

# **A Genetic Algorithm-based Local Outlier Factor for Efficient Big Data Stream Processing**

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Omar S. Alghushairy

Major Professor: Xiaogang Ma, Ph.D.


Committee Members: Terence Soule, Ph.D.; Frederick Sheldon, Ph.D.; Jia Song, Ph.D.

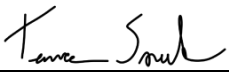
Department Administrator: Terence Soule, Ph.D.

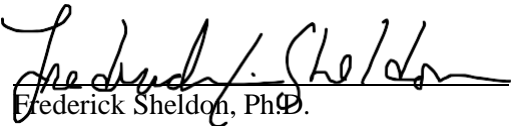
May 2021

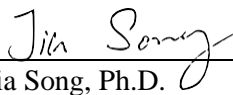
## Authorization to Submit Dissertation

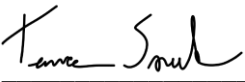
This dissertation of Omar S. Alghushairy, submitted for the degree of Doctor of Philosophy with a Major in Computer Science and titled “A Genetic Algorithm-based Local Outlier Factor for Efficient Big Data Stream Processing,” has been reviewed in final form. Permission, as indicated by the signatures and dates below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor:  Date: 04/06/2021  
Xiaogang Ma, Ph.D.

Committee Members:  Date: April 7, 2021  
Terence Soule, Ph.D.

 Date: 6 April 2021  
Frederick Sheldon, Ph.D.

 Date: 04/06/2021  
Jia Song, Ph.D.

Department Administrator:  Date: April 7, 2021  
Terence Soule, Ph.D.

## Abstract

Interest in outlier detection methods is increasing because detecting outliers is an important operation for many applications such as detecting fraud transactions in credit card, network intrusion detection and data analysis in different domains. We are now in the big data era, and an important type of big data is data stream. With the increasing necessity for analyzing high-velocity data streams, it becomes difficult to apply older outlier detection methods efficiently. Local Outlier Factor (LOF) is a well-known outlier algorithm. A major challenge of LOF is that it requires the entire dataset and the distance values to be stored in memory. Another issue with LOF is that it needs to be recalculated from the beginning if any change occurs in the dataset. This research proposes a novel local outlier detection algorithm for data streams, called Genetic-based Incremental Local Outlier Factor (GILOF). Moreover, we further improved the GILOF performance in data streams by proposing a new calculation method for LOF, called Local Outlier Factor by Reachability distance (LOFR). The improved algorithm for local outlier detection in data stream is called the Genetic-based Incremental Local Outlier Factor by Reachability distance (GILOFR). The GILOF and GILOFR algorithms work without any previous knowledge of data distribution, and they are able to execute in limited memory. The outcomes of our experiments with various real-world datasets demonstrate that the proposed algorithms have very good performance in execution time and accuracy of outlier detection.

## Acknowledgements

Alhamdulillah, I thank the GOD for his greatness and for giving me the ability and knowledge to complete and finalize this dissertation. This work could not be done without the GOD blessings.

First and foremost, I would like to express my sincere gratitude for my major professor Prof. Marshall (Xiaogang) Ma. It has been an honor to be a PhD student under your supervision. Thank you for your help, support, patience, motivation, and guidance. The completion of this dissertation would not be done without your constant availability and expert professional support. I would also like to thank Prof. Terence Soule who helped and advised me a lot. Your support and assistance had a great impact on reaching this achievement.

Moreover, I would like to thank two other members on my dissertation committee, Prof. Frederick Sheldon and Prof. Jia (Cindy) Song for their insightful comments, positive motivation, and encouragement.

My appreciation also goes to all faculty members, and colleagues in the Department of Computer Science at University of Idaho. It has been a pleasure to work in this friendly environment.

Furthermore, I would like to express my gratitude to my family, parents (Saleh and Hanan), wife (Shaza), children (Layan and Saleh), sister (Ghudir), and brothers for their endless trust, encouragement, support, help and love in this journey.

Finally, I would like to thank and express my gratitude to the Saudi Arabia Culture Mission, University of Jeddah and the government of the Kingdom of Saudi Arabia for providing an opportunity and funding to complete this dissertation.

## **Dedication**

I am dedicating this dissertation to my parents, wife, children, sister, and brothers.

## Table of Contents

<b>Authorization to Submit Dissertation .....</b>	<b>ii</b>
<b>Abstract .....</b>	<b>iii</b>
<b>Acknowledgements .....</b>	<b>iv</b>
<b>Dedication.....</b>	<b>v</b>
<b>Table of Contents.....</b>	<b>vi</b>
<b>List of Tables .....</b>	<b>x</b>
<b>List of Figures .....</b>	<b>xi</b>
<b>Chapter 1: Introduction .....</b>	<b>1</b>
1.1 Definition of Outlier Detection.....	1
1.2 Big Data .....	2
1.2.1 Data Stream .....	2
1.2.2 Data Modes .....	3
1.2.3 FAIR Data .....	3
1.3 Data Storage.....	4
1.3.1 Data Storage Devices .....	4
1.3.2 Data Storage Technologies.....	5
1.3.3 Impacts of Big Data Storage.....	7
1.4 Significance and Contribution .....	8
1.5 Research Questions.....	9
1.6 Structure of the Dissertation .....	9
<b>Chapter 2: Background and Statement of Research Challenge.....</b>	<b>11</b>
2.1 Background of Local Outlier Detection in Big Data Streams .....	11
2.1.1 Data Stream Processing .....	11
2.1.2 Outlier Detection Approaches .....	12
2.1.3 Local Outlier Detection .....	13
2.1.4 Incremental Local Outlier Factor (ILOF).....	14

2.1.5 <i>Memory Efficient Incremental Local Outlier Factor (MILOF)</i> .....	15
2.1.6 <i>Density Saummarization Incremental Local Outlier Factor (DILOF)</i> .....	16
2.2 Definition of Research Challenge.....	16
2.3 Datasets.....	17
2.3.1 <i>UCI Vowel Dataset</i> .....	18
2.3.2 <i>UCI Pendigit Dataset</i> .....	18
2.3.3 <i>KDD Cup 99 SMTP Dataset</i> .....	19
2.3.4 <i>KDD Cup 99 HTTP Dataset</i> .....	19
2.4 Genetic Algorithm.....	19
2.4.1 <i>Steps in Genetic Algorithm</i> .....	20
2.5 Conclusion.....	23
<b>Chapter 3: The State-Of-The-Art Algorithms in Local Outlier Detection</b> .....	<b>24</b>
3.1 Introduction.....	24
3.2 Outlier Detection Background.....	25
3.2.1 <i>Statistical Outlier Detection</i> .....	26
3.2.1.1 Parametric Method.....	26
3.2.1.2 Non-parametric Method.....	27
3.3 Literature Review Methodology.....	28
3.4 Literature Review Results.....	30
3.4.1 <i>Local Outlier Detection in Static Environment</i> .....	31
3.4.1.1 Local Outlier Factor (LOF).....	31
3.4.1.2 Connectivity-Based Outlier Factor (COF).....	35
3.4.1.3 Local Correlation Integral (LOCI).....	35
3.4.1.4 Approximate Local Correlation Integral (aLOCI).....	36
3.4.1.5 Cluster-Based Local Outlier Factor (CBLOF).....	36
3.4.1.6 Influenced Outlierness (INFLO).....	36
3.4.1.7 Local Outlier Probability (LoOP).....	37

3.4.1.8 Local Density Cluster-Based Outlier Factor (LDCOF) .....	37
3.4.1.9 Other Local Outlier Detection Algorithms .....	39
3.4.2 <i>Local Outlier Detection in Stream Environment</i> .....	41
3.4.2.1 Overviwe of Local Outlier Detection Algorithms in Stream Environment .....	42
3.5 Applications of Outlier Detection .....	46
3.5.1 <i>Fraud Detection</i> .....	46
3.5.2 <i>Intrusion Detection</i> .....	47
3.5.3 <i>Outlier Detection in Medical applications</i> .....	48
3.6 Computational Complexity.....	48
3.7 Strengths and Weaknesses.....	49
3.7.1 <i>Nearest Neighbor-Based Technique</i> .....	49
3.7.2 <i>Clustering-Based Technique</i> .....	50
3.8 Conclusion.....	51
<b>Chapter 4: Genetic-Based Incremental Local Outlier Factor (GILOF) Algorithm.....</b>	<b>52</b>
4.1 Introduction.....	52
4.2 Methodology.....	53
4.2.1 <i>Detection Phase</i> .....	55
4.2.1.1 Incremental Local Outlier Factor (ILOF).....	55
4.2.1.2 Skipping Scheme .....	56
4.2.2 <i>Summarization Phase</i> .....	56
4.2.2.1 Genetic Density Summarization .....	56
4.3 Concept Drift .....	57
4.4 Experment.....	60
4.4.1 <i>Experment Procedures</i> .....	60
4.5 Results Discussion.....	61
4.5.1 <i>Accuracy of Outlier Detection</i> .....	61
4.5.1.1 Accuracy of Outlier Detection in Large Window Size.....	62



