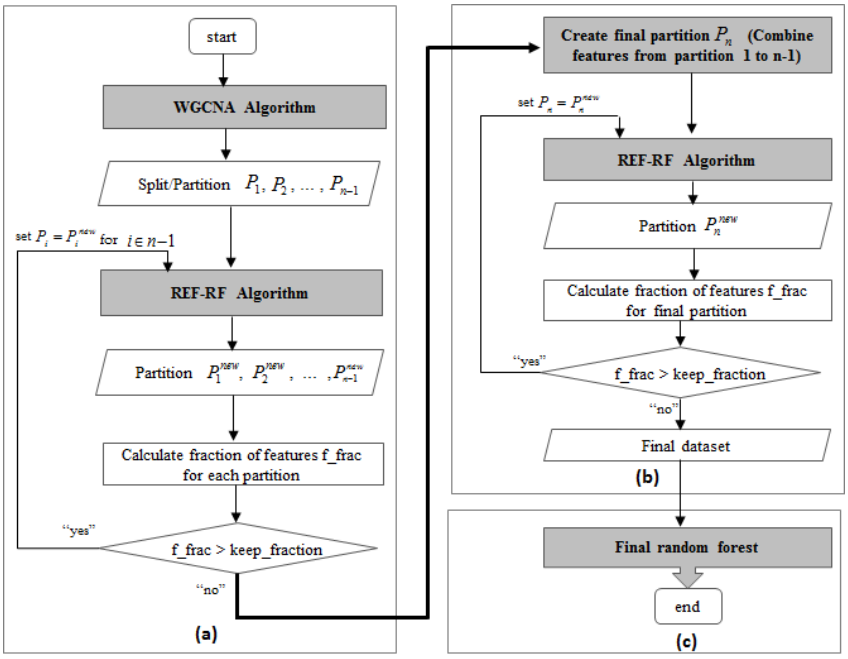

Figure 2-3: Flow diagram for fuzzy forest classifier. (a) screening step (b) selection step (c) final step.

WGCNA is widely used to construct the correlation module or network using correlation as follows:

$$r = \frac{n(\sum xy) - (\sum x \sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}. \qquad \textbf{Eq. 2-11}$$

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:

$$VIM_k(n) = \frac{\sum_{i \in B_k} (Y_i - \hat{f}_k(\hat{X}_i))^2 - (Y_i - \hat{f}_k(X_i))^2}{|B_k|}. \qquad \textbf{Eq. 2-12}$$

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

$$VIM(n) = \frac{\sum_{k=1}^{ntree} VIM_k(n)}{ntree}. \qquad \textbf{Eq. 2-13}$$

where Y_i is the target class for the i^{th} instance X_i, $\hat{X}_i$ is the i^{th} instances in out of bag samples of the k^{th} tree. Similarly, $\hat{f}_k(X_i)$ is the conditional mean, and $\hat{f}_k(\hat{X})$ is the conditional mean $E[Y_i / \hat{X}_i]$, 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 $\Theta(pstmnlog(n))$. Complexity of final random forest algorithm is $\Theta(tfmn\log(n))$. Hence, the average run time complexity of fuzzy forest algorithm is as follows:

$$\Theta(pl^2 n + pstmn\log(n) + tfmn\log(n)). \quad \textbf{Eq. 2-14}$$

$$\Rightarrow \Theta(p\left\lceil \frac{m}{p} \right\rceil^2 n + tmn\log(n)(ps + f)). \quad \textbf{Eq. 2-15}$$

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

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

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

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:

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

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.

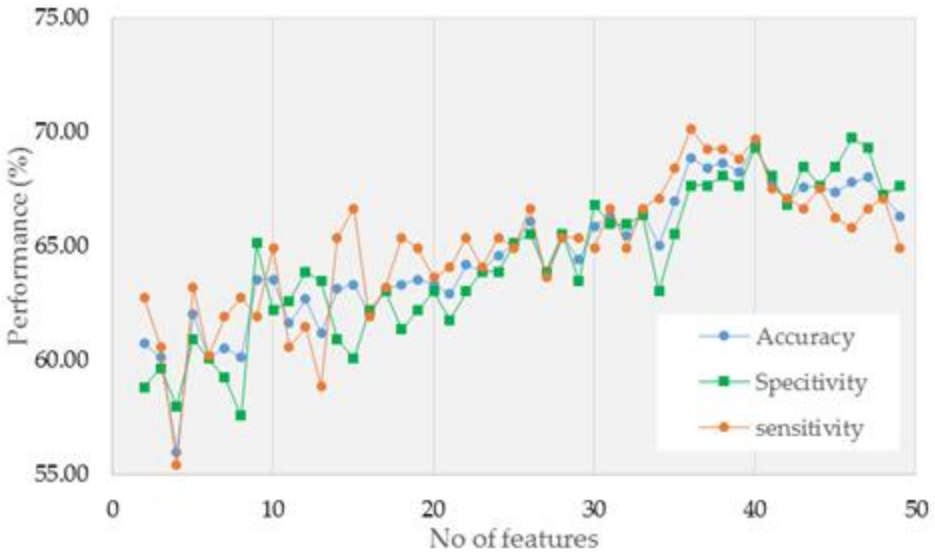

Figure 2-4: Performance of relief ranked features

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**).

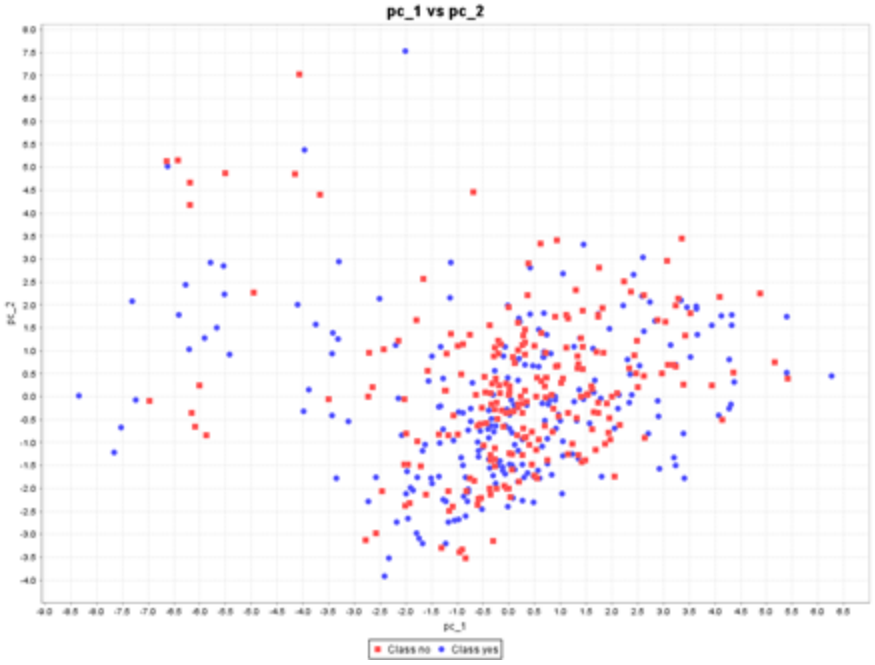

Figure 2-5: PC1 Vs. PC2 of 796 features prior to relief feature selection

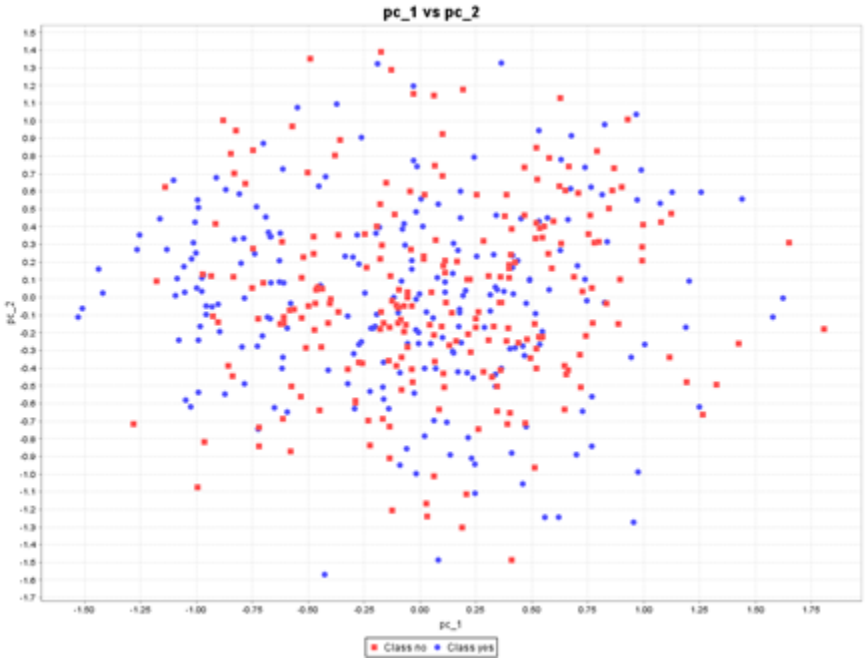

Figure 2-6: PC1 Vs. PC2 of 39 features post relief feature selection

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

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

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

	Measures of overlap		**Measures of Class inseparability**				**Measures of non-linearity**	
No of fold	**F1**	**F3**	**N1**	**N2**	**L1**	**L2**	**L3**	**N4**
1	0.0814	0.0332	0.5569	0.9151	0.7649	0.3246	0.2720	0.1945
2	0.0946	0.0000	0.5664	0.9123	0.7454	0.2749	0.2395	0.1265
3	0.1095	0.0355	0.5190	0.9370	0.7684	0.3175	0.2540	0.1810
4	0.1173	0.0332	0.4834	0.9037	0.7348	0.2867	0.2440	0.1990
5	0.0945	0.0332	0.5450	0.9322	0.7254	0.2725	0.2585	0.1540
6	0.0814	0.0355	0.5592	0.9257	0.7631	0.3152	0.2500	0.1605
7	0.1046	0.0355	0.5237	0.9020	0.7439	0.2891	0.2525	0.1360
8	0.0797	0.0190	0.4929	0.9144	0.7393	0.2891	0.2240	0.1385
9	0.0747	0.0355	0.5308	0.9205	0.7262	0.2678	0.2535	0.1385
10	0.0916	0.0355	0.4634	0.9065	0.7387	0.2766	0.2175	0.1060
Average	**0.0942**	**0.0292**	**0.5204**	**0.9171**	**0.7428**	**0.2877**	**0.2437**	**0.1489**

The comparison in measure of eight data complexity descriptors are shown in **Figure 2-7**.

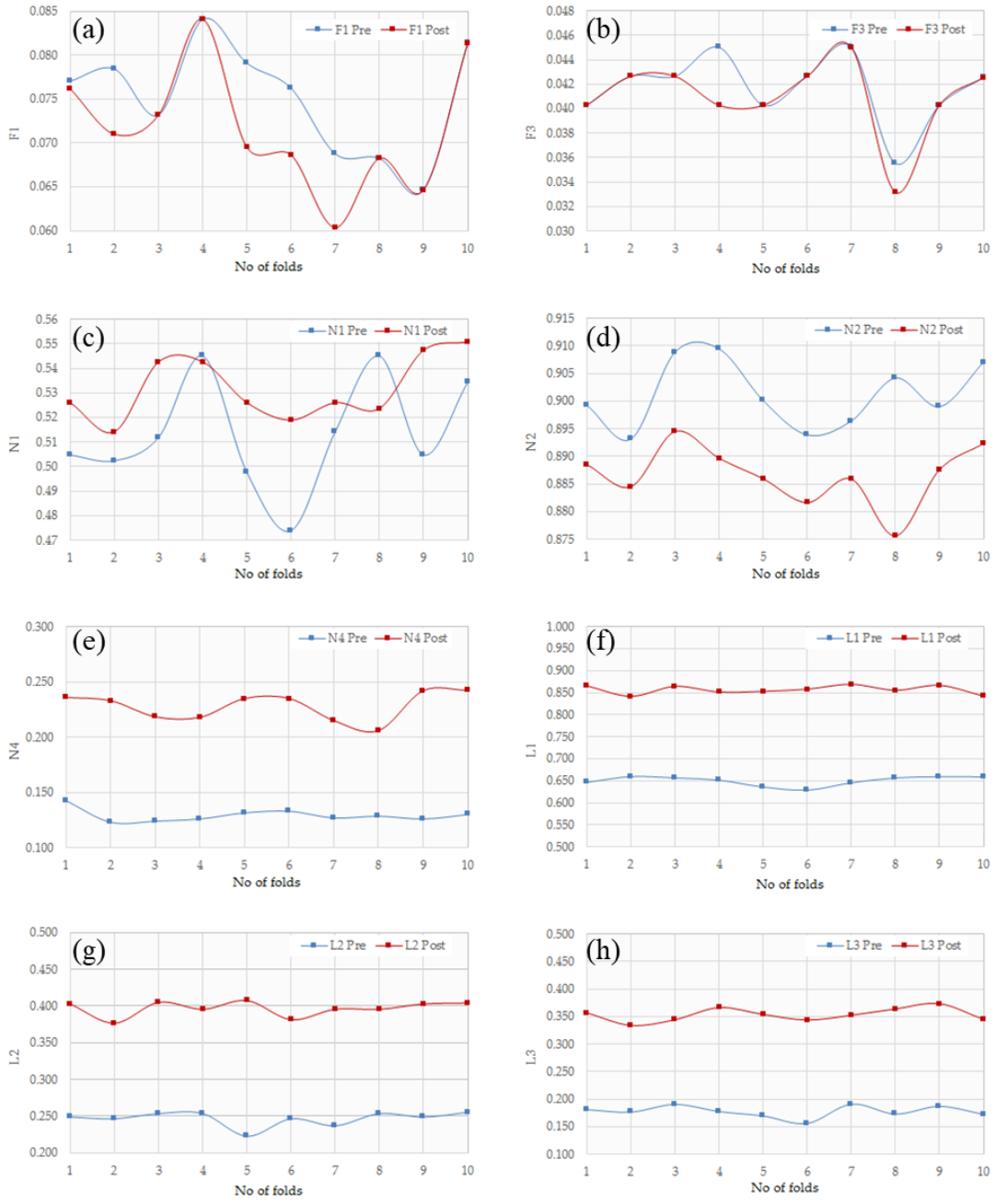

Figure 2-7: (a-h) Comparison of measures of eight data complexity descriptors pre and post feature selection process.

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.

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

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.