4.5.2 <i>Execution Time</i> .....	63
4.6 Conclusion .....	75
<b>Chapter 5: Improving the Algorithm of Genetic-based Incremental Local Outlier Factor for Network Intrusion Detection .....</b>	<b>77</b>
5.1 Introduction.....	77
5.2 Network Intrusion Detection System (NIDS).....	78
5.3 Methodology.....	79
5.3.1 <i>Local Outlier Factor by Reachability Distance (LOFR)</i> .....	79
5.3.2 <i>Genetic-based Incremental Local Outlier Fator by Reachability Distance (GILOFR)</i> .....	81
5.3.2.1 Genetic Density Summarization .....	82
5.3.2.2 Skipping Scheme .....	84
5.4 Experment.....	84
5.5 Results Discussion .....	85
5.5.1 <i>Accuracy of Outlier Detection</i> .....	85
5.5.2 <i>Execution Time</i> .....	86
5.6 Conclusion .....	99
<b>Chapter 6: Synthesis and Conclusions .....</b>	<b>100</b>
6.1 Summary of the Key Research Activities and Outcomes .....	100
6.2 Summary of answers to research questions .....	102
6.3 Main Conclusion .....	105
6.4 List of Publications .....	106
6.4.1 <i>As a First Author</i> .....	106
6.4.2 <i>As a Co-Author</i> .....	108
<b>References .....</b>	<b>109</b>
<b>Appendix A - Experiment's performance results with different hyperparameters .....</b>	<b>122</b>

## List of Tables

Table 2.1. The Real-World Datasets Features .....	18
Table 3.1. Summary of the most-cited local outlier algorithms in a static environment .....	32
Table 3.2. Previous works summary for LOF in a stream environment .....	46
Table 4.1. The performance results of all algorithms for unnormalized UCI Vowel dataset .....	73
Table 4.2. The performance results of all algorithms for normalized UCI Vowel dataset .....	73
Table 4.3. The performance results of all algorithms for unnormalized UCI Pendigit dataset .....	73
Table 4.4. The performance results of all algorithms for normalized UCI Pendigit dataset .....	74
Table 4.5. The performance results of all algorithms for unnormalized KDD Cup99 SMTP dataset ..	74
Table 4.6. The performance results of all algorithms for normalized KDD Cup99 SMTP dataset .....	74
Table 4.7. The performance results of all algorithms for unnormalized KDD Cup99 HTTP dataset ..	75
Table 4.8. The performance results of all algorithms for normalized KDD Cup99 HTTP dataset .....	75
Table 5.1. The performance results of GILOFR and GILOF algorithms for unnormalized UCI Vowel dataset .....	95
Table 5.2. The performance results of GILOFR and GILOF algorithms for normalized UCI Vowel dataset .....	95
Table 5.3. The performance results of GILOFR and GILOF algorithms for unnormalized UCI Pendigit dataset .....	96
Table 5.4. The performance results of GILOFR and GILOF algorithms for normalized UCI Pendigit dataset .....	96
Table 5.5. The performance results of GILOFR and GILOF algorithms for unnormalized KDD Cup 99 SMTP dataset .....	97
Table 5.6. The performance results of GILOFR and GILOF algorithms for normalized KDD Cup 99 SMTP dataset .....	97
Table 5.7. The performance results of GILOFR and GILOF algorithms for unnormalized KDD Cup 99 HTTP dataset .....	98
Table 5.8. The performance results of GILOFR and GILOF algorithms for normalized KDD Cup 99 HTTP dataset .....	98
Table 6.1. The direction to the research question answers .....	104

## List of Figures

Figure 1.1. A two-dimensional example illustrating the outlier's categories, where $p_2$ and $p_3$ are global outliers and $p_1$ is a local outlier .....	1
Figure 2.1. The Key definitions of LOF algorithm.....	15
Figure 2.2. Two-point crossover.....	22
Figure 3.1. The search strategy flowchart for selecting articles .....	29
Figure 3.2. Summary of references used in the literature review for static data .....	32
Figure 3.3. The reachability distance for different data points $p$ with regard to $o$ , when $k$ equals 5 ....	33
Figure 3.4. Comparison between the LOF and the INFLO .....	38
Figure 3.5. Summary of references used in the literature review for streaming data .....	47
Figure 4.1. GILOF process for a data stream in two dimensional from time $T_0$ to $T_{current}$ .....	55
Figure 4.2. GDS flowchart.....	59
Figure 4.3. The accuracy of outlier detection for all datasets when $W = 1000$ .....	64
Figure 4.4 The comparisons of outlier detection accuracy for Unnormalized UCI VOWEL datasets.	65
Figure 4.5 The comparisons of outlier detection accuracy for normalized UCI VOWEL datasets.....	65
Figure 4.6 The comparisons of outlier detection execution time for unnormalized UCI VOWEL datasets.....	66
Figure 4.7 The comparisons of outlier detection execution time for normalized UCI VOWEL datasets.....	66
Figure 4.8 The comparisons of outlier detection accuracy for unnormalized UCI Pendigit datasets...	67
Figure 4.9 The comparisons of outlier detection accuracy for normalized UCI Pendigit datasets.....	67
Figure 4.10. The comparisons of outlier detection execution time for unnormalized UCI Pendigit datasets.....	68
Figure 4.11. The comparisons of outlier detection execution time for normalized UCI Pendigit datasets.....	68
Figure 4.12. The comparisons of outlier detection accuracy for unnormalized KDD Cup99 SMTP datasets .....	69
Figure 4.13. The comparisons of outlier detection accuracy for normalized KDD Cup99 SMTP datasets.....	69

Figure 4.14. The comparisons of outlier detection execution time for unnormalized KDD Cup 99 SMTP dataset.....	70
Figure 4.15. The comparisons of outlier detection execution time for normalized KDD Cup 99 SMTP dataset .....	70
Figure 4.16. The comparisons of outlier detection accuracy for unnormalized KDD Cup99 HTTP datasets.....	71
Figure 4.17. The comparisons of outlier detection accuracy for normalized KDD Cup99 HTTP datasets.....	71
Figure 4.18. The comparisons of outlier detection execution time for unnormalized KDD Cup 99 HTTP dataset.....	72
Figure 4.19. The comparisons of outlier detection execution time for normalized KDD Cup 99 HTTP dataset .....	72
Figure 5.1. The reachability distance for different data points ( $p$ ) with regard to $o$ , when $k$ equals 5 .	80
Figure 5.2. An accuracy comparison of outlier detection for unnormalized UCI Vowel dataset.....	87
Figure 5.3. An accuracy comparison of outlier detection for unnormalized UCI Vowel dataset.....	87
Figure 5.4. An accuracy comparison of outlier detection for unnormalized UCI Pendigit dataset .....	88
Figure 5.5. An accuracy comparison of outlier detection for normalized UCI Pendigit dataset .....	88
Figure 5.6. An accuracy comparison of outlier detection for unnormalized KDD Cup 99 SMTP dataset .....	89
Figure 5.7. An accuracy comparison of outlier detection for normalized KDD Cup 99 SMTP dataset .....	89
Figure 5.8. An accuracy comparison of outlier detection for unnormalized KDD Cup 99 HTTP dataset .....	90
Figure 5.9. An accuracy comparison of outlier detection for normalized KDD Cup 99 HTTP dataset .....	90
Figure 5.10. The execution time comparison of outlier detection for unnormalized UCI Vowel dataset .....	91
Figure 5.11. The execution time comparison of outlier detection for normalized UCI Vowel dataset.....	91
Figure 5.12. The execution time comparison of outlier detection for unnormalized UCI Pendigit dataset .....	92
Figure 5.13. The execution time comparison of outlier detection for normalized UCI Pendigit dataset .....	92

Figure 5.14. The execution time comparison of outlier detection for unnormalized KDD Cup99 SMTP dataset datasets .....	93
Figure 5.15. The execution time comparison of outlier detection for normalized KDD Cup99 SMTP dataset datasets .....	93
Figure 5.16. The execution time comparison of outlier detection for unnormalized KDD Cup99 HTTP dataset .....	94
Figure 5.17. The execution time comparison of outlier detection for normalized KDD Cup99 HTTP dataset .....	94
Figure 6.1. The overall design and workflow for the methodology .....	103
Figure 6.2. The dissertation division map.....	105

## Chapter 1: Introduction

"Data Storage." In: *Schintler L., McNeely C. (eds) Encyclopedia of Big Data*. Springer, Cham. DOI: [https://doi.org/10.1007/978-3-319-32001-4\\_323-1](https://doi.org/10.1007/978-3-319-32001-4_323-1).

"An Efficient Local Outlier Factor for Data Stream Processing: A Case Study" Forthcoming in *the 7th International Conference on Computational Science and Computational Intelligence 2020*, IEEE.

### 1.1 Definition of outlier detection

Outlier detection attempts to distinguish a data point that is distinct from the rest of the given data. Outliers occur during a procedure or as a consequence of an error of measurement [1]. By detecting outliers, essential information can be obtained to make better decisions in various applications, such as fraud transactions in credit card and intrusion detection [2]. Outlier detection techniques have been widely used in machine learning and data mining to extract information and to clean data, for example, in various domains for the purpose of decision-making, clustering, classification, and identifying frequent patterns [3, 4].

Outliers have two categories: global outliers and local outliers. If the data point  $p_0$  is far away from other data points, it is considered a global outlier [5]. Local outlier is a data point that is outlier with respect to its  $k$ -nearest neighbors. LOF introduced the idea of local outlier. It is considered a means

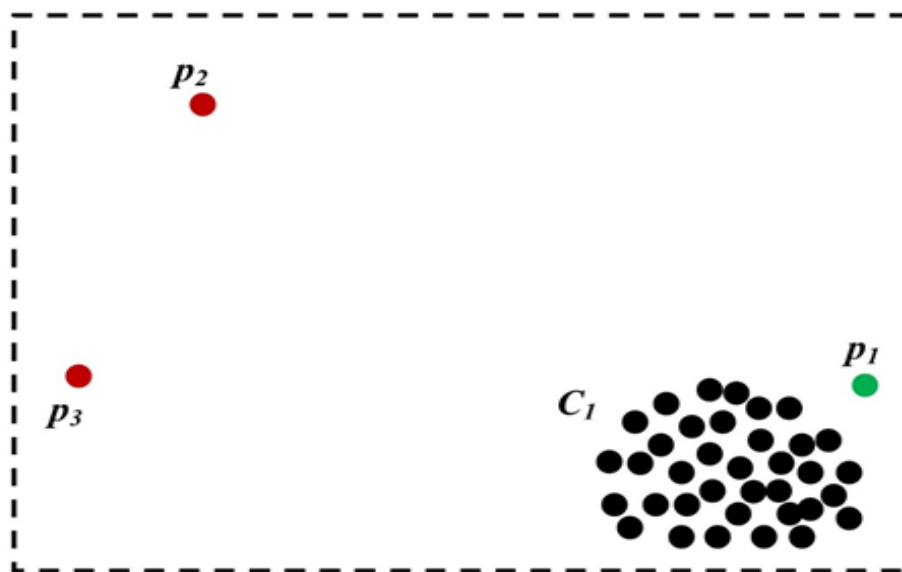


Figure 1.1. A two-dimensional example illustrating the outlier's categories, where  $p_2$  and  $p_3$  are global outliers and  $p_1$  is a local outlier.

of density-based outlier detection because it makes a comparison between the density of data points in the dataset and their local neighbors. To determine the local outlier score, LOF assigns a measured degree of the density of data points and their local neighbors (Figure 1.1).

## **1.2 Big Data**

Big data is a set of data of a size that exceeds the ability of regular databases to process, store, transfer, manage, share and analyze within an acceptable period of time [6]. Big data comes in three different modes: structured data, semi-structured data, and unstructured data [7]. The most important type of big data is the data stream, which has the characteristics of volume, velocity, variety, value, and variability. Therefore, it is not possible to process big data by traditional methods. Currently, there is an urgent need to develop new algorithms of processing and managing big data.

### *1.2.1 Data Stream*

A data stream is a collection of continuous data processed to collect knowledge and extract information [8]. Data streams represent big data as primary sources by having various applications in different properties such as volume, velocity, variety, value, and veracity [9, 10]. Volume refers to the large volume of data assembled and analyzed. Velocity involves the pace at which data between various systems and devices are generated and transported. Variety applies multiple types of data that may be used to obtain the required knowledge or performance; it involves the data modes of structured, unstructured, and semi-structured data [7, 10]. Value refers to the advantages in extracting the information from the big data. Lastly, veracity involves the quality of the data for precision, integrity, confidence, protection, and reliability. Due to the nature of the data stream with regard to these five significant data properties, data stream processing requires various methods to evaluate the data points in the data stream environment.

### 1.2.2 *Data Modes*

There are various modes (forms) of data that are stored, such as texts, numbers, videos, etc. These data can be divided into the following three categories: structured, unstructured, and semi-structured data. Structured data is considered high-level data that is in an organized form, such as data in an Excel sheet. For example, a university database has around half a million pieces of information for about 20 thousand students, which contain names, phone numbers, addresses, majors, and other data. Unstructured data is random and disorganized data, for example, data that is presented on a social network, such as text and multimedia data. Various unstructured data are posted to social media platforms like Twitter and YouTube every day. Semi-structured data is provided by several types of data combined to represent the data in a specific pattern or structure. For example, information about a user's call contains an entity of information based on the logs of the call center. However, not all the data is structured, such as a complaint recorded in audio format, which is unstructured, so it is hard to be synthesized in data storage [11].

### 1.2.3 *FAIR Data*

FAIR data is a new point of view on data management, which follows the guidelines of findability, accessibility, interoperability, and reusability [12]. FAIR data focuses on two principles which enhance machine's ability for finding and using data automatically and supporting the data reuse via humans.

Findability is based on placing the data with its metadata in searchable and global identifiers; then looking for data through several links on the World Wide Web should be possible. Accessibility is based on ensuring easy access to data and its metadata [13] through the Internet by an authorized person or a machine. Metadata should be made accessible even if the data is not accessible. Interoperability is based on containing qualified references for both data and metadata and by representing the records informal, shareable, and machine-readable language. Reusability is based on detailed information of



metadata with accessible license for suitable citation to the data. In addition, software tools and other related provenance information should also be accessible to support data reuse.

### **1.3 Data Storage**

Data storage means storing and archiving data in electronic storage devices that are dedicated for preservation, where data can be accessed and used at any time. Storage devices are hardware that are used for reading and writing data through a storage medium. Storage media are physical materials for storing and retrieving data. The popular data storage devices are hard drives, flash drives, and cloud storage. The term big data reflects not only the massive volume of data but also the increased velocity and variety in data generation and collection, for example, the massive amounts of digital photos shared on the Web, social media networks, and even the Web search records. Many conventional documents such as books, newspapers, and blogs can also be data sources in the digital world. Storing big data in appropriate ways will greatly support data discovery, access, and analytics. Accordingly, various data storage devices and technologies have been developed to increase the efficiency in data management and enable information extraction and knowledge discovery from data. In domain-specific fields, a lot of scientific researches has been conducted to tackle the specific requirements on data collection, data formatting, and data storage, which have also generated beneficial feedback to computer science.

Data storage is a key step in the data science life cycle. At the early stage of the cycle, well organized data will provide strong support to the research program where the data is collected. At the late stage, the data can be shared and made persistently reusable to other users. The FAIR data principles that already mentioned above (findable, accessible, interoperable, and reusable) provide a set of guidelines for data storage.

#### *1.3.1 Data Storage Devices*

There are many different types of devices that store data in digital forms, which have the fundamental capacity measurement unit called bit, and every eight bits are equal to one byte. Often, the

data storage device is measured in megabytes (MB), gigabytes (GB), terabytes (TB), and other bigger units. The data storage devices are categorized into two types based on their characteristics: the primary storage and the secondary storage.

Primary storage devices such as cache memory, random-access memory (RAM), and read-only memory (ROM) are connected to a central processing unit (CPU) that reads and executes instructions and data stored on them. Cache memory is very fast memory, which is used as the buffer between the CPU and RAM. RAM is temporary memory, which means the content of stored data is lost once the power is turned off. ROM is nonvolatile memory, so the data stored on it cannot be changed because it has become permanent data [14]. In general, these memories have limited capacity, where it is difficult to handle big data streaming.

Secondary storage such as hard disk drive (HDD), solid-state drive (SSD), server, CD, and DVD are external data storage that are not connected to the central processing unit [14]. This type of data storage device is usually used to increase computer capacity. Secondary storage is nonvolatile, and the data can be retained. HDD stores the data in the magnetic platter, and it uses the mechanical spindle to read and write data. The operating system identifies the paths and sectors of data stored on the platters. SSD is faster than HDD because it is a flash drive, which stores data in microchips and has no mechanical parts. Also, SSD is smaller in size, is less in weight, and is energy-efficient in comparison with HDD.

### *1.3.2 Data Storage Technologies*

In recent years, data has grown fast and has become massive. With so many data generating sources, there is an urgent need to provide technologies that can deal with the storage of big data. This section provides an overview of well-known data storage technologies that are able to manipulate large volumes of data, such as relational database, NoSQL database, distributed file systems, and cloud storage.