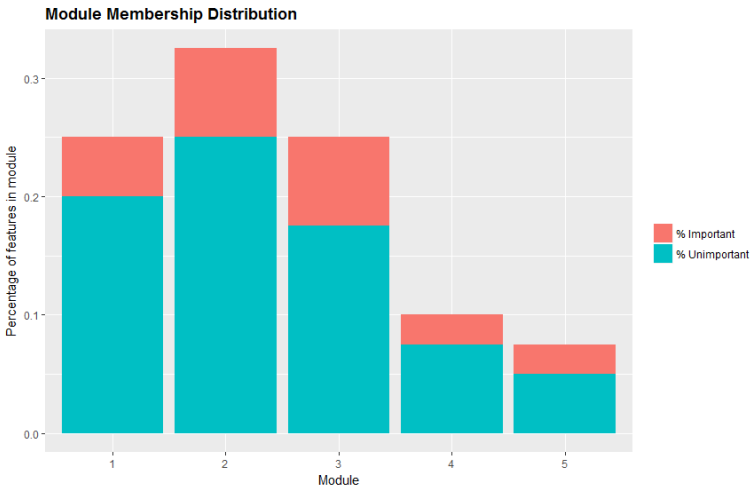

Figure 2-8: Results of module/partition membership distribution

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.

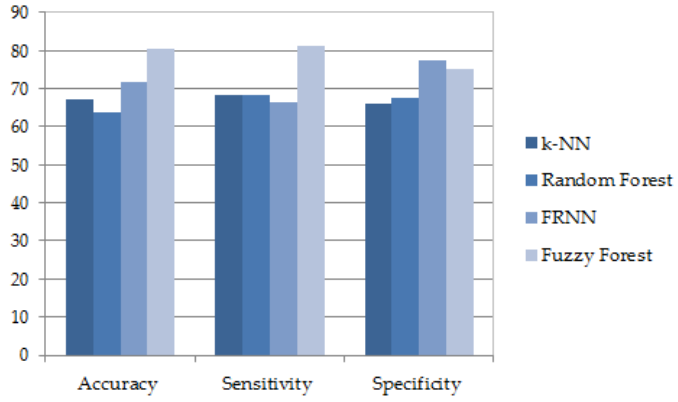

Figure 2-9: Comparison of performance of non-fuzzy and fuzzy classifiers

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 2.33×10^{-11}.

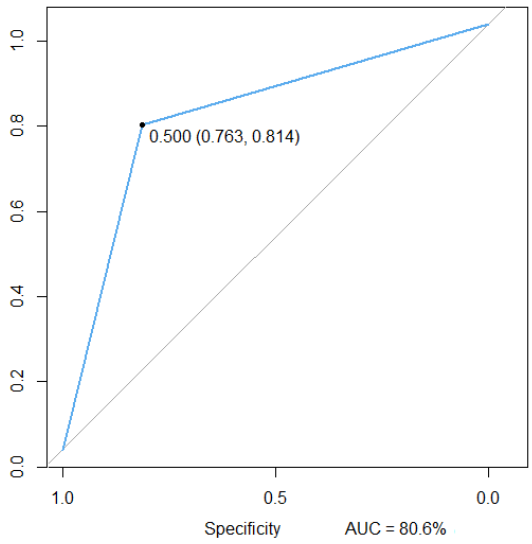

Figure 2-10: ROC curve (sensitivity vs specificity) for fuzzy forest

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

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

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 run time is very close to the linear runtime as shown in **Figure 2-11**

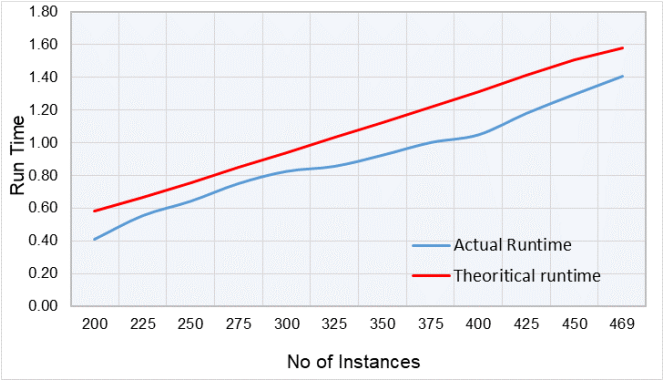

Figure 2-11: Comparison between actual versus theoretical runtime with respect to number of instances.

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

Authors	**No. of Subjects**	**Method**	**Classifier**	**Performance**
Tang [27]	Normal: 95 Malignant: 121	Four statistical moments	Kernel partial least square classifier	Acc: 99.35% Sen: 99.5% Spe: 99.16%
Petricoin [28]	Benign: 66 Malignant:50	Proteomic spectra	Genetic algorithm + self-organizing cluster analysis	Sen: 100% Spe: 95%
Biagiottiet [8]	Benign:175 Malignant:51	Age and parameters from TVUS images	Three-layer back propagation network	Sen: 96%
Tailor [9]	Benign:52 Malignant:15	Clinical and ultrasound based variables from TVUS images	Back propagation neural network	Sen: 100% Spe: 98.1%
Lucidarme [10]	Benign:234 Malignant:141	Backscattered ultrasound (3D TVUS)	Ovarian HistoScanning (OHS) system	Sen: 98% Spe: 88% Acc: 91.73%
Zimmer [11]	–	B-scan ultrasound images	Morphological Analysis	Acc: 70%
Acharya [12]	Benign:10 Malignant:10	Local Binary Pattern+ Law's Texture Energy	SVM	Sen: 100% Spe: 99.8% Acc: 99.9%
Acharya [13]	Benign:10 Malignant:10	Hu's invariant moments + Gabor wavelet features + Entropies	PNN + Tuned with Genetic algorithm	Sen: 99.2% Spe: 99.6% Acc: 99.8%
Acharya [14]	Benign:10 Malignant:10	Texture + higher order spectral features	Decision Tree	Sen: 94.3% Spe: 99.7% Acc: 97.0%
Acharya [6]	Benign:10 Malignant:10	First order statistics+ GLCM + run length matrix	k-NN/PNN	Sen: 100% Spe: 100% Acc: 100%
Our Method	**Non-suspicious:238 Suspicious:231**	**Radon Transform + Non-linear Feature extraction**	**Fuzzy Forest**	**Sen: 81.40% Spe: 76.30% Acc: 80.60%**

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 FRAEWORK

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 S**ection 3.4**, we described algorithm and detail methodologies for our proposed technique. In S**ection 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 S**ection 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: $\mu_1 = \mu_2 = \mu_3 = \cdots \ldots \ldots = \mu_n = 1 \ (n = 25, 50, 75, 100)$

Covariance matrix: $\begin{bmatrix} 1/2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/2 \end{bmatrix}$

In a covariance matrix, a diagonal element $m_{i,i}$ refers to the variances of the i^{th} variable and an off-diagonal element, $m_{i,j}$ $(i \neq j)$ refers to the covariance between i^{th} and j^{th} variables. Since we used $n \times n$ diagonal matrix as covariance matrix, for any $i \neq j, m_{i,j} = 0 \ and \ i = j, m_{i,i} = 1/2,$ 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 μ_N and variance σ_N at time *t* as follows:

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

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

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:	$V_{cur}, V_{min}, V_{max}, V_{inc}, V_{dec}$
Magnify Variances:	$V_{cur} = V_{cur} + V_{inc}\ until\ V_{cur} < V_{max}$
	$V_{inc} = V_{inc} + 0.5\ when\ V_{cur} = V_{max}$
Shrink Variances:	$V_{cur} = V_{cur} - V_{dec}\ until\ V_{cur} > V_{min}$
	$V_{dec} = V_{dec} + 1\ when\ V_{cur} = V_{min}$

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: $m_{cur}, m_{min}, m_{max}, m_{inc}, m_{dec}$

Magnify mean: $m_{cur} = m_{cur} + m_{inc}$ $until\ m_{cur} < \quad m_{max}$

$m_{inc} = m_{inc} + 2.5$ $when\ m_{cur} = m_{max}$

Shrink mean: $m_{cur} = m_{cur} - m_{dec}\ until\ m_{cur} > \quad m_{min}$

$m_{dec} = m_{dec} + 1\ when\ m_{cur} = m_{min}$

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 [55] 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.

Dataset	Source	# of Records	# of Dimensions Chosen
Spruce	Forest Cover Type	211840	10
Lodgepole Pine	Forest Cover Type	283301	10
Ascending Stairs	PAMAP2	117216	30
Cycling	PAMAP2	164600	30
Descending stairs	PAMAP2	104944	30
Ironing	PAMAP2	238690	30
Vacuum cleaning	PAMAP2	175353	30

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

ALGORITHM 1: Framework MCD-PuLSIF

Procedure MCD-PuLSIF
Parameters: Sliding window size l, sliding step size s, method name mn, cumulative variance cv
Input: streaming data $S_1 \leftarrow \{x_1, x_2, \dots, x_t, \dots\}$
Output: time t_{cp} when detecting a change
%set current time equal to window size
Initialize current time $t_{cur} \leftarrow l$
while a new sample x_t arrives in the stream **do**
Set sliding window $W_{cur} \leftarrow \{x_t; t \in [t_{cur} - l + 1: t_{cur}]\}$
Divide W_{cur} into two data segments, reference data segment DS_{ref} and test data segment DS_{test}
Set $DS_{ref} \leftarrow \{x_t; t \in \left[t_{cur} - l + 1: \frac{t_{cur}}{2}\right]\}$
Set $DS_{test} \leftarrow \{x_t; t \in \left[\frac{t_{cur}}{2} + 1: t_{cur}\right]\}$
%apply PCA on both data segments
$PC_{ref} \leftarrow CalculatePCA(DS_{ref})$
$PC_{test} \leftarrow CalculatePCA(DS_{test})$
Set n = number of first n principle component which capture cumulative variance equal to cv
$PC'_{ref} \leftarrow$ first n componemts from PC_{ref}
$PC'_{test} \leftarrow$ first n componemts from PC_{test}
% define forward divergence f_r and backward divergence f_b

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

ALGORITHM 1: Framework MCD-PuLSIF

13: $f_r \leftarrow D(PC'_{ref} \,||\, PC'_{test}) \leftarrow Calculatef(PC'_{ref}, PC'_{test}, mn)$
14: $f_b \leftarrow D(PC'_{test} \,||\, PC'_{ref}) \leftarrow Calculatef(PC'_{test}, PC'_{ref}, mn)$
15: $f_{sym} \leftarrow f_r + f_b$
% f_τ is the cutoff range to detect change point
16: $f_\tau \leftarrow CalculateCutOffF(f_{sym}, listf_{cp})$
17: **if** $f_{sym} \in f_\tau$
Report a change point at time $\frac{t_{cur}}{2}$ 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 Calculatef**(X_{ref}, X_{test}, mn)
% $mn \leftarrow$ KLIEP or uLSIF
22: find density ration estimator $\widehat{g(X)}$ using density ration estimation method
23: Estimate divergence $f(X_{ref}||X_{test})$ using $\widehat{g(X)}$
24: return f
25: **end procedure**
26: **Procedure CalculateCutOffF**($f_{cur}, listf_{cp}$)
% $listf_{cp}$ contains f values which detected change point at time $t_{cp}, t_{cp+1}, t_{cp+2}, \ldots\ldots, t_{cp+n}, \ldots$
% $t_{cp+n} < t_{cur}$
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: Centralize X
34: Apply SVD on X
35: u ← 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 $t_{landmark}$ and $t_{current}$ ($t_{landmark} > t_{current}$) to be the landmark time point and the current time point respectively. For Landmark Window, records only from the time interval defined by $[t_{landmark} : t_{current}]$ 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 $[t_{current} - l + 1 : t_{current}]$ 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 1.5×10^2 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 $\boldsymbol{x_i}$. 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 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 $\lambda_1, \lambda_2, \ldots, \lambda_n$ and corresponding eigenvectors $v_1, v_2, \ldots, v_n$. The eigenvector $(v_{max})_1$ associated with the highest eigenvalue is the first principle component. Similarly, the eigenvector $(v_{max})_2$ 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 λ_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 $O(n^2p + p^2n)$, 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:

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

$$DD(P_{ref}||P_{test}) = \int P_{ref}(x) f(\frac{P_{ref}(x)}{P_{test}(x)})\, dx. \qquad \textbf{Eq. 3-2}$$

Here, P_{ref} and P_{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:

$$f = xlog(x). \quad \textbf{Eq. 3-3}$$

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

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

$$f = \frac{1}{2}(x-1)^2. \quad \textbf{Eq. 3-5}$$

$$PE(P_{ref}||P_{test}) = \int P_{ref}(x)(\frac{P_{ref}(x)}{P_{test}(x)} - 1)^2 \, dx. \quad \textbf{Eq. 3-6}$$

Since $\frac{P_{ref}(x)}{P_{test}(x)} \neq \frac{P_{test}}{P_{ref}}$, $D(P_{ref}||P_{test}) \neq \mathrm{D}(P_{test}\big|\big|P_{ref})$, 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:

$$KL_{(sym)} = \mathrm{KL}\,(P_{ref}||P_{test}) + \mathrm{KL}\,(P_{test}\big|\big|P_{ref}). \quad \textbf{Eq. 3-7}$$

$$PE_{(sym)} = \mathrm{PE}\,(P_{ref}||P_{test}) + \mathrm{PE}\,(P_{test}\big|\big|P_{ref}). \quad \textbf{Eq. 3-8}$$

In reality, actual probability distributions, both $P_{ref}(x)$ and $P_{test}(x)$ 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 $P_{ref}(x)$ and $P_{test}(x)$ separately, the density-ratio estimation method focuses in learning the density-ratio function $\frac{P_{ref}(x)}{P_{test}(x)}$ 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:

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

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

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

$$\widehat{P_{ref}}(x) = \widehat{r(x)} * P_{test}(x). \qquad \textbf{Eq. 3-11}$$

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

$$\int \widehat{P_{ref}}(x) dx = \int \widehat{r(x)} * P_{test}(x)\, dx = 1. \qquad \textbf{Eq. 3-12}$$

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

$$KL_d = KL(P_{ref} \parallel \widehat{P_{ref}}) = \int P_{ref}(x) \log\left(\frac{P_{ref}(x)}{\widehat{P_{ref(x)}}}\right) dx. \qquad \textbf{Eq. 3-13}$$