**Relational Database:** The relational system that emerged in the 1980s is described as a cluster of relationships, each relationship having a unique single name. These relationships interconnect a number of tables. Each table contains a set of rows (records) and columns (attributes). The set of columns in each table is fixed, and each column has a specific pattern that is allowed to be used. In each row, the record represents a relationship that linked a set of values together. Relational database is functional in data storage, but it also has some limitations that make it less efficient to deal with big data. For example, relational database cannot tackle unstructured data. For datasets with network or graph patterns, it is difficult to use relational database to find the shortest route between two data points.

**NoSQL Database:** “Not only SQL” (NoSQL) database is considered the most important big data storage technology in database management systems. It is a method that depends on disposal of restrictions. NoSQL databases aim to eliminate complex relationships and provide many ways to preserve and work on data for specific use cases, such as storing full-text documents. In NoSQL database, it is not necessary for data elements to have the same structure, because it is able to deal with structured, unstructured, and semi-structured data [15].

**Distributed File Systems (DFS):** DFS manages datasets that are stored in different servers. Moreover, DFS accesses the datasets and processes them as if they are stored in one server or device. The Hadoop Distributed File System (HDFS) is the most popular method in the field. HDFS separates the data into multiple servers. Thus, it supports big data storage and high-efficiency parallel processing [11].

**Cloud Storage:** Cloud storage can be defined as servers that contain large storage space where users can manage their files. In general, this service is provided by companies known in the field of cloud storage. Cloud storage led to the term cloud computing, which means using applications over a virtual interface by connecting to the Internet. For example, Microsoft installs the Microsoft Office on its cloud servers. If a user has an account in the Microsoft cloud storage service and an available Internet connection through a computer or smart phone, the user will be allowed to use the cloud system by

logging into the account from anywhere. Besides cloud computing, cloud storage also has many other features, such as file synchronization, file sharing, and collaborative file editing.

### *1.3.3 Impacts of Big Data Storage*

Based on a McKinsey Global Institute study, the information that has been captured through organizations about their customers, operations, and suppliers by digital systems has been estimated as trillions of bytes. That means data volume grows at a great rate, so it needs advanced tools and technologies for storing and processing. Data storage has played a major role in the big data revolution [11].

Many companies are using emotion and behavior analysis from their data or social media to identify their audiences and costumers to predict the marketing and sales results. Smart decisions reduce costs and improve productivity. Data is the basis for informed big business decision-making. Analyzing the data offers more information options to make the right choice.

There are many techniques for managing the big data, but Hadoop is currently the best technology for this purpose. Hadoop offers the data scientists and data analysts flexibility to deal with data and extract information from it whether the data is structured or unstructured, and it offers many other convenient services. Hadoop is designed to follow up on any system failures. It constantly monitors the stored data on the server. As a result, Hadoop provides reliable, fault-tolerant, and scalable servers to store and analyze data at a low cost.

The development of cloud storage with the widespread use of Internet services, as well as the development of mobile devices such as smart phones and tablets, has enhanced the spread of cloud storage services. Many people carry their laptops when they are not in their offices, and they can easily access their files through their own cloud storage over the Internet. They can use cloud storage services like Google Docs, Dropbox, and many more to access their files wherever they are and whenever they want.

Companies are increasingly using cloud storage for several reasons, most notably because cloud services are becoming cheaper, faster, and easier to maintain and retrieve data. In fact, cloud storage is the better option for a lot of companies to address challenges caused by the lack of office space, the inability to host servers, and the expensive cost of using servers in the company, in terms of maintenance and cost of purchase. By using cloud storage, companies can save the servers' space and cost for other things. Google, Amazon, and Microsoft are the most popular companies in cloud storage services, just to name a few.

#### **1.4 Significance and Contribution**

This is the big data era, where the data is important for many sectors to get benefit from it by extracting knowledge and information. This information is used to discover clusters, profiles, factors, relationships, predictions, outliers and patterns, which help for decision making. Detecting outliers is an important step in data mining and machine learning because if outliers are not detected, the extracting information will be inaccurate, and many challenges may occur such as wrong decision or prediction. An important type of big data is data stream. Local outlier factor (LOF) is an important algorithm for local outlier detection. In fact, the main challenge of traditional local outlier detection algorithms is their application in data streams, such as intrusion detection system. Thus, the major challenge of LOF is that it requires the entire dataset and the distance values to be stored in memory. Another issue with LOF is that it needs to be recalculated from the beginning if any change occurs in the dataset.

**This research proposes a novel local outlier detection algorithm for data streams and proves its efficiency in several experiments.** The algorithm is called Genetic-based Incremental Local Outlier Factor (GILOF), which addresses the limitation of several existing LOF algorithms. Furthermore, GILOF algorithm has been enhanced by proposing new calculation method for local outlier factor, which called Local Outlier Factor by Reachability distance (LOFR). LOFR has a positive impact on the accuracy of outlier detection in real-world datasets that have sequence of outliers and high-density region (i.e. data points are very close to each other). Specifically, this is true for the KDD Cup99 HTTP

dataset, which includes a simulation of normal data, with abnormal data as attack traffic, on an IP scale in computer networks for testing intrusion detection systems. The new GA-based process and new LOFR method are the major contribution of this Dissertation. Both algorithms have two versions: GILOF, GILOF\_NS; and GILOFR, GILOFR\_NS, where GILOF\_NS and GILOFR\_NS do not contain skipping scheme.

## 1.5 Research Questions

This Dissertation contained six research questions. These questions trying to improve the efficiency of Local Outlier Factor in data stream. The questions are:

- 1) How can the GILOF algorithm solve the issue of memory consumption?
- 2) How does the GILOF algorithm detect the LOF in a data stream?
- 3) How does the genetic algorithm retain the shape of the density of data points?
- 4) Can a skipping scheme make a difference in the accuracy of local outlier detection and why?
- 5) Do GILOF and GILOF\_NS perform better than DILOF and DILOF\_NS when considering the accuracy of outlier detection?
- 6) Do GILOF and GILOF\_NS perform better than DILOF and DILOF\_NS when considering execution time?

## 1.6 Structure of the Dissertation

This dissertation will greatly benefit researchers in the field of local outlier detection because it contains two novel methods for detecting local outlier in data stream. Also, a comprehensive review of local outlier algorithms in static and stream environments is provided. The dissertation has five remaining chapters: the second chapter is Background and Statement of Research Challenge ; the third chapter is The State-Of-The-Art Algorithms in Local Outlier Detection; the fourth chapter is Genetic-Based Incremental Local Outlier Factor (GILOF) Algorithm; the fifth chapter is Improving the

Algorithm of Genetic-based Incremental Local Outlier Factor for Network Intrusion Detection; and the sixth chapter is the Synthesis and Conclusions.

## Chapter 2: Background and Statement of Research Challenge

"A Genetic-Based Incremental Local Outlier Factor Algorithm for Efficient Data Stream Processing." *the 4th International Conference on Compute and Data Analysis*, 2020 pp. 38-49, ACM.

"An Efficient Local Outlier Factor for Data Stream Processing: A Case Study" Forthcoming in *the 7th International Conference on Computational Science and Computational Intelligence 2020*, IEEE.

### 2.1 Background of Local Outlier Detection in Big Data Streams

#### 2.1.1 Data Stream Processing

Data stream processing is a key topic in the field of information flow processing (IFP) [126]. The aim of processing the data stream is to understand the data behavior and extract the information to make a better decision. During a workflow, data stream needs to be analyzed and then stored. The data stream is generated in many places, such as a stock market, sensors, log activity, social media, and so on. Therefore, processing the data stream is useful and important. It can be conducted through two general categories of methods [127]:

- Processing the data stream as queries.
- Processing the data stream as data stream mining.

A lot of studies show impressive results in both categories, which also prove the significance of the data stream processing in the big data era. For processing the data stream as queries, there are several systems that can manage the data stream sources. One of these systems is the data stream management system (DSMS). It processes the data as a structure of query which handles both continuous and updated data compared to the traditional database management system (DBMS) in a static environment [128, 7]. STREAM is another platform defined by Stanford University. It processes the data stream to get an approximate answer based on the need of the query optimization and data rate performance [129]. The other category, data stream mining, is about adapting data mining methods to process the data stream. There are various methods for data stream mining, such as classification, clustering, frequency item mining, and outlier detection [127].



### 2.1.2 *Outlier Detection Approaches*

Outlier detection is receiving increasing attention in research areas for discovering hidden information, data behavior, or long sequence data in an unexpected result. The main idea of outlier detection is to find a suspicious data point that is unusual compared to other data points. Several approaches have been introducing models such as distance-based outlier detection, density-based outlier detection, clustering based outlier detection, statistical based outliers detection, classification based outliers detection, and frequent pattern mining based outliers detection [2,3,4].

Distance based outlier detection depends on the object's distance from its neighbors. In [2,3], the authors defined the anomaly detection by using the object's distance between  $k$  and  $R$  as a parameter. In [130], the authors described the benefit of using the  $k$ -mean distance as a measurement between object neighbors.

Density based outlier detection depends on the comparison between the density of points and their local neighbors. Therefore, the outlier score is based on the local outlier factor (LOF). It gives a scale of the data point density and their neighbors [4]. In [91], the authors used sliding windows to distinguish between outlier and new data behavior. Also, in [88], the authors proposed an incremental local outlier factor (ILOF), which calculates the LOF value in the data stream. Despite the value, it cannot distinguish between the data behavior and an outlier.

Clustering based outlier detection regards a set of similar data points as a cluster. Thus, the outliers can be either far from the center of the cluster or not belong to any cluster. In [131], the authors used the weight of other attributes according to the mining task. In [132], the authors defined the micro-cluster, which is the differential between the outlier and the normal data.

Statistical based outlier detection is based on the data distribution to identify the outlier data point and normal data point. The outlier usually has a lower probability generated from the data distribution. Statistical based outlier detection has two categories for detecting an outlier, which are the parametric method and the non-parametric method. The data distribution in the parametric method is known. The non-parametric method cannot assume the data distribution, so it is suitable to be used for

a part of the data stream [4, 133].

Classification based outlier detection first trains the dataset through the classification method, then classifies the model into normal data or an outlier class. Moreover, in one class of unsupervised approach, the boundary learns about the normal data, and the data point that is outside the boundary is an outlier [134]. For example, the authors in [135] determined the decision boundary by using a *kNN* method. In [136], the authors defined two decision boundaries by using Support Vector Machine (SVM), polynomial kernel and *kNN* method [136,137].

Frequent pattern mining based outlier detection is another unsupervised method used to find a frequent pattern under normal data behavior. The outlier is a pattern that does not fit to the basis of the normal data behavior. Usually, it has a smaller number of frequent patterns. In [138], the authors applied the MEFOP which is a measurement used between the data point and the set of maximum frequent pattern.

### 2.1.3 Local Outlier Detection

Outliers are divided into two categories: global outliers and local outliers. The data points  $p_o$  and  $p_l$  are considered global outliers if they are very far from all data points [139]. In [140] and [141], the authors detected the global outliers by using a sliding window with respect to the current window. In [142] the authors detected outliers in the fast streaming data and sensors with limited resources by using a data editing technique. Their work needs normal data because they use a supervised approach. In [143] the time complexity and memory consumption were improved compared to [140] and [141]. When the outliers in any dataset are calculated based on all points, the outliers are considered global outliers. However, these methods fail to detect the outliers with good accuracy in data that has inhomogeneous density.

Local outliers are the probability of data points to be outliers compared to their local neighborhood, which is measured based on the *k*-Nearest Neighbors (*kNN*) algorithm. Local outlier detection is extensively discussed in [58]. The LOF algorithm obtains good performance in detecting

local outliers in inhomogeneous densities without previous knowledge about the dataset distribution, and it has become a widely used technique for local outlier detection.

The accuracy of LOF has been improved by several techniques that use improved approaches for calculating  $kNN$  [67]. These approaches extend the range of  $kNN$  to detect outliers and discover the relationship of symmetrical neighborhoods [70]. For example, in [68] the authors improved the LOF time complexity by using approximate computations. Other LOF studies concentrated on covering the data types that are more complex such as spatial datasets [140] or categorical data sets [145]. Normalizing the outliers and finding the probabilities of local outliers is another type of LOF [71]. The authors of [88] proposed the efficient ILOF algorithm, which is designed to calculate the local outliers in data streams. In [96], the MILOF algorithm was used to address limitations of ILOF by using  $k$ -means to summarize old data points with respect to the LOF score. The authors of [104] proposed DILOF as a memory limited algorithm. DILOF used the gradient descent method to summarize the old data, which leads to better results than MILOF.

#### 2.1.4 Incremental Local Outlier Factor (ILOF)

Although the LOF does not perform well in the stream environment, the Incremental Local Outlier Factor (ILOF) overcomes this weakness and is able to calculate the LOF score in a stream environment. The LOF algorithm is defined above and figure 2.1 shows the key definitions of the LOF. The core principle of the ILOF method is to discover outliers in the local density in the stream environment [88]. ILOF is a density-based outlier detection technique. The major task of the ILOF is to update and re-calculate the score of the LOF when a new data point  $np$  is inserted.

ILOF uses the same components of the LOF for measuring the outlierness score:  $k$ -distance,  $k$ -nearest neighbors, reachability distance ( $Rd$ ), and local reachability distance ( $Lrd$ ). There are two parts to the method of insertion for the ILOF: (1) the  $Rd$ , the  $Lrd$ , and the  $LOF$  scores are computed according to the  $np$ ; and (2) it updates the  $k$ -distance, the  $Rd$ , the  $Lrd$ , and the  $LOF$  score for the existing data points. A strength of the ILOF algorithm is that it can calculate LOF scores in streaming data. Despite

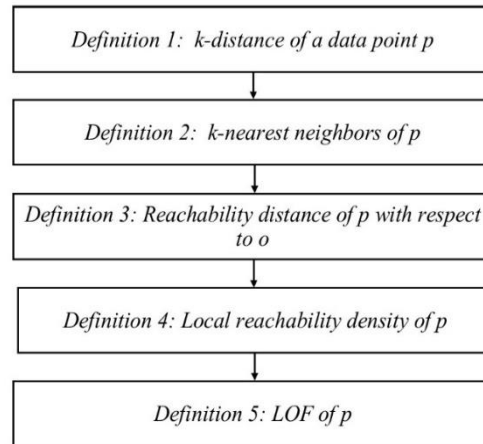


Figure 2.1. The Key definitions of LOF algorithm.

the strengths of the ILOF algorithm for data streams, its main issue is memory consumption because it stores all of the previous data points. This takes up a large amount of memory and significantly increases the analysis time.

#### 2.1.5 Memory Efficient Incremental Local Outlier Factor (MILOF)

The Memory Efficient Incremental Local Outlier Factor (MILOF) is an unsupervised outlier method for evaluating the local outliers in the data stream. The MILOF is able to reduce the time complexity and the memory requirements of ILOF, and it is suitable for a wide range of applications. The MILOF processes a data point in three phases: the summarization phase, the merging phase, and the revised insertion phase.

The summarization phase is initiated when the number of points reaches the memory limit. The first half of the data points are then summarized, whereas the most recent points are left intact to keep a higher resolution of the present streaming data points. The *k*-mean algorithm is used for clustering the first half of the data points to summarize them [96].

The merging phase takes place after the last half of data points is obtained from the summarization phase; a new cluster is created from the incoming data points. Afterward, the new cluster is combined with the existing cluster to produce a single set of clusters.

The revised insertion phase in the MILOF algorithm is determined based on both the recent data points and the cluster points, which have a similar concept of the ILOF algorithm as follows. First, it computes the LOF for the new incoming data point. Second, if necessary, it updates the  $k$ -distance,  $R_d$ ,  $lrd$  and LOF values for the established data points [4].

The advantage of the MILOF is that it is able to reduce the computational complexity and memory requirements. The major weakness of this method is that the  $k$ -means cannot maintain the density of data points.

#### 2.1.6 *Density Summarization Incremental Local Outlier Factor (DILOF)*

The DILOF algorithm was proposed to address the limitation of the ILOF when processing a data stream [104]. The DILOF contains two steps: the detection step and the summarization step. The detection step is used to detect the outliers and update the data points in the window when new data points are inserted. The ILOF algorithm is used in this step to detect the outliers while a skipping scheme is used to detect the sequence of outliers, such as during DoS attacks. The summarization step is used to summarize the older half of the data points in the window based on non-parametric density summarization (NDS). A strength of the DILOF is that it performs better than the MILOF for the accuracy of outlier detection and execution time [104]. However, the NDS algorithm depends on a gradient-descent method, which may get stuck in the local minima.

## **2.2 Definition of Research Challenge**

In the big data era, outlier detection is a very important step in many applications, such as network intrusion detection systems and decision support systems. The objective of outlier detection is to detect suspicious items and unusual activities. For example, in practice, analyzing the dataset to extract information without removing the outlier's data will lead to inaccurate information, which will result in wrong decisions. Recently, outlier detection has gained a lot of attention from researchers, especially regarding data streams. Nevertheless, there are challenges in applying traditional local outlier detection algorithms for data streams. One of the well-known algorithms of outlier detection is Local

Outlier Factor (LOF). The issue with LOF is that it needs to store the whole dataset with its distances' results in memory. In addition, it needs to start from the beginning and recalculate all processes if any change happens in the dataset, such as inserting a new data point.