Now the approximation of KL-divergence between P_{ref} and P_{test} will be closer to the actual KL-divergence between P_{ref} and P_{test} when KL_d, the divergence between the

actual probability distribution P_{ref} and estimated probability distribution $\widehat{P_{ref}}(x)$ will be minimum. Combining **Eq. 3-10** and **Eq. 3-13**, we have the following mathematical formulation:

$$KL(P_{ref} \parallel \widehat{P_{ref}}) = \int P_{ref}(x)\log\left(\frac{P_{ref}(x)}{\widehat{r(x)} * P_{test}(x)}\right)dx. \quad \textbf{Eq. 3-14}$$

$$KL\left(P_{ref} \parallel \widehat{P_{ref}}\right) = \int P_{ref}(x)\, log\left(\frac{P_{ref}(x)}{P_{test}(x)}\right)dx - \int P_{ref}(x) log(\widehat{r(x)})dx. \quad \textbf{Eq. 3-15}$$

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

$$KL\left(P_{ref} \parallel \widehat{P_{ref}}\right) = Const - \int P_{ref}(x)\log\left(\widehat{r(x)}\right)dx. \quad \textbf{Eq. 3-16}$$

$$KL_d\left(\widehat{r(x)}\right) = Const - \int P_{ref}(x)\log\left(\widehat{r(x)}\right)dx. \quad \textbf{Eq. 3-17}$$

We define the 2nd part of equation (8) as follows:

$$KL_{2nd}\left(\widehat{r(x)}\right) = \int P_{ref}(x) log(\widehat{r(x)})dx. \quad \textbf{Eq. 3-18}$$

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

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

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

$$\frac{1}{n_{\widehat{test}}}\sum_{i=1}^{n_{\widehat{test}}} \hat{r}\left(x_i^{n_{\widehat{test}}}\right) = 1. \quad \textbf{Eq. 3-20}$$

The optimization problem was formulated using **Eq. 3-15** - **Eq. 3-20** as follows:

$$maximize\ \frac{1}{n_{\widehat{ref}}}\sum_{i=1}^{n_{\widehat{ref}}}\log\hat{r}(x_i^{n_{\widehat{ref}}})\ such\ that\ \frac{1}{n_{\widehat{test}}}\sum_{i=1}^{n_{\widehat{test}}}\hat{r}\ (x_i^{n_{\widehat{test}}}) := 1. \qquad \textbf{Eq. 3-21}$$

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

$$\widehat{KL}(x) = \frac{1}{n}\sum_{i=1}^{n} log\ (\hat{G}(x_i)). \qquad \textbf{Eq. 3-22}$$

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 $\widehat{r(x)}$ from actual density ratio $r(x)$ (see **Eq. 3-13**) such that squared error $\widehat{SQ}$ in the following equations is minimized:

$$SQR' = \frac{1}{2}\int(\widehat{r(x)} - r(x))^2 P_{test}(x)\,dx. \qquad \textbf{Eq. 3-23}$$

$$SQR' = \frac{1}{2}\int(\widehat{r(x)})^2 P_{test}(x)\,dx - \int\widehat{r(x)}\,r(x)P_{test}(x)\,dx + \frac{1}{2}\int(r(x))^2 P_{test}(x)\,dx. \qquad \textbf{Eq. 3-24}$$

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

$$SQR' = \frac{1}{2}\int(\widehat{r(x)})^2 P_{test}(x)\,dx - \int\widehat{r(x)}\,P_{ref}(x)\,dx + \frac{1}{2}\int r(x)P_{ref}(x)\,dx. \qquad \textbf{Eq. 3-25}$$

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

$$SQR(\widehat{r(x)}) = \frac{1}{2}\int(\widehat{r(x)})^2 P_{test}(x)\,dx - \int\widehat{r(x)}\,P_{ref}(x)\,dx. \qquad \textbf{Eq. 3-26}$$

Applying empirical averages, the approximation $SQR\left(\widehat{r(x)}\right) from\ SQR\left(\widehat{r(x)}\right)$ is achieved as follows:

$$\widehat{SQR}\left(\widehat{r(x)}\right) = \frac{1}{2 * n_{\widehat{test}}} \sum_{i=1}^{n_{\widehat{test}}} \left(\hat{r}\left(x_i^{n_{\widehat{test}}}\right)\right)^2 - \frac{1}{n_{\widehat{ref}}} \sum_{i=1}^{n_{\widehat{ref}}} \hat{r}\left(x_i^{n_{\widehat{ref}}}\right). \qquad \textbf{Eq. 3-27}$$

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

$$\widehat{PE}(x) = -\frac{1}{2 * n} \sum_{i=1}^{n} \left(\hat{G}(x_i)\right)^2 + \frac{1}{n} \sum_{i=1}^{n} \hat{G}(x_i) + \frac{1}{2}. \qquad \textbf{Eq. 3-28}$$

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}, $t_{cp+3}, \ldots\ldots, t_{cp+n}$ with divergence $f_{cp}, f_{cp+1}, f_{cp+2}, \ldots\ldots, f_{cp+n}$ 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 fulltime 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:

$$f_{mean} = mean(f_{cp}, f_{cp+1}, f_{cp+2}, \dots\dots, f_{cp+n}). \quad \textbf{Eq. 3-29}$$

$$f_{std} = std(f_{cp}, f_{cp+1}, f_{cp+2}, \dots\dots, f_{cp+n}). \quad \textbf{Eq. 3-30}$$

.

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

$$f_{cutoff} = f_{mean} - f_{std}. \quad \textbf{Eq. 3-31}$$

$$cp = \begin{cases} 1, & f_{cur} \geq f_{cutoff} \\ 0, & otherwise \end{cases}. \quad \textbf{Eq. 3-32}$$

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_{detect}, and the closest actual change point exists at time t_{actual}. The detected change point is considered to be correct if $(t_{actual} - t_{adjust}) \leq t_{detect} \leq (t_{actual} + t_{adjust})$ where t_{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 $t_k - t_{k-1} \leq 2 * t_{adjust}$.

For artificial datasets with a sample size 5000, we set t_{adjust} equal to be 12.5, whereas for real datasets with a record size of 5×10^5, we choose t_{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:

$$TP = \frac{n_{cr}}{n_{cp}}. \quad \textbf{Eq. 3-33}$$

$$FP = \frac{n_{al} - n_{cr}}{n_{al}}. \quad \textbf{Eq. 3-34}$$

$$FN = \frac{n_{ncr}}{n_{cp}}. \quad \textbf{Eq. 3-35}$$

$$P = \frac{TP}{TP + FP}. \quad \textbf{Eq. 3-36}$$

$$R = \frac{TP}{TP + FN}. \quad \textbf{Eq. 3-37}$$

$$F1 - Score = \frac{2 * P * R}{P + R}. \quad \textbf{Eq. 3-38}$$

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' packag e 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.

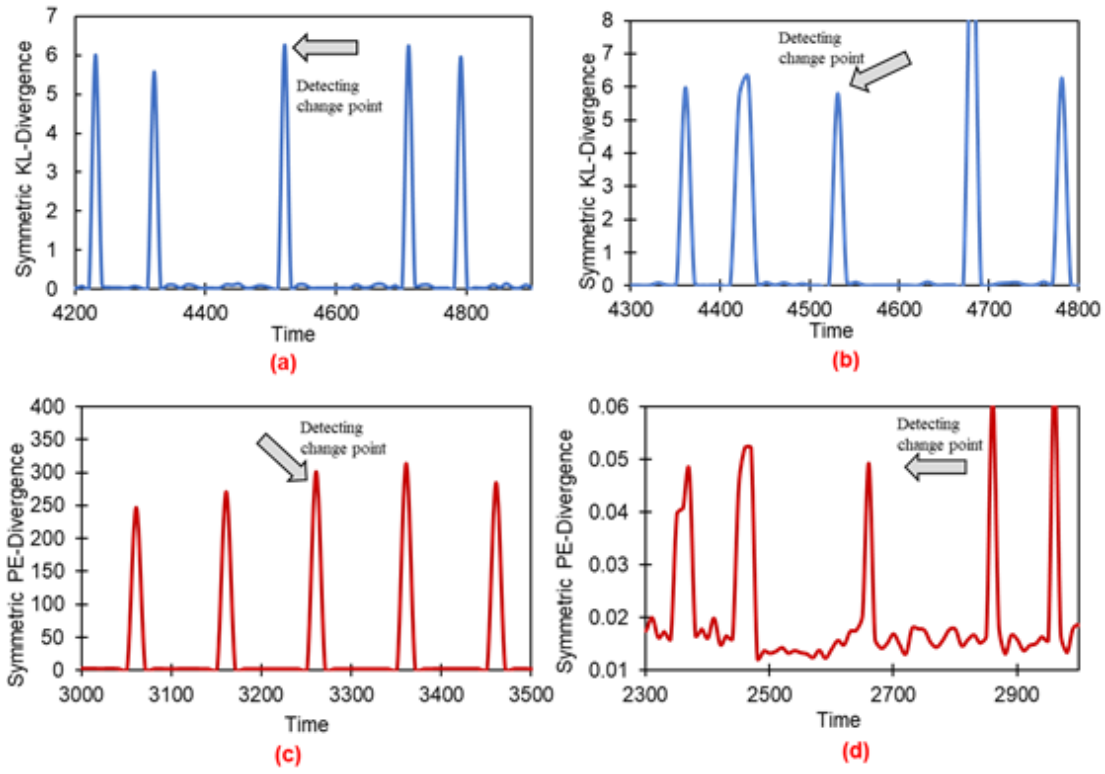

Figure 3-1: Potential change points detection for type 01 datasets where KL-divergence and PE-divergence undergoes noticeable changes. (a) & (b) time *vs.* KL-divergence between PCA transformed data segments (original feature dimension equals to 25, and 75, respectively). (c) & (d) time *vs.* PE-divergence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

	Only KLIEP				**PCA + KLIEP**			
# of features	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
25	73.469	19.027	26.531	11.13	85.714	16.372	14.286	9.47
50	77.551	18.502	22.449	14.41	83.673	16.814	16.327	10.79
75	75.510	18.943	24.490	17.33	81.633	17.257	18.367	11.28
100	69.388	20.264	30.612	25.84	75.510	18.584	24.490	13.67
Average	**73.980**	**19.184**	**26.020**	**17.18**	**81.633**	**17.257**	**18.367**	**11.30**

Table 3-4: Comparison of performance of uLSIF on full features and PCA reduced features for four type #01 synthetic datasets

	Only uLSIF				PCA + uLSIF			
# of features	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
25	85.714	16.740	14.286	6.83	91.837	15.419	8.163	4.69
50	81.633	17.621	18.367	7.23	89.796	15.859	10.204	4.80
75	79.592	18.062	20.408	8.96	83.673	17.181	16.327	5.59
100	71.429	19.824	28.571	10.16	79.592	17.333	20.408	6.32
Average	**79.592**	**18.062**	**20.408**	**8.30**	**86.224**	**16.448**	**13.776**	**5.35**

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.

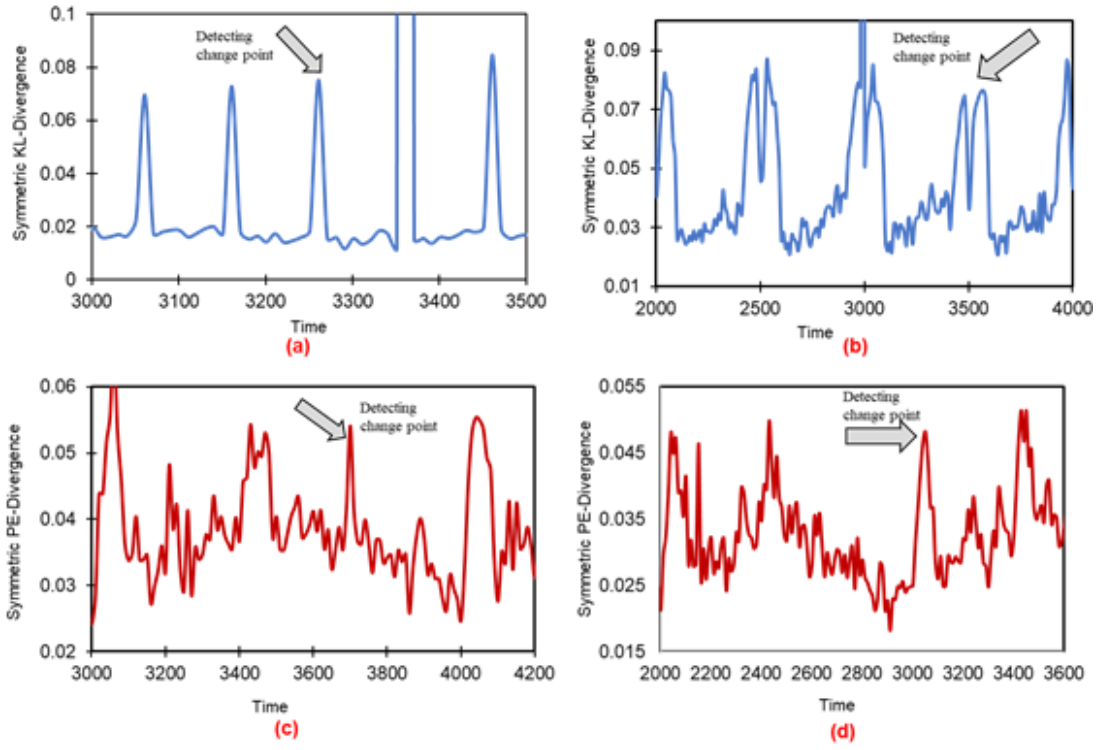

Figure 3-2: Potential change points detection for type 02 datasets where KL-divergence or PE-divergence undergoes noticeable changes. (a) & (b) Time vs. KL-divergence between PCA transformed data segments with original feature dimension equals to 50 and 100 respectively. (c) & (d) Time vs. PE-divergence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

	Only KLIEP				**PCA + KLIEP**			
# of features	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
25	85.714	15.247	14.286	14.71	91.837	15.419	8.163	9.04
50	83.673	15.315	16.327	13.74	89.796	15.111	10.204	9.99
75	81.633	16.889	18.367	18.90	83.673	16.444	16.327	11.83
100	77.551	17.040	22.449	39.66	79.592	16.964	20.408	13.36
Average	**82.143**	**16.123**	**17.857**	**21.75**	**86.224**	**15.985**	**13.776**	**11.06**

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

	Only uLSIF				**PCA + uLSIF**			
# of features	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
25	87.755	16.300	12.245	7.31	97.959	13.717	2.041	4.01
50	83.673	16.444	16.327	7.62	91.837	15.044	8.163	4.33
75	77.551	18.142	22.449	9.71	87.755	15.929	12.245	4.92
100	73.469	19.027	26.531	11.65	83.673	16.444	16.327	5.55
Average	**80.612**	**17.478**	**19.388**	**9.07**	**90.306**	**15.284**	**9.694**	**4.70**

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.

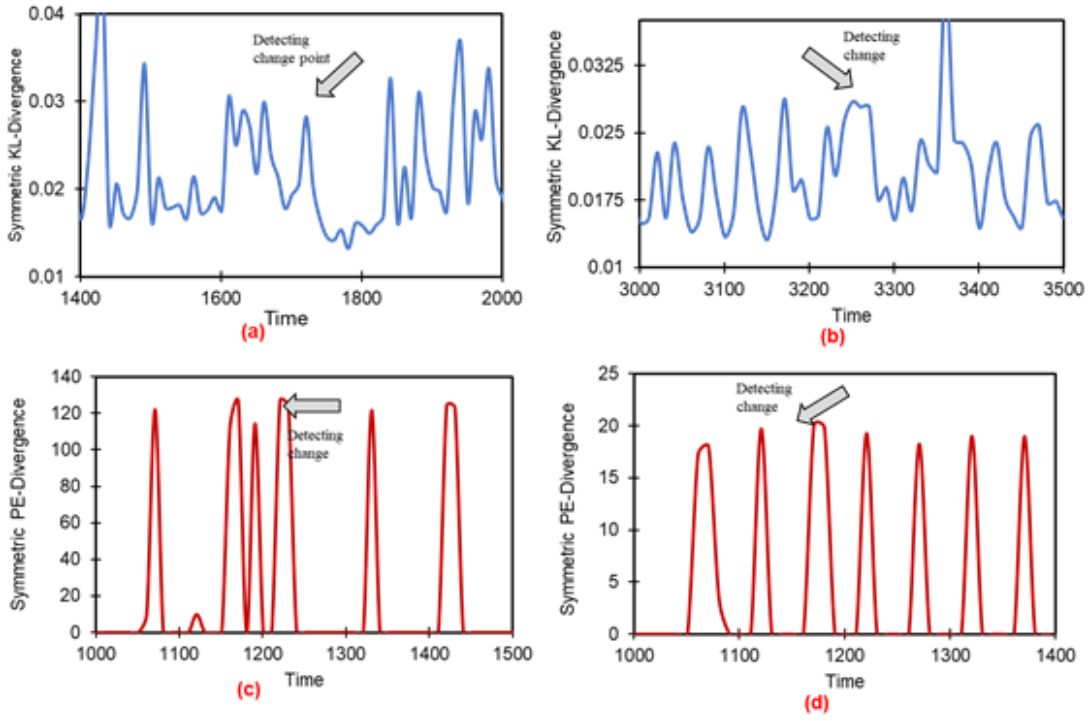

Figure 3-3: Potential change points detection for type 03 datasets where KL-divergence or PE-divergence undergoes noticeable changes. (a) & (b) Time vs. KL-divergence between PCA transformed data segments with original feature dimension equals to 25 and 100 respectively. (c) & (d) Time vs. PE-divergence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

	Only KLIEP				**PCA + KLIEP**			
# of features	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
25	73.469	19.383	26.531	11.651	81.633	17.257	18.367	9.73
50	83.673	16.071	16.327	27.411	83.673	16.071	16.327	10.94
75	75.510	18.943	24.490	31.817	79.592	22.034	20.408	11.51
100	69.388	19.912	30.612	33.311	77.551	18.502	22.449	13.16
Average	**75.510**	**18.577**	**24.490**	**26.05**	**80.612**	**18.466**	**19.388**	**11.33**

Table 3-8: Comparison of performance of uLSIF on full features and PCA reduced features on four type #03 datasets.

	Only uLSIF				**PCA + uLSIF**			
# of features	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
25	83.673	16.814	16.327	7.31	89.796	15.111	10.204	4.05
50	79.592	16.964	20.408	7.62	85.714	16.740	14.286	4.69
75	77.551	18.142	22.449	9.71	81.633	17.257	18.367	5.05
100	75.510	17.117	24.490	11.65	79.592	18.062	20.408	5.70
Average	**79.082**	**17.259**	**20.918**	**9.07**	**84.184**	**16.792**	**15.816**	**4.87**

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

$$Improvement_X = \frac{X_{average}^{Full+method} - X_{average}^{PCA+method}}{X_{average}^{Full+method}} * 100. \qquad \textbf{Eq. 3-39}$$

Here, $X = TP, FP, FN, execution\ time$ and $method = KLIEP, uLSIF$ 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.

	(Original Features + KLIEP) Vs. (PCA + KLIEP)				**(Original Features + uLSIF) Vs. (PCA + uLSIF)**			
Dataset Type	Change in TP (%)	Change in FP (%)	Change in FN (%)	Change in Avg runtime per window (%)	Change in TP (%)	Change in FP (%)	Change in FN (%)	Change in Avg runtime per window (%)
Type 01	(+)10.34	(-)10.05	(-) 29.41	(-) 34.21	(+) 8.33	(-) 8.93	(-) 32.50	(-) 35.50
Type 02	(+) 4.97	(-) 0.86	(-) 22.86	(-) 49.18	(+) 12.03	(-) 12.55	(-) 24.39	(-) 48.18
Type 03	(+) 6.76	(-) 0.60	(-) 20.83	(-) 56.49	(+) 4.50	(-) 6.46	(-) 22.22	(-) 46.32
Average	(+) **7.36**	(-) **3.83**	(-) **24.37**	(-) **46.62**	(+) **8.29**	(-) **9.32**	(-) **26.37**	(-) **43.33**

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**.

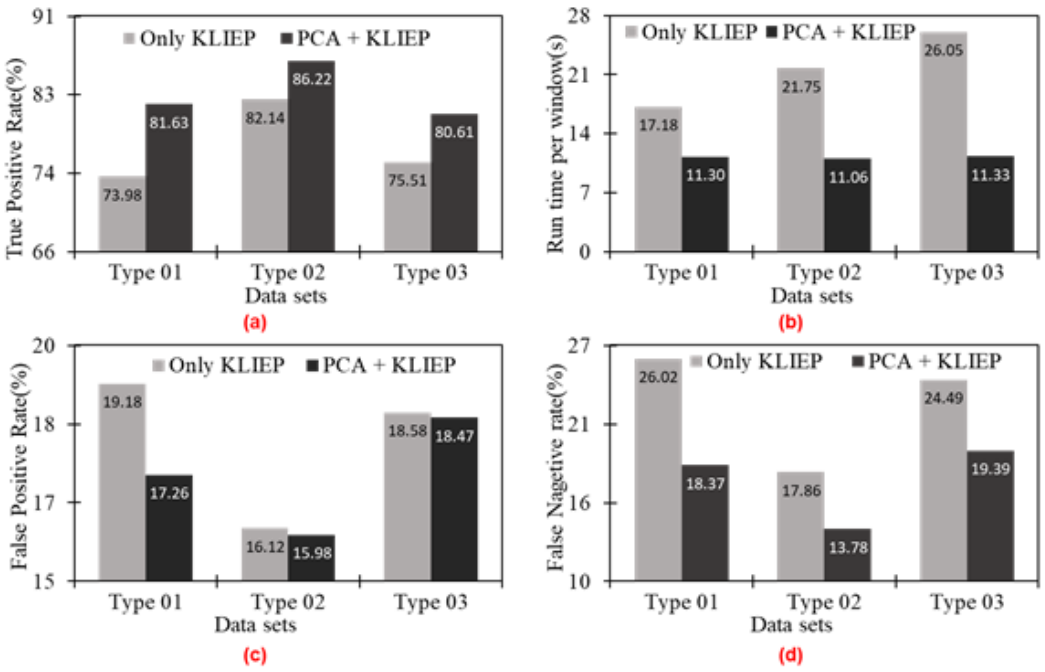

Figure 3-4: The results of performance improvement of KLIEP method in terms of TP, FP, FN, and execution time for PCA

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**.

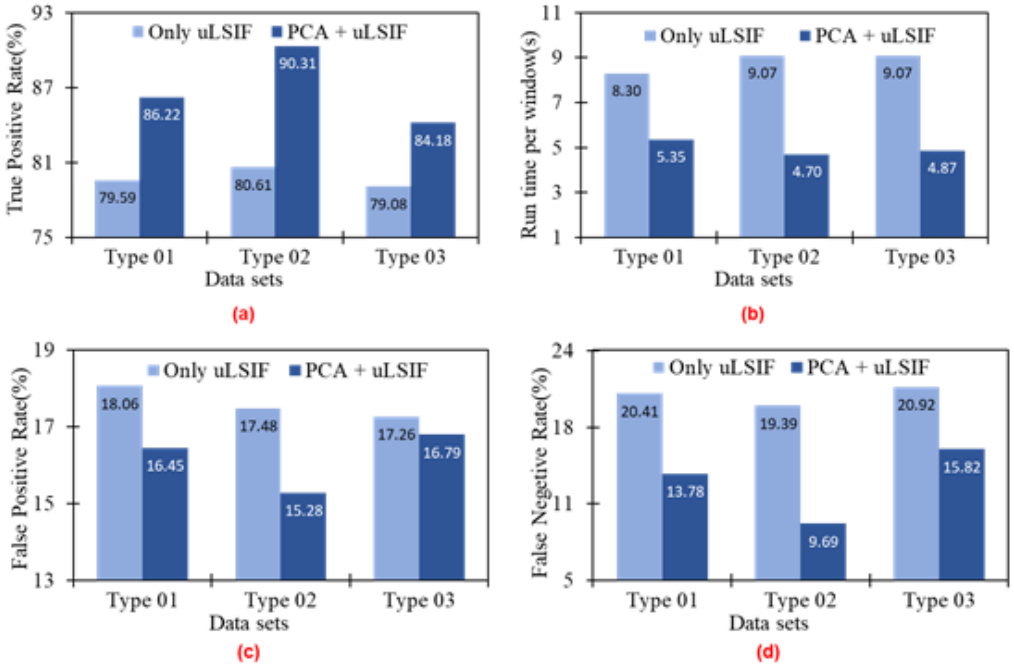

Figure 3-5: The results of performance improvement of uLSIF method in terms of TP, FP, FN, and execution time for PCA.

The results demonstrated that KLIEP and uLSIF, both density ration 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.

	PCA + KLIEP				**PCA + uLSIF**			
dataset Type	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)	TP (%)	FP (%)	FN (%)	Avg runtime per window (in sec)
Type 01	81.633	17.257	18.367	11.30	86.224	16.448	13.776	5.35
Type 02	86.224	15.985	13.776	11.06	90.306	15.284	9.694	4.70
Type 03	80.612	18.466	19.388	11.33	84.184	16.792	15.816	4.87
Average	**82.823**	**17.236**	**17.177**	**11.230**	**86.905**	**16.175**	**13.095**	**4.97**

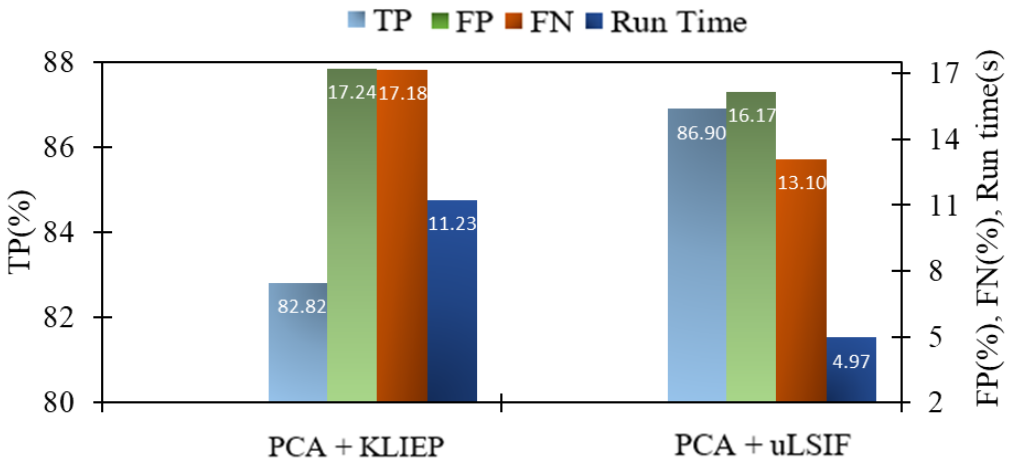

Figure 3-6: Comparison of performance improvement between KLIEP and uLSIF method in terms of TP, FP, FN, and execution time after dimensionality reduction by PCA.

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 5×10^5. 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.

	Results from published work [27]					**Results from proposed framework**
Datasets	**kdt-tree**	**PCA-SPLL**	**CD-LLH**	**CD-MKL**	**CD-Area**	**MCD-PuLSIF**
Spruce(G1D)	0.676	0.493	0.913	0.846	**0.925**	0.917
Spruce(S1D)	0.676	0.346	0.778	0.772	0.884	0.895
Lodgepole Pine(G1D)	0.492	0.446	0.872	0.856	0.904	0.920
Lodgepole Pine(S1D)	0.346	0.735	0.794	0.741	0.839	0.914
Ascending Stairs(G1D)	0.585	0.504	0.838	0.785	0.848	0.891
Ascending Stairs(S1D)	0.592	0.772	**0.995**	0.761	0.821	0.843
Cycling(G1D)	0.614	0.352	0.814	0.832	0.849	0.870
Cycling(S1D)	0.658	0.387	**1.000**	0.805	0.833	0.895
Descending Stairs(G1D)	0.563	0.535	0.809	0.828	0.811	0.901
Descending Stairs(S1D)	0.746	0.406	**0.989**	**0.878**	**0.872**	0.866
Ironing(G1D)	0.599	0.559	0.862	0.753	0.772	0.894
Ironing(S1D)	0.649	0.615	**0.995**	0.794	0.828	0.869
Vacuum cleaning(G1D)	0.473	0.628	0.809	0.831	0.822	0.892
Vacuum cleaning(S1D)	0.691	0.691	**0.985**	0.782	0.814	0.920
Average (F1-Score)	**0.597**	**0.534**	**0.890**	**0.805**	**0.844**	**0.892**

At first, we searched five nearest neighbors $x_1, x_2, \dots, x_5$ of x (FNN R package [62]) followed by drawing $\left\lceil \frac{5\times10^5 - n_{D_i}}{n_{D_i}} \right\rceil$ 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 5×10^5, we infused two types of changes every 2×10^4 data samples. The first type of change was incorporated by adding one dimensional Gaussian random variable (G1D) to a randomly selected feature every 2×10^4 data samples and the second type of change was fused by

scaling a randomly selected feature by two (S1D) every 2×10^4 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 method 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 0.0.890), 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.