The ILOF, MILOF and DILOF algorithms address the LOF issues in data streams, but they also have some issues that limit their performance. The major issue for the ILOF is that it needs to retain all data points in memory, which leads to a large usage of memory and long computational time. The MILOF algorithm used  $k$ -mean to summarize the old half data point in the window, but  $k$ -mean cannot preserve the density of data points. The DILOF algorithm summarize the old half of data points in the current window by using the gradient-descent method. An issue for gradient-descent is that it might be stuck in local minima.

As mentioned before in Chapter 1, this research aims to develop the GILOF as a novel local outlier detection algorithm for data streams. The primary purpose of this work is to measure LOF score under the following conditions: 1) A small part of the dataset is stored in memory, 2) algorithm has no previous knowledge of the data distribution, 3) outlier detection for the data point  $pt$  should be done at the current time  $T$  and 4) the algorithm has no previous knowledge about the future data points when detecting the current outliers. Those four conditions lead to the six detailed research questions listed in Section 1.5, which in turn guide the algorithm development and experiment in this research.

## 2.3 Datasets

For practical reasons, label information is not processed and analyzed in unsupervised outlier detection because it must be compared and evaluated. When a new outlier detection algorithm is developed, it is usual to apply the new algorithm to publicly available datasets and compare the results of the new algorithm with the common unsupervised outlier detection algorithms, such as the LOF. There are many classification datasets that are fully available in the UCI machine learning repository [146]. Additionally, some outlier detection datasets are provided in [147]. The datasets below are real-world datasets and they contain outlier data points. These benchmark datasets are used in this study, and

they were used to analyze the new GILOF and GILOFR algorithms and to compare them with existing algorithms. Table 2.1 summarizes the features of the following datasets.

### 2.3.1 UCI Vowel Dataset

A Vowels dataset is considered a multivariate time series dataset as well as a classification dataset, which classifies speakers. In one particular case, nine speakers spoke two Japanese vowels, respectively. One speech by a speaker shapes a time series from 7 to 29 lengths and each point in the time series consists of twelve characteristics. In outlier (anomaly) detection, any frame in the training dataset is treated as a single data point, although the UCI machine learning repository deems a block of frames (talk) as a single point. Furthermore, classes six, seven, and eight are considered as inliers. The dataset contains 12 dimensions with 1,456 data points and 3.4% of these data points are outliers [146, 147].

Table 2.1. The Real-World Datasets Features

<b>Datasets</b>	<b>Number of data points</b>	<b>Dimension</b>	<b>Class</b>
<b>UCI Vowel dataset</b>	1,456	12	11
<b>UCI Pendigit dataset</b>	3,498	16	10
<b>KDD Cup 99 Smtip dataset</b>	95,156	3	unknown
<b>KDD Cup 99 Http dataset</b>	567,479	3	unknown

### 2.3.2 UCI Pendigit Dataset

The pendigits dataset is originally from the UCI machine learning repository [146]. This dataset is a multiclass classification that has 16 dimensions with 10 classes. The Pendigits dataset consists of 25 samples, which were generated by 44 writers. Thirty of the writer's samples are used for training,

while the other 14 writers' samples are used for testing. The original training set contains 7,494 data points, and the testing set has 3,498 data points [147].

### 2.3.3 *KDD Cup 99 SMTP Dataset*

In this instance, the SMTP service is used from the KDD CUP 1999 dataset, which is from the UCI machine learning repository [146]. The original dataset (KDDCUP99) contains 4,898,431 data points, where 3,925,651 data points are considered as attack (80.1%) data points. A smaller set is forged by 976,157 data points that include 3,377 (0.35%) attacks. The SMTP service data is used to create the SMTP KDDCUP99 dataset from this smaller dataset. The SMTP KDDCUP99 dataset contains 3 dimensions and 95,156 data points that include (0.03%) outliers [147].

### 2.3.4 *KDD Cup 99 HTTP Dataset*

KDD CUP 1999 is the original dataset from the UCI machine learning repository [146]. This dataset contains 41 attributes, but it is reduced to 4 (service, dst\_bytes, src\_bytes, duration), where service is only categorical. By using the service, the data is split into HTTP, FTP, FTP\_data, SMTP. The original dataset KDDCUP99 contains 4,898,431 data points, where 3,925,651 data points are considered as attack (80.1%) data points. A smaller set is created by 976,157 data points that include 3,377 (0.35%) attacks. The HTTP service data is used to create the HTTP KDDCUP99 dataset from that smaller dataset, which simulates normal data with attack traffic on an IP. The HTTP KDDCUP99 dataset contains 3 dimensions and 567,497 data points that include (0.4%) outliers [147].

## **2.4 Genetic Algorithm**

A Genetic Algorithm (GA) is a heuristic search algorithm that is based on Charles Darwin's theory of natural evolution. It is based on the natural selection process, which selects the fittest chromosomes (individuals) for reproduction, to produce offspring for the next generation. A GA is a



form of evolutionary computation. GAs can often solve complex problems under time consuming or other difficult requirements. The search for optimal solutions is based on fitness and genetic operations such as selection, crossover, and mutation [148]. The authors in [149] described GAs as a powerful method for finding outliers compared to other data mining techniques in a static environment. The authors in [150] used classification method to preprocess the data stream and improved the performance of adaptive learning by using GA. In [151], the authors used the concept drift with GA to mine classification rules by taking a small portion of the data based on the fitness computation [152]. In [153], the authors proposed a solution to identify the number of clusters and separated the overlapping clusters by using GA and the Gaussian mixture model (GMM). In short, GA-based data stream processing has generated significant results because it is easier to adapt for many different environments.

#### *2.4.1 Steps in Genetic Algorithm*

Genetic Algorithm (GA) is a search algorithm from the field of evolutionary computation based on biological evolution. The purpose of the GA is to search for an optimum, which will improve the performance of a system. There are several GA operations which have analogs in biology [154,155]:

1) Objective (Fitness) function: This function is used to calculate the fitness of each chromosome in the population. Usually, each chromosome's fitness value represents the quality of the potential solution defined by the chromosome. Therefore, a fitness function is essential to know which are the fittest chromosomes [156].

2) Chromosome (Individual): It is an array of binary or numerical values that represents the genes being evolved and it defines a candidate solution. Typically, each possible candidate solution is encoded as an array of parameters or as a bit string based on the problem given for the genetic algorithm [156,157]. For example, a chromosome represented as [0,1,1,0,0,0,1] means the 2nd, 3rd, and 7th candidates are to be selected for the fitness function.

3) Population: This is a set of chromosomes. A set of chromosomes is set randomly as an initial population. Then, each chromosome is evaluated based on the fitness function [156]. Thus, the population carries multiple possible solutions encoded in the chromosomes. This allows a GA to explore many areas within a search space simultaneously.

4) Selection: It is an essential operation for the genetic algorithm to produce the offspring for the next generation. It selects two parents from the population based on their fitness to undergo crossover [157,158]. The purpose of selection is to find chromosomes that have above average fitness values. Very strong selection reduces the diversity of the population and may reduce additional change and progress. Very weak selection can lead to slower evolution [157].

Some of the most popular selection methods are roulette wheel selection (RWS), stochastic universal sampling selection (SUS), linear rank-based selection (RNK), linear rank-based selection with selective pressure (RSP), the Elitism, and tournament selection (TNT) [157,159,160].

RWS is based on the number of individuals within a population and their relative fitness values. Each chromosome has a segment of a virtual roulette wheel where the fitness of the chromosome determines the size of the segment [157]. Selecting the parent's chromosome is based on the position of the selecting point in the wheel. The process of RWS is repeated until all the chromosomes are examined. In the end, the chromosomes with higher fitness values are more likely to be selected.

SUS is derived from fitness proportionate selection. It samples all solutions by selecting them in equally spaced intervals. This gives a chance for a weak chromosome to be chosen [161].

RNK was developed by Baker [162]. In ranking selection, each chromosome is ranked based on its fitness. Therefore, it reduces the disadvantages inherent in using the absolute fitness, which may converge too quickly when one chromosome has far better fitness than the next nearest chromosome [157].

In RSP, the selection pressure uses a mechanism to scale the selection pressure. It measures the range of fitness from maximum to average in the population and uses that to scale the selection pressure [160]. This manages the slope of selection pressure in the linear ranking method.

Elitism aims to find good chromosome fitness values in the population for the next generation by making sure that at least one copy of the best individual (or individuals) is preserved [157]. Thus, the next generation's population will have at least one copy of the best chromosome based on the number of Elitism defined by the user.

TNT selects a subset of chromosomes randomly from the population. Then each chromosome in the subset competes against the others [159]. The chromosome with the highest fitness from the subset of chromosomes is selected for crossover.

5) Crossover: The name of crossover comes from crossing the 'genes' of two parents. It is conducted after the selection step to produce one or more new chromosomes, which are known as the children, which combine the characteristics of the parents. Crossover works by combining the first chromosome and the second chromosome based on a cut-off point defined in the crossover methods. The most common methods for the crossover are one-point crossover, two-point crossover, and uniform crossover [163,164].

The one-point crossover involves two stages [164]. First, a point on both parent chromosomes is selected randomly as the crossover point. Second, the two chromosomes swap bits to the right of the point to generate two new chromosomes.

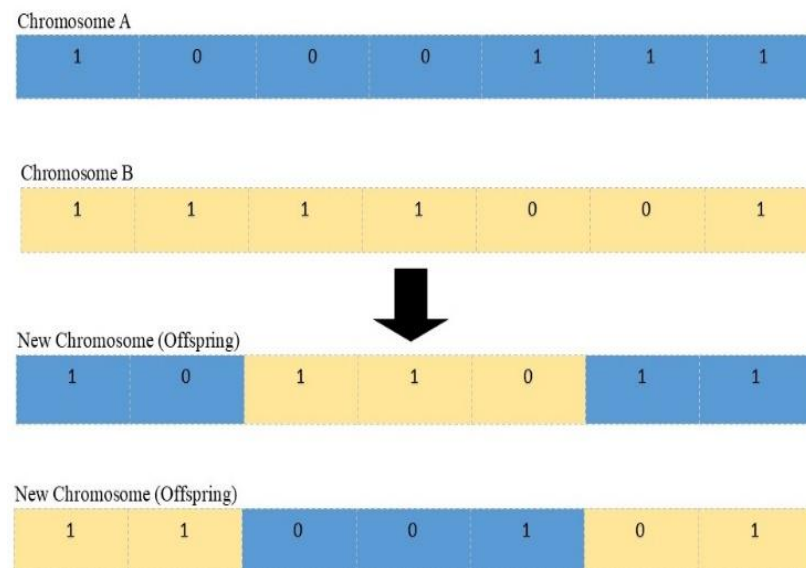


Figure 2.2. Two-point crossover

Two-point crossover picks two crossover points instead of one. The two parent chromosomes swap the bits between the two points to generate the new chromosomes (Figure 2.2).

In uniform crossover, usually each bit from either parent chromosome has equal possibility to be used in the two offspring. Sometimes a mixing ratio may be applied. Then, more bits from one parent chromosome are inherited in the offspring [164].

6) Mutation: is another GA operation used to preserve the diversity in the population and prevent the population from being stuck in a local minimum [165]. It is based on changing one or a few random values in a chromosome. The probability of changing a value is based on the mutation rate [158]. Usually, mutation creates one offspring compared to the two offspring created by crossover. There are several mutation methods, such as single point mutation (SPM), uniform mutation (UNM), and boundary mutation (BDM). SPM flips a value from 0 to 1 and vice versa based on the mutation rate [158]. In contrast, UNM is applied to non-binary chromosomes, it replaces the value of the selected gene with a uniform random value taken between the lower and upper bounds determined by a user-defined range [166,167]. BDM replaces the value of the selected bit with either the lower or upper bound, which is determined through a variable assigned by the user.

## **2.5 Conclusion**

This chapter addressed the main issues of the LOF, also it shows the issues of the extension algorithms of the LOF in data stream. In addition, the related works such as data stream, outlier detection approaches, and genetic algorithm were highlighted. To test a new unsupervised local outlier detection algorithm, it usually applied to available real-world datasets. Then, the results are compared with the common local outlier detection algorithm. Therefore, the datasets that were used in this dissertation are covered in detail.

## Chapter 3: The State-Of-The-Art Algorithms in Local Outlier Detection

"A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams." *Big Data and Cognitive Computing*. 2020, 5, 1. MDPI. <https://doi.org/10.3390/bdcc5010001>

### 3.1 Introduction

Outlier detection is the term used for anomaly detection, fraud detection, and novelty detection. The purpose of outlier detection is to detect rare events or unusual activities that differ from the majority of data points in a dataset [16]. Recently, outlier detection has become an important challenge in many applications such as in health care, fraud transaction detection for credit cards, and intrusion detection in computer networks. Therefore, many algorithms have been proposed and developed to detect outliers. However, most of these algorithms are designed for a static data environment and are difficult to apply to streams of data, which represent a significant number of important application areas.

Outlier detection can be viewed as a specific application of generalized data mining. The process of data mining involves two steps: data processing and data mining. The aim of data processing is to create data in good form, i.e., the outliers are removed. Data mining is the next step, which involves the discovery and understanding of the behavior of the data and extracting this information by machine learning [17]. Outlier detection (or data cleaning) is an important step in data processing because, if an outlier data point is used during data mining, it is likely to lead to inaccurate outputs [18]. For example, a traffic pattern of outliers in the network might mean the data has been sent out by a hacked computer to an unauthorized device [19]; if the outliers cannot be detected and removed, a machine learning technique applied to the data is likely to be misled. Additionally, in many cases, the goal of data mining is to find the outliers. For example, an outlier image in magnetic resonance imaging could mean the presence of malignant tumors [20]. An outlier reading from an aircraft sensor may indicate a fault in some components of the aircraft [21].

Outlier detection (also known as anomaly detection) is split into two types, global and local detection. For a global outlier, outlier detection considers all data points, and the data point  $pt$  is

considered an outlier if it is far away from all other data points [22]. The local outlier detection covers a small subset of data points at a time (Figure 1.1). A local outlier is based on the probability of data point  $pt$  being an outlier as compared to its local neighborhood, which is measured by the  $k$ -Nearest Neighbors ( $k$ NN) algorithm [4].

Recently, many studies have been published in the area of outlier detection, and there is a need for these studies to be reviewed to understand the depth of the field and the most promising areas for additional research. This review is focused on local outlier detection algorithms in data stream environments. This review is distinguished from previous reviews as it specifically covers the Local Outlier Factor (LOF) algorithms in data stream environments and reviews the most recent, state-of-the-art techniques. In this chapter, different methods and algorithms for local outlier detection are summarized. The limitations and challenges of the LOF and other local outlier detection algorithms in a stream environment are analyzed. Moreover, new methods for the detection of an LOF score in a data stream are proposed. This literature review will greatly benefit re-searchers in the field of local outlier detection because it provides a comprehensive explanation of the features of each algorithm and shows their strengths and weaknesses, particularly in a data stream environment. The chapter has seven remaining parts: the second section is Outlier Detection Background; the third section is Literature Review Methodology; the fourth section is Literature Review Results; the fifth section is Applications of Outlier Detection; sixth section is Computational Complexity; seventh section is Strengths and Weaknesses; and the eighth section is the Conclusion.

### **3.2 Outlier Detection Background**

Many methods and algorithms have been developed to detect outliers in datasets. Most of these methods and algorithms have been developed for specific applications and domains. Outlier detection has been the focus of many review and survey papers, as well as several books. Patcha et al. and Snyder [23,24] published a survey of outlier detection techniques for intrusion detection. Markou and Singh [25,26] provided a review of outlier detection methods that use statistical approaches and neural

networks. A comprehensive survey of outlier detection algorithms developed in statistical domains and by machine learning was provided by Hodge et al. [27]. Goldstein et al. [28] provided a comparative evaluation of several unsupervised outlier detection algorithms. Wang et al. [29] provided the progress of outlier detection algorithms up until 2019 and illustrated the different methods of outlier detection. Tellis et al. [2] presented a survey that aimed to describe the various techniques for outlier detection in data streams. The objective of their paper was to identify the various techniques that can be used in the detection of an anomaly in applications involving data streams, such as criminal behaviors, fraud detection for credit cards, finding computer intrusions, and calling cards. Park et al. [30] presented a comprehensive review of concept drift detection, outlier detection, and anomaly pattern detection for data streams. The main contribution of their paper was that the approach used varied, based on the size of the streams of data. Other review and survey papers of outlier detection for different applications and domains are provided in [31-36].

### *3.2.1 Statistical Outlier Detection*

Outlier detection is a statistical process that aims to discover the data points that are inconsistent with the dataset [37]. In general, outliers and inliers are determined based on the model of data distribution. Statistical outlier detection techniques can be categorized as supervised, semi-supervised, and unsupervised modes. The scientific research on statistical outlier detection provides two approaches to handle outliers in a dataset. First, the outliers must be identified for further study. Second, the data model should be designed to accurately handle the outlier data points [38]. Statistical-based methods for outlier detection are further categorized into two methods: parametric and non-parametric [39].

#### *3.2.1.1 Parametric Method*

Parametric-based methods assume an underlying distribution model (e.g., Gaussian distribution) for describing normal data and fit the assumed model to the data by estimating the model parameters. Such models are based on the Gaussian model or non-Gaussian data models.