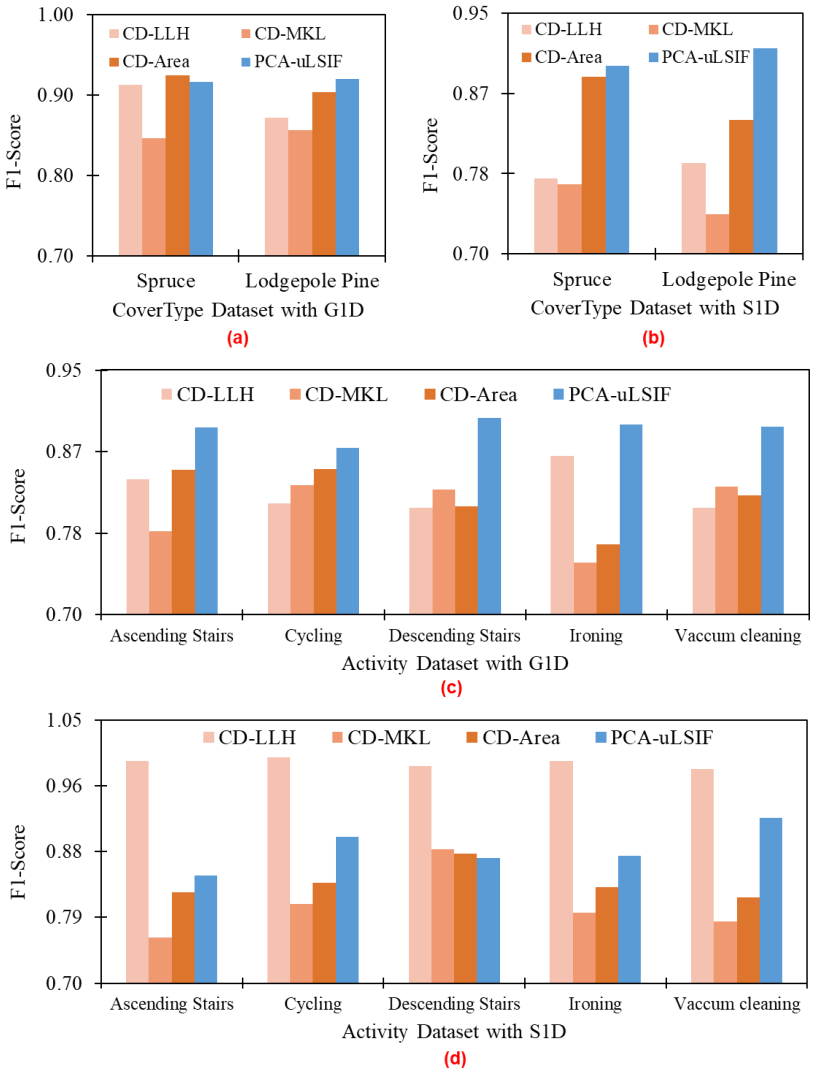

Figure 3-7: Comparison of experimental results of MCD-PuLSIF method with CD-LLH, CD-MKL, and CD-Area using cover type and activity datasets separately.

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.

Window Size	No of PCAs	Run time(sec)	Window Size	No of PCAs	Run time(sec)
10000	10	46.12	15000	30	110.49
10000	20	44.99	20000	10	185.89
10000	30	45.42	20000	20	186.20
15000	10	111.17	20000	30	185.67
15000	20	111.00	----	---	---

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.

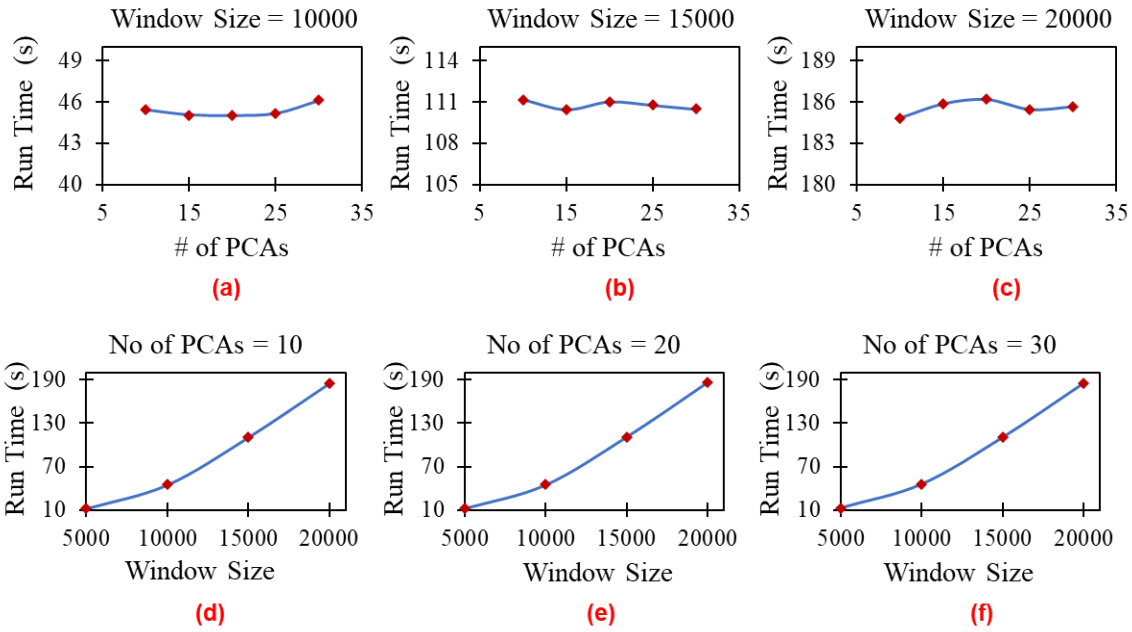

Figure 3-8: Comparison of runtime against window size and feature dimension for MCD-PuLSIF.

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 $O(n^2 p)$. 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 BASLITE 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

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:

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

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

$$X = QR. \quad \textbf{Eq. 4-2}$$

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:

$$u_1 = \frac{f_i}{\| f_i \|_F}. \quad \textbf{Eq. 4-3}$$

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:

$$u_2 = \frac{f_2 - (u_1^T f_1)u_1}{\| f_2 - (u_1^T f_1)u_1 \|_F}. \quad \textbf{Eq. 4-4}$$

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

$$f_j^* = f_j - (u_1^T f_1 u_1 + \cdots + u_{i-1}^T f_j u_{i-1}). \quad \textbf{Eq. 4-5}$$

where $i = 2, \dots, n \; and \; j = i, \dots, n$ 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:

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

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

$$Q^T X P = R. \qquad \textbf{Eq. 4-7}$$

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

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

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

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

where U is a column-orthonormal $r \times r$ matrix, Σ is a diagonal r × n matrix with the singular values σ_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 × 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 $\{X_t \in \mathbb{R}^{n\times m}, t = 1,2, \dots\}$ 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 $\{X_{t_{(r)}} \epsilon \mathbb{R}^{n\times r}, t = 1,2, \dots\}$. 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:

$$\mathrm{X}^T_{t\ (r)} = U_{t_{(r)}} \sum_{t\ (r)} V^T_{t\ (r)}. \qquad \textbf{Eq. 4-10}$$

where $X_{t_{(r)}} X^T_{t\ (r)} = V_{t_{(r)}} \sum^2_{t\ (r)} V^T_{t\ (r)}$ forms the cosine affinity matrix of $X_{t_{(r)}}$ due to column-wise Frobenius-normalization. We adapted the concept mentioned in [92] by using $V^T_{t\ (r)}$ as the target variable in the regression. We formulate the resulting regression problem as

$$_{min\ A} \| \mathrm{X}_{t_{(r)}} A - V_{t_{(r)}} \|^2_F. \qquad \textbf{Eq. 4-11}$$

where each column in A refers to the combination coefficient for different features in approximating the right singular vectors $V_{t_{(r)}}$ of $U_{t_{(r)}} \sum_{t_{(r)}} V^T_{t\ (r)}$. 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:

$$P_{ex} = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}. \quad \textbf{Eq. 4-12}$$

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:

$$X_{t_{(k)}} = U_{t_{(k)}} \sum_{t_{(k)}} V_t^T{}_{(k)}. \quad \textbf{Eq. 4-13}$$

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:

$$_{min\ A} \| \mathrm{X}_t A - V_{t_{(k)}} \|_F^2. \quad \textbf{Eq. 4-14}$$

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:

$$\min_{A_t} \| X_t A_t - V_{t_{(k)}} \|_F^2 + \beta | \left| \mathrm{a}_{t_{i,j}} \right| |_p^p. \quad \textbf{Eq. 4-15}$$

where β is the regularization parameter. The purpose of using β is to control the magnitude and sparsity of the coefficients with larger values of β being both smaller and sparser. β 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 $w_t = \left\{w_{1_{(t)}}, w_{2_{(t)}}, \dots, w_{r_{(t)}}\right\} \in \mathbb{R}^r$ is defined by

$$\mathrm{i} \in \mathbb{Z}, 0 < \mathrm{i} \leq \mathrm{r}, w_{i_{(t)}} = \overset{max}{j} \left| a_{t_{\mathrm{i,j}}} \right|. \qquad \textbf{Eq. 4-16}$$

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:

$$W_{ex} = \frac{\sum_{i=1}^{h} w_{t_i}^2}{\sum_{i=1}^{m} w_{t_i}^2}. \qquad \textbf{Eq. 4-17}$$

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:

$$r(x) = \frac{P_{ref}(x)}{P_{cur}(x)}. \qquad \textbf{Eq. 4-18}$$

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 α-

relative density-ratio of P_{test} and $P_{\boldsymbol{train}}$ has been proposed in [95] to compare the probability density of two data distribution as follows:

$$r_\alpha(x) = \frac{P_{ref}(x)}{\alpha P_{ref}(x) + (1-\alpha)P_{cur}(x)}, 0 \le \alpha \le 1. \qquad \textbf{Eq. 4-19}$$

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

$$\mathrm{q}_\alpha(\mathrm{x}) = \alpha \mathrm{P}_{\mathrm{ref}}(\mathrm{x}) + (1-\alpha)\mathrm{P}_{\mathrm{cur}}(\mathrm{x}), 0 \le \alpha \le 1. \qquad \textbf{Eq. 4-20}$$

Hence, we can rewrite **Eq. 4-19** as follows:

$$r_\alpha(x) = \frac{P_{ref}(x)}{q_\alpha(x)}, 0 \le \alpha \le 1. \qquad \textbf{Eq. 4-21}$$

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

$$\begin{cases} r_\alpha(x) \ge \tau, & x \text{ is benign or normal} \\ r_\alpha(x) < \tau, & x \text{ is malicious} \end{cases}. \qquad \textbf{Eq. 4-22}$$

where τ 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 α-relative density-ratio, $r_\alpha(x)$, has been given by

$$g(x;\theta) = \sum_{l=1}^{n_{ref}} \theta_l \, K\left(x, x_{ref}\right). \qquad \textbf{Eq. 4-23}$$

The parameter $\theta = (\theta_1, \theta_2, \dots, \theta_n)^T$ in **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, $J(\theta)$:

$$J(\theta) = \frac{1}{2} E\left[\left(q_\alpha(x)\right)\right]\left[\left(g(x;\theta) - r_\alpha(x)\right)^2\right]. \qquad \textbf{Eq. 4-24}$$

$$J(\theta) = \frac{1}{2}\, E\big[\big(q_\alpha(x)\big)\big][(g(x;\theta))^2 - 2g(x;\theta)r_\alpha(x) + (r_\alpha(x))^2]. \quad \textbf{Eq. 4-25}$$

$$J(\theta) = \tfrac{1}{2}\, E\big[\big(q_\alpha(x)\big)\big][(g(x;\theta))^2 - 2g(x;\theta)r_\alpha(x)] + const. \quad \textbf{Eq. 4-26}$$

$$J(\theta) = \frac{1}{2}\, E\big[\big(\alpha P_{ref}(x) + (1-\alpha)P_{cur}\big)\big](g(x;\theta))^2 - E\big[\big(q_\alpha(x)\big)\big]g(x;\theta)r_\alpha(x) + const. \quad \textbf{Eq. 4-27}$$

$$J(\theta) = \frac{\alpha}{2}\, E[P_{ref}(x)]\big(g(x;\theta)\big)^2 + -E\big[\big(q_\alpha(x)\big)\big]g(x;\theta)\frac{P_{ref}(x)}{q_\alpha(x)} + const. \quad \textbf{Eq. 4-28}$$

$$J(\theta) = \frac{\alpha}{2}\, E[P_{ref}(x)]\big(g(x;\theta)\big)^2 - E\left[\left(q_\alpha(x)\frac{P_{ref}(x)}{q_\alpha(x)}\right)\right]g(x;\theta) + const. \quad \textbf{Eq. 4-29}$$

$$\mathrm{J}(\theta) = \frac{\alpha}{2}\, \mathrm{E}[\mathrm{P_{ref}(x)}]\big(\mathrm{g(x;\theta)}\big)^2 + -\mathrm{E}\big[\big(\mathrm{P_{ref}(x)}\big)\big]\mathrm{g(x;\theta)} + \mathrm{const.} \quad \textbf{Eq. 4-30}$$

The optimization problem by using the empirical averages to approximate the expectations in **Eq. 4-22** is given by

$$\hat{\theta} = argmin_{\theta \in R^n}\left[\frac{1}{2}\theta^T\hat{H}\theta - \hat{h}^T\theta + \frac{\lambda}{2}\theta^T\theta\right]. \quad \textbf{Eq. 4-31}$$

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

$$\hat{H}_{l_{ref},l'_{ref}} = \frac{\alpha}{n_{ref}}\sum_{i=1}^{n_{ref}} K\left(x_{i_{ref}}, x_{l_{ref}}\right)K\left(x_{i_{ref}}, x_{l'_{ref}}\right) + \frac{1-\alpha}{n_{cur}}\sum_{j=1}^{n_{cur}} K\left(x_{j_{cur}}, x_{l_{ref}}\right)K\left(x_{j_{cur}}, x_{l'_{ref}}\right). \quad \textbf{Eq. 4-32}$$

$$\hat{h}_{l_{ref}} = \frac{1}{n_{ref}}\sum_{i=1}^{n_{ref}} K\left(x_{i_{ref}}, x_{l_{ref}}\right). \quad \textbf{Eq. 4-33}$$

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

$$\hat{\theta} = (\hat{H} + \lambda I_n)^{-1}\hat{h}. \quad \textbf{Eq. 4-34}$$

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 σ and λ as proposed in [49]:

$$\hat{\theta} = (\hat{H} + \lambda I_n)^{-1}\hat{h}. \qquad \textbf{Eq. 4-35}$$

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 σ. The best combination of σ and λ is chosen by grid search through 5-fold cross-validation. Finally, a density-ratio estimator is given as

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

This mathematical formulation is known as Relative Unconstrained Least Square Importance Fitting (RuLSIF). True α-relative density-ratio $r_\alpha(x)$ described in **Eq. 4-36** is modeled by $\hat{r}_\alpha(x)$ in **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.

4.4.3 *k*-Fold Dynamic Threshold

We computed the value of the threshold, τ, 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 **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), τ_{min} are defined as follows:

$$\mathrm{SSC}\left(s_k(x_i)\right) = \left(\hat{r_\alpha}(x_i)\right)_{s_k}, i = 1,2,\ldots,\frac{2}{3}n. \quad \textbf{Eq. 4-37}$$

$$\mathrm{MDSC}(s_k) = \mathrm{median}\left(\left(\hat{r_\alpha}(x_i)\right)_{s_k}, i = 1,2,\ldots,\frac{2}{3}n. \right. \quad \textbf{Eq. 4-38}$$

$$\mathrm{MNSC}(s_k) = \mathrm{mean}\left(\left(\hat{r_\alpha}(x_i)\right)_{s_k}, i = 1,2,\ldots,\frac{2}{3}n. \right. \quad \textbf{Eq. 4-39}$$

$$\tau_{\mathbf{min}}(s_k) = \min\left(\mathrm{MDSC}(s_k), \mathrm{MNSC}(s_k)\right). \quad \textbf{Eq. 4-40}$$

Next, we created a sequence of 2k evenly spaced numbers $\tau_1, \tau_2, \ldots, \tau_{2k,}$ $0 < \tau \leq \tau_{\min}(s_k)$). We categorized each instance of subset s_k as a benign or malignant category using τ_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 τ-list as threshold τ_{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 τ for W_{ref} with size n as follows:

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

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.

ALGORITHM 1: framework IBDS
1: **Procedure** IBDS
Parameters: Value of α and fold k
Input: Reference or training window W_{ref}, current window W_{cur}, and α

Table 4-1: ALGORITHM 1: framework IBDS.

ALGORITHM 1: framework IBDS

Output: Safety score and response category (benign or malignant) of all traffic instances in the current window

2: $\tau \leftarrow \text{CalculateThreshold}(W_{ref}, k)$

3: $f_{ref} \leftarrow \text{CalculateImportantFeature}(W_{ref}, 1)$

4: $f_{cur} \leftarrow \text{CalculateImportantFeature}(W_{cur}, 1)$

5: $f \leftarrow$ superset of f_{ref} and f_{cur} $\%Safety\ Score\ SSC\ of\ x = r_a(x)$

6: $SSC(W_{cur}(x)) \leftarrow \text{RuLSIF}(W_{ref_{(f)}}, W_{cur_{(f)}})$

7: **If** $SSC(W_{cur}(x))$ is greater than or equal to τ **then**

8: traffic instance x is benign or normal

9: **Else**

10: traffic instance x is malignant and is launched from a botnet

11: **End If**

12**:** **end procedure**

13: **Procedure CalculateImportantFeature**(W, p)

Input: Reference or training window W and $p = 1$ for Lasso regularization or $p = 2$ for ridge regularization

Output: feature list f with higher feature importance coefficient

14: Apply Frobenius normalization on W

15: $W' \leftarrow$ Apply QRcp on W to find actual rank r and list of independent features of W

16: $U_{t_{(r)}} \sum_{t_{(r)}} V_{t}^{T}{}_{(r)} \leftarrow \text{SVD}\ (W')$

Table 4-1: ALGORITHM 1: framework IBDS.

ALGORITHM 1: framework IBDS

17: Find the value of k using 95% cumulative variance % Solve for A_t the feature importance coefficient matrix

18: $\min_{A_t} \| X_t A_t - V_{t_{(k)}} \|_F^2 + \beta ||a_{t_{i,j}}||_p^p$

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

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 τ for window W

24: Create k number of subsets $s_1, s_2, \dots, s_k$ 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: $MDSC(s_k) \leftarrow median((\widehat{r_\alpha}(x_i))_{s_k}$,

28: $MNSC(s_k) \leftarrow mean((\widehat{r_\alpha}(x_i))_{s_k}$

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

30: Create a sequence of 2k number $\tau_1, \tau_2, \dots, \tau_{2k,}\ 0 < \tau \leq \tau_{\min}(s_k))$

Table 4-1: ALGORITHM 1: framework IBDS.

ALGORITHM 1: framework IBDS
31: **For** each τ_i **do**
32: Calculate TPR s_k using τ_i
33: Sort $\tau_1, \tau_2, \ldots, \tau_{2k,}$ based on TPR
34: Select first quartile value of sorted τ list as threshold τ_{s_k}
35: **End For**
36: $\tau_{W_{ref}} \leftarrow average(\tau_{s_k}), for\ all\ k$
37: return $\tau_{W_{ref}}$
38: **end procedure**

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

$$Run\ time\ ratio, RTR(x) = \frac{Average\ run\ time\ of\ method\ x}{Average\ runtime\ of\ proposed\ \ IBDS}. \qquad \textbf{Eq. 4-42}$$

where $x = Deep\ Autoencoder, OSVM, LOF, ISF, IBDS$. 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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						**OSVM**		**LOF**		**ISF**	
		alpha=0		**alpha=0.5**		**alpha=0.95**		**Kernel Width=0.1**		**k=5**		**# of tree = window size/10**	
Win Size	**Frac (%)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**
5000	2	87.9	8.4	88.2	8.6	90.4	8.6	62.5	6.6	50.4	142.3	54.8	59.5
5000	5	83.9	8.4	85.3	8.7	82.9	8.6	63.3	6.2	50.5	146.5	60.3	56.1
5000	10	84.1	8.4	84.4	8.4	83.3	8.7	77.8	6.0	49.2	133.9	70.1	58.8
15000	2	91.3	26.2	92.7	26.1	89.2	26.9	62.6	71.0	50.1	1278.7	54.8	239.6
15000	5	86.8	26.3	81.5	26.2	79.9	27.1	73.0	69.5	50.1	1007.8	60.3	259.3
15000	10	81.2	26.4	76.4	26.3	84.0	26.9	82.0	65.6	48.6	1076.6	70.1	305.3
25000	2	89.2	45.6	87.2	46.4	89.2	45.4	62.6	204.2	50.8	3424.2	54.8	557.7
25000	5	87.6	44.2	83.1	44.5	84.1	45.4	72.8	197.8	50.3	3376.5	60.3	639.3
25000	10	78.1	44.4	80.3	44.4	79.4	45.1	81.9	189.6	49.5	3492.8	70.1	763.1
Average		**85.6**	**26.5**	**84.3**	**26.6**	**84.7**	**27.0**	**70.9**	**90.7**	**50.0**	**1564.4**	**61.8**	**326.5**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						OSVM		LOF		ISF	
		alpha=0		alpha=0.5		alpha=0.95		Kernel Width=0.1		k=5		# of tree = window size/10	
Win Size	Frac (%)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)
2000	2	59.2	3.2	62.0	3.3	59.7	3.3	62.1	1.0	53.0	9.3	54.8	15.5
2000	5	54.1	3.3	54.3	3.3	57.3	3.4	72.4	1.0	49.6	9.9	60.3	15.1
2000	10	54.8	3.3	49.6	3.2	48.8	3.4	81.6	0.9	46.3	9.7	65.5	14.9
5000	2	67.8	10.0	71.1	10.0	67.4	10.1	62.3	5.9	50.5	69.7	54.8	76.0
5000	5	58.5	9.8	61.5	9.8	56.2	10.0	72.8	5.7	49.6	70.5	60.3	75.7
5000	10	52.5	10.1	55.0	10.4	53.8	10.1	79.7	5.5	46.5	70.0	65.1	79.5
10000	2	76.9	26.0	78.3	25.6	79.9	26.1	62.7	29.0	49.7	348.1	54.8	219.7
10000	5	76.3	24.0	75.9	23.3	76.1	25.0	73.1	28.1	48.4	360.3	60.3	229.8
10000	10	68.8	25.1	66.3	24.6	69.7	25.2	80.4	26.6	45.8	359.8	65.6	243.5
Average		**63.2**	**12.7**	**63.8**	**12.6**	**63.2**	**13.0**	**71.9**	**11.5**	**48.8**	**145.3**	**60.2**	**107.7**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						OSVM		LOF		ISF	
		alpha=0		alpha=0.5		alpha=0.95		Kernel Width=0.1		k=5		# of tree = window size/10	
Win Size	Frac (%)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)
5000	2	84.1	8.7	83.0	8.4	86.4	8.8	62.4	5.9	49.5	63.1	54.8	64.9
5000	5	80.1	8.2	78.8	8.7	81.8	8.8	72.8	5.7	48.4	61.8	60.3	63.4
5000	10	76.1	8.2	79.8	8.2	79.5	8.2	81.7	5.5	45.4	60.8	62.1	66.6
15000	2	88.2	25.5	82.7	25.4	85.3	26.3	62.7	69.9	49.2	1113.6	54.8	341.0
15000	5	82.1	25.5	82.4	25.7	85.8	26.2	72.7	68.0	47.1	1123.8	60.3	361.5
15000	10	80.0	25.4	80.1	26.1	81.3	27.2	81.8	64.8	44.4	1078.9	63.0	380.0
25000	2	87.5	45.4	86.2	45.2	91.2	46.4	62.6	200.3	49.3	3747.4	54.8	850.7
25000	5	85.1	46.2	84.5	45.9	83.6	47.2	73.0	195.6	47.3	3506.9	60.3	904.3
25000	10	80.0	45.4	81.8	45.1	80.9	46.5	82.0	186.3	44.4	3520.3	66.5	950.1
Average		**82.6**	**26.5**	**82.1**	**26.5**	**84.0**	**27.3**	**72.4**	**89.1**	**47.2**	**1586.3**	**59.7**	**442.5**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						**OSVM**		**LOF**		**ISF**	
		alpha=0		**alpha=0.5**		**alpha=0.95**		**Kernel Width=0.1**		**k=5**		**# of tree = window size/10**	
Win Size	**Frac (%)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**
5000	2	86.3	8.3	87.8	8.3	85.7	8.9	62.5	6.0	49.9	59.8	54.8	48.4
5000	5	84.2	8.7	84.8	8.5	84.2	9.6	73.3	6.0	49.8	59.1	60.3	47.9
5000	10	83.5	8.5	83.6	8.5	83.4	9.0	82.0	5.7	48.9	59.4	70.1	52.5
15000	2	85.5	27.1	88.3	26.9	81.9	28.3	62.7	72.9	50.0	1315.3	54.8	224.1
15000	5	85.6	27.0	83.7	27.0	85.5	29.3	73.2	70.8	50.1	1255.5	60.3	250.2
15000	10	82.9	27.5	83.0	27.7	84.0	29.1	81.8	67.8	48.0	1201.5	70.1	284.8
25000	2	87.1	47.2	86.4	47.4	86.5	48.4	62.6	206.6	50.4	3585.2	54.8	576.6
25000	5	83.8	48.1	87.4	47.7	85.9	48.6	72.9	201.7	48.8	3643.9	60.3	646.7
25000	10	83.2	47.4	83.6	47.3	75.8	47.9	81.9	192.5	47.5	3503.2	70.1	724.1
Average		**84.7**	**27.7**	**85.4**	**27.7**	**83.7**	**28.8**	**72.5**	**92.2**	**49.3**	**1631.4**	**61.8**	**317.2**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						**OSVM**		**LOF**		**ISF**	
		alpha=0		**alpha=0.5**		**alpha=0.95**		**Kernel Width=0.1**		**k=5**		**# of tree = window size/10**	
Win Size	**Frac (%)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**	**F1**	**Run time (sec)**
5000	2	79.3	8.6	79.6	8.4	80.0	8.9	62.4	5.9	50.3	59.6	53.4	81.0
5000	5	78.3	8.9	75.1	8.7	76.9	9.1	72.7	5.8	49.5	59.3	55.3	77.6
5000	10	71.5	8.9	72.8	8.8	72.1	9.2	81.4	5.5	50.7	62.9	57.9	76.9
15000	2	83.3	27.4	83.3	27.3	83.0	29.0	62.4	70.5	50.0	1050.9	53.1	424.6
15000	5	80.0	27.4	80.8	27.5	80.6	28.5	72.8	68.6	50.3	1083.9	56.4	429.5
15000	10	72.9	27.1	74.0	27.6	73.3	28.7	81.6	65.4	48.4	1076.1	58.3	460.2
25000	2	84.5	50.9	80.5	52.3	84.2	54.3	62.5	201.1	50.3	3187.0	53.6	1069.1
25000	5	79.8	49.9	82.5	50.6	83.3	52.7	73.1	196.1	49.8	3476.0	56.0	1111.0
25000	10	76.7	54.5	77.4	54.7	77.9	54.9	81.7	186.6	48.1	3561.1	59.0	1178.7
Average		**78.5**	**29.3**	**78.4**	**29.6**	**79.0**	**30.6**	**72.3**	**89.5**	**49.7**	**1513.0**	**55.9**	**545.4**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						OSVM		LOF		ISF	
		alpha=0		alpha=0.5		alpha=0.95		Kernel Width=0.1		k=5		# of tree = window size/10	
Win Size	Frac (%)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)
5000	2	66.2	8.6	72.2	9.0	69.4	9.1	62.2	5.9	49.7	59.2	53.0	85.6
5000	5	68.3	8.5	65.7	8.5	65.0	9.1	72.7	5.7	50.0	66.7	55.4	82.4
5000	10	59.8	8.5	59.8	8.6	61.3	8.7	81.5	5.5	49.9	67.0	58.2	83.9
15000	2	79.1	28.0	80.7	28.0	76.0	29.1	62.6	70.4	49.7	800.2	53.1	480.9
15000	5	71.0	28.0	68.6	27.7	73.8	29.0	73.2	68.5	49.5	803.8	55.5	484.6
15000	10	63.9	27.6	67.5	27.8	68.7	28.9	81.8	65.2	47.1	916.3	57.8	510.3
25000	2	81.5	53.4	76.9	52.5	77.6	52.2	62.6	199.7	50.2	3242.8	53.0	1216.3
25000	5	73.5	52.5	69.7	50.0	75.3	53.4	73.1	195.0	49.1	3498.9	55.8	1251.7
25000	10	67.0	50.5	67.7	52.2	70.7	52.6	81.9	185.5	47.9	3328.7	58.0	1317.3
Average		**70.0**	**29.5**	**69.9**	**29.4**	**70.9**	**30.2**	**72.4**	**89.0**	**49.2**	**1420.4**	**55.5**	**612.6**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						OSVM		LOF		ISF	
		alpha=0		alpha=0.5		alpha=0.95		Kernel Width=0.1		k=25		# of tree = window size/10	
Win Size	Frac (%)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)
5000	2	84.7	9.4	85.6	9.5	84.7	9.5	62.4	5.9	51.1	59.3	54.8	83.7
5000	5	75.5	9.4	73.1	9.6	74.4	9.8	72.9	5.7	50.3	59.8	60.3	84.9
5000	10	75.6	8.9	73.8	8.9	76.7	9.2	81.8	5.5	48.9	59.7	70.2	89.2
15000	2	86.4	26.4	89.0	26.6	86.5	27.3	62.6	70.3	50.5	1052.9	54.8	377.5
15000	5	80.5	26.4	82.1	26.3	80.2	27.9	73.3	68.7	49.2	1007.9	60.3	397.6
15000	10	79.4	26.4	79.9	26.3	78.4	27.3	81.6	65.7	48.6	935.7	70.1	426.7
25000	2	84.3	45.8	85.1	45.9	85.4	47.0	62.7	201.3	50.1	3706.2	54.8	884.9
25000	5	75.9	46.1	80.2	45.8	78.8	46.9	73.1	196.3	49.3	3453.4	60.3	947.1
25000	10	75.6	46.2	74.7	46.0	75.5	47.8	81.7	187.1	48.9	3482.5	70.1	1018.7
Average		**79.8**	**27.2**	**80.4**	**27.2**	**80.1**	**28.1**	**72.4**	**89.6**	**49.7**	**1535.3**	**61.8**	**478.9**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						OSVM		LOF		ISF	
		alpha=0		alpha=0.5		alpha=0.95		Kernel Width=0.1		k=5		# of tree = window size/10	
Win Size	Frac (%)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)
5000	2	88.1	8.6	88.5	8.7	88.3	8.8	62.8	5.8	50.2	59.4	55.2	52.7
5000	5	85.9	8.6	85.1	8.5	82.2	8.9	72.8	5.7	49.6	59.8	60.7	53.8
5000	10	83.5	8.6	84.5	8.5	83.9	8.8	81.8	5.5	46.7	60.2	70.5	58.3
15000	2	87.7	30.2	87.5	29.4	88.9	30.3	62.9	69.7	50.0	1147.1	55.2	276.3
15000	5	85.6	29.6	85.6	29.7	80.2	31.5	73.1	68.0	48.7	1107.8	60.7	304.4
15000	10	83.6	29.5	84.7	30.9	83.6	31.6	81.6	64.7	47.0	860.8	70.5	332.4
25000	2	88.6	51.0	90.6	52.1	89.8	52.7	62.5	198.9	50.1	3587.5	55.2	698.5
25000	5	86.4	52.3	85.4	50.9	84.9	51.4	73.1	193.7	49.3	3509.9	60.7	769.0
25000	10	84.5	51.4	84.9	52.1	83.9	53.0	81.9	184.5	48.0	3183.1	70.5	856.0
Average		**86.0**	**30.0**	**86.3**	**30.1**	**85.1**	**30.8**	**72.5**	**88.5**	**48.8**	**1508.4**	**62.1**	**377.9**

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.

		Proposed Framework (Unsupervised Feature Selection + RuLSIF)						OSVM		LOF		ISF	
		alpha=0		alpha=0.5		alpha=0.95		Kernel Width=0.1		k=5		# of tree = window size/10	
Win Size	Frac (%)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)	F1	Run time (sec)
5000	2	92.6	8.3	91.9	8.2	87.9	8.4	62.7	6.0	50.6	65.8	54.1	44.7
5000	5	84.9	8.4	83.8	8.2	86.8	8.6	73.3	5.8	48.9	66.1	59.7	46.2
5000	10	81.1	8.2	81.1	8.2	82.2	8.7	82.0	5.6	46.8	66.2	69.5	51.6
10000	2	81.5	17.4	90.5	17.4	91.0	18.5	62.6	29.7	49.8	359.6	54.1	125.1
10000	5	85.3	17.7	85.3	17.3	86.8	17.8	73.2	28.8	49.5	359.1	59.7	137.0
10000	10	82.0	17.7	81.5	18.0	80.4	17.9	82.1	27.3	46.8	347.8	69.5	159.2
15000	2	89.0	27.8	86.9	27.9	90.9	29.4	62.6	36.0	50.1	1004.5	54.1	219.0
15000	5	77.9	27.2	81.3	27.7	80.6	29.0	73.0	34.7	48.9	957.1	59.7	249.6
15000	10	79.4	27.8	82.0	27.3	83.1	28.0	81.9	33.1	47.1	964.7	69.5	288.3
Average		**83.7**	**17.8**	**84.9**	**17.8**	**85.5**	**18.5**	**72.6**	**23.0**	**48.7**	**465.7**	**61.1**	**146.7**

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

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

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

		D1			D3			D4			D5		
Win Size	**Frac (%)**	**F1 (Fixed tr)**	**F1 (k)**	**Run Time (sec)**	**F1 (tr)**	**F1 (k)**	**Run Time (sec)**	**F1 (Fixed tr)**	**F1 (k)**	**Run Time (sec)**	**F1 (tr)**	**F1 (k)**	**Run Time (sec)**
5000	2	42.4	83.1	33.5	2.1	39.8	44.6	9.8	47.3	53.3	4.0	47.8	37.7
5000	5	65.0	89.8	35.8	4.2	57.8	47.4	18.5	64.3	46.8	9.3	72.6	42.0
5000	10	72.3	95.2	29.6	9.8	79.7	47.7	35.2	80.9	41.3	18.2	83.2	40.1
15000	2	47.1	80.6	59.2	2.0	30.5	80.7	9.9	45.5	71.4	4.0	50.5	51.6
15000	5	72.3	93.0	59.9	5.1	45.9	78.1	25.9	70.0	86.4	10.4	70.8	48.4
15000	10	83.4	95.6	58.1	9.9	55.1	73.4	34.5	82.0	91.0	15.6	79.7	49.0
25000	2	49.6	83.3	109.0	2.1	18.1	115.2	11.0	47.1	139.3	3.3	42.0	68.2
25000	5	76.8	83.4	110.2	5.0	36.9	109.1	23.0	70.4	125.5	8.4	65.6	76.0
25000	10	88.9	96.7	106.0	9.6	51.3	97.2	36.2	82.5	122.3	16.9	78.1	87.8
Average		**66.4**	**89.0**	**66.8**	**5.5**	**46.1**	**77.0**	**22.7**	**65.6**	**86.4**	**10.0**	**65.6**	**55.6**

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

		D6			D7			D8		
Win Size	**Frac (%)**	**F1 (tr)**	**F1 (k)**	**Run Time (sec)**	**F1 (Fixed tr)**	**F1 (k)**	**Run Time (sec)**	**F1 (tr)**	**F1 (k)**	**Run Time (sec)**
5000	2	4.3	70.3	22.4	16.0	62.7	23.2	2.2	47.8	31.9
5000	5	8.6	84.5	22.0	30.9	78.1	25.2	4.3	68.1	29.3
5000	10	18.4	92.3	19.0	48.5	88.9	27.4	9.6	82.3	24.0
15000	2	4.6	78.3	53.1	14.8	63.7	46.7	2.2	55.3	43.0
15000	5	12.1	90.4	45.0	39.8	83.1	48.6	5.7	73.4	60.2
15000	10	22.1	95.3	50.6	60.6	91.8	43.6	10.5	82.3	58.5
25000	2	4.6	76.9	70.4	13.4	65.5	67.6	2.2	57.1	81.5
25000	5	12.9	91.2	74.6	30.8	75.0	90.8	5.9	74.3	94.0
25000	10	22.3	95.8	69.4	58.7	92.2	61.0	10.8	81.1	81.6
Average		**12.2**	**86.1**	**47.4**	**34.8**	**77.9**	**48.2**	**5.9**	**69.1**	**56.0**

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

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

Frac	Method 1 (Proposed)	Method 2 (Baseline)	t-value	p-value	Difference of means	Is Significant?	Conclusion (At 95% confidence interval, mean detection rate of IBDS is greater/less than the mean detection rate of method 2)
2	IBDS	OSVM	9.082	1.73E-05	20.988	yes	20.988% greater
2	IBDS	LOF	14.391	4.84E-07	33.331	yes	33.331% greater
2	IBDS	ISF	12.527	1.26E-06	29.113	yes	29.113% greater
2	IBDS	Autoencoder	4.779	0.0005876	27.793	yes	27.793% greater
5	IBDS	OSVM	2.509	0.03529	6.359	yes	6.359% greater
5	IBDS	LOF	11.763	2.11E-06	29.648	yes	29.648% greater
5	IBDS	ISF	7.588	3.06E-05	19.707	yes	19.707% greater
5	IBDS	Autoencoder	3.110	0.01016	20.047	yes	20.047% greater
10	IBDS	OSVM	-1.888	0.09546	-5.626	no	There is not enough evidence that the mean detection rates of IBDS and OSVM are different
10	IBDS	LOF	9.378	9.57E-06	28.268	yes	28.27% greater
10	IBDS	ISF	2.816	0.01481	9.654	yes	9.65% greater
10	IBDS	Autoencoder	-1.175	0.2587	-5.856	no	There is not enough evidence that the mean detection rates of IBDS and Autoencoder are different

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.

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

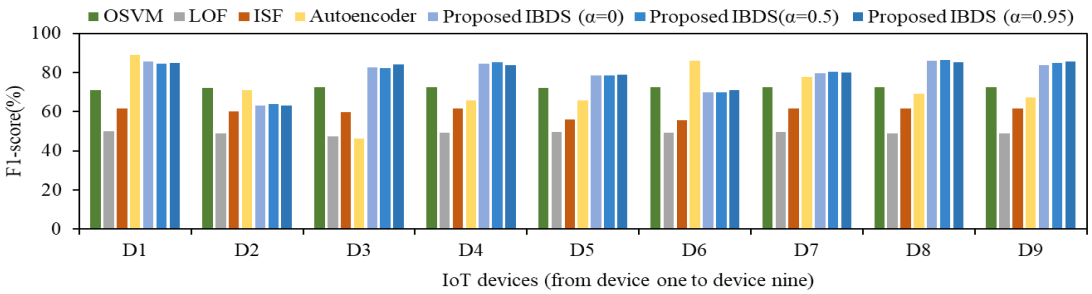

Figure 4-1: Comparison of performance in terms of F1-score of proposed IBDS framework with OSVM, LOF, and ISF for device 9 IoT devices.

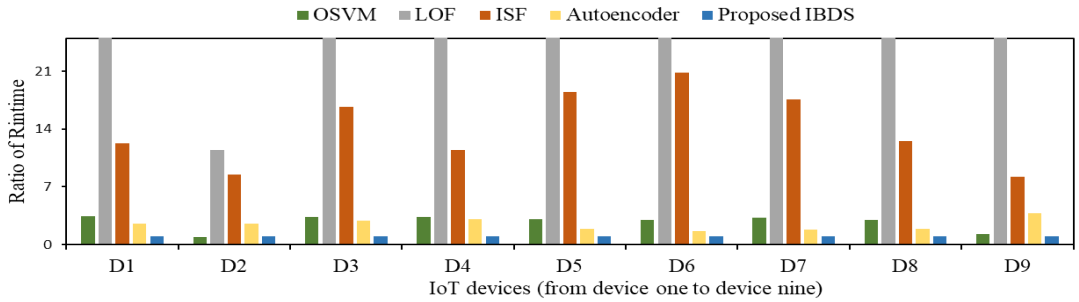

Figure 4-2: Comparison of run time ration of proposed IBDS framework with OSVM, LOF, and ISF for device 9 IoT devices.

We particularity 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.

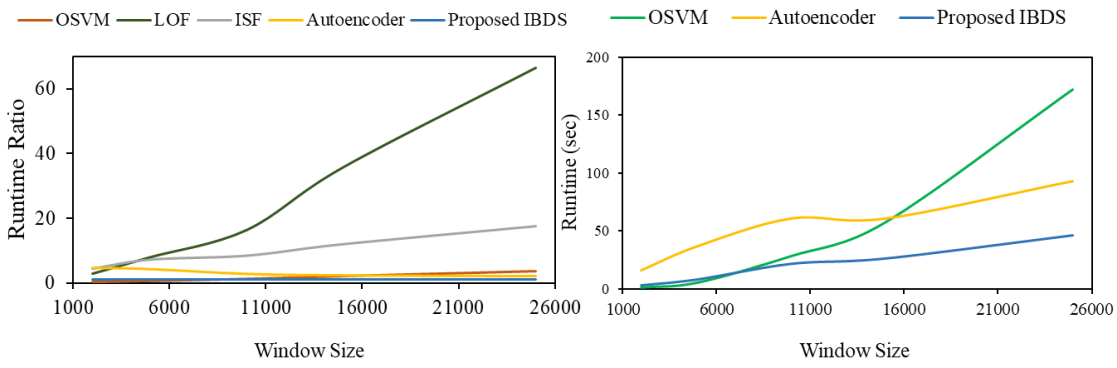

Figure 4-3: (a) Comparison of runtime ratio of proposed IBDS with OSVM, LOF, ISF, and Deep Autoencoder concerning window size (b) Comparison of average runtime in seconds of proposed IBDS with OSVM, and deep autoencoder concerning window size.

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 α-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

[1] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," 2012.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine,* vol. 17, no. 3, pp. 37-37, 1996.

[3] U. R. Acharya *et al.,* "Use of Nonlinear Features for Automated Characterization of Suspicious Ovarian Tumors Using Ultrasound Images in Fuzzy Forest Framework," *International Journal of Fuzzy Systems,* vol. 20, no. 4, pp. 1385-1402, 2018.

[4] T. Z. Tan, C. Quek, G. S. Ng, and K. Razvi, "Ovarian cancer diagnosis with complementary learning fuzzy neural network," *Artificial intelligence in medicine,* vol. 43, no. 3, pp. 207-222, 2008.

[5] U. R. Acharya *et al.,* "Ovarian tissue characterization in ultrasound: a review," *Technology in cancer research & treatment,* vol. 14, no. 3, pp. 251-261, 2015.

[6] U. R. Acharya *et al.,* "GyneScan: an improved online paradigm for screening of ovarian cancer via tissue characterization," *Technology in cancer research & treatment,* vol. 13, no. 6, pp. 529-539, 2014.

[7] K.-L. Tang, T.-H. Li, W.-W. Xiong, and K. Chen, "Ovarian cancer classification based on dimensionality reduction for SELDI-TOF data," *BMC bioinformatics,* vol. 11, no. 1, p. 109, 2010.

[8] R. Biagiotti, C. Desii, E. Vanzi, and G. Gacci, "Predicting ovarian malignancy: application of artificial neural networks to transvaginal and color Doppler flow US," *Radiology,* vol. 210, no. 2, pp. 399-403, 1999.

[9] A. Tailor, D. Jurkovic, T. Bourne, W. Collins, and S. Campbell, "Sonographic prediction of malignancy in adnexal masses using multivariate logistic regression analysis," *Ultrasound in Obstetrics & Gynecology,* vol. 10, no. 1, pp. 41-47, 1997.

[10] O. Lucidarme *et al.,* "A new computer-aided diagnostic tool for non-invasive characterisation of malignant ovarian masses: results of a multicentre validation study," *European radiology,* vol. 20, no. 8, pp. 1822-1830, 2010.

[11] Y. Zimmer, R. Tepper, and S. Akselrod, "An automatic approach for morphological analysis and malignancy evaluation of ovarian masses using B-scans," *Ultrasound in medicine & biology*, vol. 29, no. 11, pp. 1561-1570, 2003.

[12] U. R. Acharya, M. M. R. Krishnan, L. Saba, F. Molinari, S. Guerriero, and J. S. Suri, "Ovarian tumor characterization using 3D ultrasound," in *Ovarian Neoplasm Imaging*: Springer, 2013, pp. 399-412.

[13] U. R. Acharya *et al.*, "Evolutionary algorithm-based classifier parameter tuning for automatic ovarian cancer tissue characterization and classification," *Ultraschall in der Medizin-European Journal of Ultrasound*, vol. 35, no. 03, pp. 237-245, 2014.

[14] U. R. Acharya, S. V. Sree, L. Saba, F. Molinari, S. Guerriero, and J. S. Suri, "Ovarian tumor characterization and classification using ultrasound—a new online paradigm," *Journal of digital imaging*, vol. 26, no. 3, pp. 544-553, 2013.

[15] T. Hata *et al.*, "Three-dimensional ultrasonographic evaluation of ovarian tumours: a preliminary study," *Human Reproduction*, vol. 14, no. 3, pp. 858-862, 1999.

[16] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Machine Learning Proceedings 1992*: Elsevier, 1992, pp. 249-256.

[17] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 289-300, 2002.

[18] J. H. Friedman and L. C. Rafsky, "Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests," *The Annals of Statistics*, pp. 697-717, 1979.

[19] A. Hoekstra and R. P. Duin, "On the nonlinearity of pattern classifiers," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, 1996, vol. 4: IEEE, pp. 271-275.

[20] D. Conn, T. Ngun, G. Li, and C. Ramirez, "Fuzzy forests: extending random forests for correlated, high-dimensional data," 2015.

[21] G. Louppe, "Understanding random forests: From theory to practice," *arXiv preprint arXiv:1407.7502*, 2014.

[22] S. Horvath, *Weighted network analysis: applications in genomics and systems biology*. Springer Science & Business Media, 2011.

[23] D. T. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[24] N. Siddique and H. Adeli, "Neural Systems and Applications," *Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing*, pp. 159-181.

[25] M. Sarkar, "Fuzzy-rough nearest neighbor algorithms in classification," *Fuzzy Sets and Systems*, vol. 158, no. 19, pp. 2134-2152, 2007.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001.

[27] K.-L. Tang, T.-H. Li, W.-W. Xiong, and K. Chen, "Ovarian cancer classification based on dimensionality reduction for SELDI-TOF data," *BMC bioinformatics*, vol. 11, no. 1, p. 109, 2010.

[28] E. F. Petricoin *et al.*, "Use of proteomic patterns in serum to identify ovarian cancer," *The lancet*, vol. 359, no. 9306, pp. 572-577, 2002.

[29] F. Carcillo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi, "Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization," *International Journal of Data Science and Analytics*, vol. 5, no. 4, pp. 285-300, 2018.

[30] W. Cheng *et al.*, "Ranking causal anomalies for system fault diagnosis via temporal and dynamical analysis on vanishing correlations," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 4, p. 40, 2017.

[31] E. Trunzer *et al.*, "Failure mode classification for control valves for supporting data-driven fault detection," in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2017: IEEE, pp. 2346-2350.

[32] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 275-300, 2004.

[33] A. F. Costa, Y. Yamaguchi, A. J. M. Traina, and C. Faloutsos, "Modeling temporal activity to detect anomalous behavior in social media," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 4, p. 49, 2017.

[34] Y. Gao, Y. Ma, and D. Li, "Anomaly detection of malicious users' behaviors for web applications based on web logs," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 2017: IEEE, pp. 1352-1355.

[35] S. Li, M. Shao, and Y. Fu, "Multi-view low-rank analysis with applications to outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 3, p. 32, 2018.

[36] U. Murad and G. Pinkas, "Unsupervised profiling for identifying superimposed fraud," in *European Conference on Principles of Data Mining and Knowledge Discovery*, 1999: Springer, pp. 251-261.

[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.

[38] W. Xie, F. Zhu, J. Xiao, and J. Wang, "Social Network Monitoring for Bursty Cascade Detection," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 12, no. 4, p. 40, 2018.

[39] J. Liang, D. Ajwani, P. K. Nicholson, A. Sala, and S. Parthasarathy, "Prioritized relationship analysis in heterogeneous information networks," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 12, no. 3, p. 29, 2018.

[40] T. Wang, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and J. Cao, "Big data reduction for a smart city's critical infrastructural health monitoring," *IEEE Communications Magazine,* vol. 56, no. 3, pp. 128-133, 2018.

[41] L. Zadvinskiy, C. Wheeler, G. Gardner, and A. Flower, "Using abrupt change detection to categorize glucose variability of Type 1 diabetes patients," in *2017 Systems and Information Engineering Design Symposium (SIEDS)*, 2017: IEEE, pp. 243-247.

[42] C. C. Aggarwal, "A framework for diagnosing changes in evolving data streams," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003: ACM, pp. 575-586.

[43] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*, 2004: Springer, pp. 286-295.

[44] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statistical Analysis and Data Mining: The ASA Data Science Journal,* vol. 5, no. 2, pp. 114-127, 2012.

[45] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742,* 2007.

[46] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993.

[47] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR),* vol. 46, no. 4, p. 44, 2014.

[48] Y. Kawahara and M. Sugiyama, "Change-point detection in time-series data by direct density-ratio estimation," in *Proceedings of the 2009 SIAM International Conference on Data Mining*, 2009: SIAM, pp. 389-400.

[49] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks,* vol. 43, pp. 72-83, 2013.

[50] Y. Kawahara, T. Yairi, and K. Machida, "Change-point detection in time-series data based on subspace identification," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007: IEEE, pp. 559-564.

[51] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[52] H. Yanai, K. Takeuchi, and Y. Takane, "Singular Value Decomposition (SVD)," in *Projection Matrices, Generalized Inverse Matrices, and Singular Value Decomposition*: Springer, 2011, pp. 125-149.

[53] S. Puntanen, G. P. Styan, and J. Isotalo, "Eigenvalue Decomposition," in *Matrix Tricks for Linear Statistical Models*: Springer, 2011, pp. 357-390.

[54] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015: ACM, pp. 935-944.

[55] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th International Symposium on Wearable Computers*, 2012: IEEE, pp. 108-109.

[56] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Demon: Mining and monitoring evolving data," *IEEE Transactions on Knowledge and Data Engineering,* vol. 13, no. 1, pp. 50-63, 2001.

[57] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics,* vol. 2, no. 4, pp. 433-459, 2010.

[58] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics,* vol. 22, no. 1, pp. 79-86, 1951.

[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*

Dublin Philosophical Magazine and Journal of Science, vol. 50, no. 302, pp. 157-175, 1900.

[60] V. Vapnik and V. Vapnik, "Statistical learning theory Wiley," *New York*, pp. 156-160, 1998.

[61] M. Yoshida *et al.*, "Luminosity Functions of Lyman Break Galaxies at z~ 4 and z~ 5 in the Subaru Deep Field," *The Astrophysical Journal*, vol. 653, no. 2, p. 988, 2006.

[62] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd international conference on Machine learning*, 2006: ACM, pp. 97-104.

[63] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007: ACM, pp. 667-676.

[64] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*, 2006: Citeseer.

[65] D. Omoifo, "Obstacle detection in autonomous vehicles using deep learning," 2018.

[66] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, no. 2, pp. 76-79, 2017.

[67] Y. Meidan *et al.*, "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12-22, 2018.

[68] A. Marzano *et al.*, "The Evolution of Bashlite and Mirai IoT Botnets," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018: IEEE, pp. 00813-00818.

[69] K. Angrishi, "Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets," *arXiv preprint arXiv:1702.03681*, 2017.

[70] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80-84, 2017.

[71] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017, pp. 1092-1110.

[72] S. García, A. Zunino, and M. Campo, "Survey on network-based botnet detection methods," *Security and Communication Networks*, vol. 7, no. 5, pp. 878-903, 2014.

[73] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani, "A taxonomy of botnet detection techniques," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, 2010, vol. 2: IEEE, pp. 158-162.

[74] S. García, A. Zunino, and M. Campo, "Botnet behavior detection using network synchronism," in *Privacy, Intrusion Detection and Response: Technologies for Protecting Networks*: IGI Global, 2012, pp. 122-144.

[75] R. Hallman, J. Bryan, G. Palavicini, J. Divita, and J. Romero-Mariona, "IoDDoS the internet of distributed denial of sevice attacks," in *2nd International Conference on Internet of Things, Big Data and Security. SCITEPRESS*, 2017, pp. 47-58.

[76] H. Dong and D. Peng, "Research on abnormal detection of ModbusTCP/IP protocol based on one-class SVM," in *2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2018: IEEE, pp. 398-403.

[77] J. Medková, M. Husák, M. Vizváry, and P. Čeleda, "Honeypot testbed for network defence strategy evaluation," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, 2017: IEEE, pp. 887-888.

[78] H. Wafi, A. Fiade, N. Hakiem, and R. B. Bahaweres, "Implementation of a modern security systems honeypot Honey Network on wireless networks," in *Young Engineers Forum (YEF-ECE), 2017 International*, 2017: IEEE, pp. 91-96.

[79] D. Fraunholz, M. Zimmermann, and H. D. Schotten, "An adaptive honeypot configuration, deployment and maintenance strategy," in *Advanced Communication Technology (ICACT), 2017 19th International Conference on*, 2017: IEEE, pp. 53-57.

[80] P. Jaikumar and A. C. Kak, "A graph-theoretic framework for isolating botnets in a network," *Security and communication networks*, vol. 8, no. 16, pp. 2605-2623, 2015.

[81] A. Shabtai, D. Potashnik, Y. Fledel, R. Moskovitch, and Y. Elovici, "Monitoring, analysis, and filtering system for purifying network traffic of known and unknown malicious content," *Security and Communication Networks*, vol. 4, no. 8, pp. 947-965, 2011.

[82] L. E. Menten, A. Chen, and D. Stiliadis, "NoBot: embedded malware detection for endpoint devices," *Bell Labs Technical Journal*, vol. 16, no. 1, pp. 155-170, 2011.

[83] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri, "A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology," in *Communications (ICC), 2016 IEEE International Conference on*, 2016: IEEE, pp. 1-6.

[84] S. Ali, G. Wang, R. L. Cottrell, and T. Anwar, "Detecting Anomalies from End-to-End Internet Performance Measurements (PingER) Using Cluster Based Local Outlier Factor," in *Ubiquitous Computing and Communications (ISPA/IUCC), 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on*, 2017: IEEE, pp. 982-989.

[85] S. Su *et al.*, "N2DLOF: A New Local Density-Based Outlier Detection Approach for Scattered Data," in *High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2017 IEEE 19th International Conference on*, 2017: IEEE, pp. 458-465.

[86] A. D. Landress, "A hybrid approach to reducing the false positive rate in unsupervised machine learning intrusion detection," in *SoutheastCon, 2016*, 2016: IEEE, pp. 1-6.

[87] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach," *Computer Communications*, vol. 98, pp. 52-71, 2017.

[88] M. Ozcelik, N. Chalabianloo, and G. Gur, "Software-Defined Edge Defense Against IoT-Based DDoS," in *2017 IEEE International Conference on Computer and Information Technology (CIT)*, 2017: IEEE, pp. 308-313.

[89] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalis—A System for Knowledge-Driven Adaptable Intrusion Detection for the Internet of Things," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, 2017: IEEE, pp. 656-666.

[90] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *arXiv preprint arXiv:1710.00811*, 2017.

[91] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010: ACM, pp. 333-342.

[92] H. Huang, S. Yoo, and S. P. Kasiviswanathan, "Unsupervised feature selection on data streams," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015: ACM, pp. 1031-1040.

[93] E. Anderson, Z. Bai, and J. Dongarra, "Generalized QR factorization and its applications," *Linear Algebra and Its Applications,* vol. 162, pp. 243-271, 1992.

[94] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997.

[95] M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, and M. Sugiyama, "Relative density-ratio estimation for robust distribution comparison," *Neural computation,* vol. 25, no. 5, pp. 1324-1370, 2013.

[96] A. Anees, J. Aryal, M. M. O'Reilly, and T. J. Gale, "A relative density ratio-based framework for detection of land cover changes in MODIS NDVI time series," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 9, no. 8, pp. 3359-3371, 2016.

[97] M. Sugiyama and M. Kawanabe, *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.

[98] L. Zadvinskiy, C. Wheeler, G. Gardner, and A. Flower, "Using abrupt change detection to categorize glucose variability of Type 1 diabetes patients," in *Systems and Information Engineering Design Symposium (SIEDS), 2017*, 2017: IEEE, pp. 243-247.

[99] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089,* 2018.

[100] C. H. Bischof and P. T. P. Tang, *Robust incremental condition estimation*. University of Tennessee. Computer Science Department, 1991.