One of the most common statistical methods for outlier detection is the Gaussian model. In the training process, the Gaussian model uses Maximum Likelihood Estimates (MLE) to estimate the variance and mean of the Gaussian distribution of the data [40]. Discordancy tests are then implemented to detect outliers. The unsupervised method for outlier detection with a globally optimal exemplar-based Gaussian Mixture Model (GMM) was introduced in [41]. For a more accurate outlier detection approach, the GMM with Locality Preserving Projections (LPP) was developed in [42] and involves the use of GMM combined with subspace learning.

Another statistical method of outlier detection is to use a regression model. When implementing this technique, the first step is to build a regression model that is suitable for the data. For the test step, the regression model is tested by evaluating each data point as it relates to the model. A data instance is labeled an anomaly when a deviation occurs between the real value and the expected value generated by the regression model [43]. To detect an outlier in linear regression, Satman [44] proposed an algorithm based on a non-interacting covariance matrix and concentricity steps implemented using the least square estimation. Another method of regression-based outlier detection was proposed by Park et al. [45]. This method was created to detect outliers based on sensor measurement. In this method, the weighted summation approach is used to construct a synthesized independent variable from the observed values [29].

#### 3.2.1.2 Non-Parametric Method

A non-parametric method must be used when the data is not normally distributed and there is no prior knowledge about the data distribution. It is also called the distribution-free method because the data does not have a specific (known) distribution. To identify whether a data point is an inlier or outlier, some criteria must be implemented for the dataset. The most common approaches are histograms and kernel density.

The main concept in histogram-based approaches is that for every dataset characteristic a histogram is created. It is similar to the Naïve Bayes algorithm, where all independent characteristic



probabilities are multiplied. Typically, it measures the distance between a histogram-based model of normality and a new test data point to determine if it is an inlier or outlier [32].

Kernel Density Estimation (KDE) is another popular non-parametric statistical approach for outlier detection [46]. Latecki et al. [47] proposed an unsupervised approach for detecting outliers using kernel features. Their process of outlier detection is executed by comparing every local density data point to the local density neighbor. A better method was proposed by Gao et al. [48]; their method improved the performance and showed better scalability in wide datasets using kernel-based methods. KDE was successfully used by [49] to estimate the distribution of sensor data points to detect malignant nodes [29].

### **3.3 Literature Review Methodology**

The aim of this literature review is to report on current works using the LOF with a focus on local outlier detection in data streams. It also proposes a new methodology for a more efficient LOF in data streams. This chapter focuses on the research that was conducted from May 2000 to April 2020. The search took place through research papers in electronic databases published in English. These databases were Google Scholar, IEEE Xplore, ACM, Springer, Science Direct, and MDPI. Keywords including outlier detection, local outlier detection, local outlier detection in data streams, Local Outlier Factor in data streams, and data stream mining were used. In the initial search, a total of 528 papers were reviewed by title and abstract. The selected papers were then categorized into two sections: the static environment and the stream environment. After that, the selected papers were filtered by the full text. The total number of selected papers was 47 (Figure 3.1). In the static environment subsection, the local outlier detection papers with the most citations were selected to be reviewed in greater detail, while the remaining papers were reviewed more briefly. In the stream environment subsection, the recent state-of-the-art papers on the LOF in data streams were reviewed in detail, while the remaining papers were reviewed briefly.

The criteria for inclusion in this literature review were the following: (1) the article contained a new density—based local outlier detection algorithm in a static environment, (2) the local outlier detection algorithm was an unsupervised algorithm, (3) the article contained a new LOF algorithm for a stream environment, or (4) it included a local outlier detection algorithm in a stream environment. The research questions that have been answered in this literature review are the following: (1) what are the LOF algorithm issues and challenges in stream environments, (2) what are the existing methods and techniques that have been applied to the LOF in a stream environment, (3) what are the existing local outlier detection algorithms that need to be developed in order to work in a stream environment, and (4) how does the new methodology execute the LOF efficiently in a stream environment?

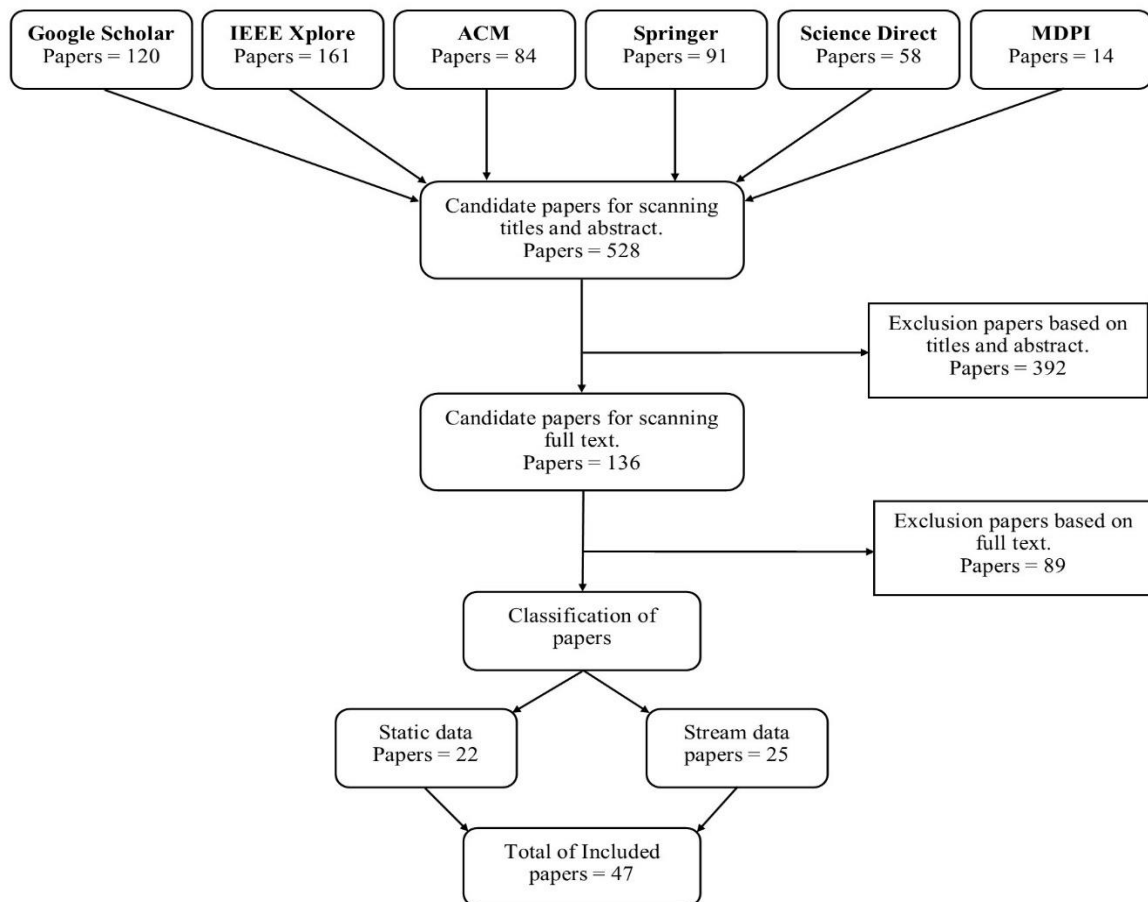


Figure 3.1. The search strategy flowchart for selecting articles.

### 3.4 Literature Review Results

Outlier detection in datasets has been studied since the nineteenth century [50]. A huge amount of research on detecting outliers has been conducted [16, 51–57]. As mentioned above, there are two forms of outlier detection: global and local. The LOF is a well-known algorithm for detecting local outliers [58]. This work reviews local outlier detection algorithms for numerical data for both static and stream environments. Generally, outlier detection algorithms work in the following three modes:

**Supervised Outlier Detection:** Supervised algorithms assume the data includes fully labeled training and test datasets. A substantial approach in this situation is to develop a predictive model for non-normal and normal classes of data. Every hidden data point is compared with the model to decide in which class the data point belongs. The main issue of supervised outlier detection is that the outlier data points are much fewer than normal data points in training datasets. This issue appears to be due to the imbalance of class distribution. However, this issue has already been addressed in machine-learning and data-mining literature [59–62]. For instance, decision trees, such as C4.5 [63], cannot perform well with imbalanced data, while Artificial Neural Networks [64] perform better.

**Semi-supervised Outlier Detection:** Semi-supervised algorithms assume the data includes labeled training data for the normal class only. Since labels are not required for the outlier class in semi-supervised algorithms, they have wider applications than supervised algorithms. The main idea of semi-supervised algorithms is that a normal class model is learned, and then, the outliers are detected by deviating from the model. This is also called a one-class classification [65]. The one-class Support Vector Machine [66] is a popular algorithm used in the semi-supervised technique.

**Unsupervised Outlier Detection:** Unsupervised algorithms do not require labels in the data and there is no discrimination between training and testing datasets. Therefore, it is a more flexible mode than the others. The major idea of unsupervised outlier detection algorithms is to score the data points based on the essential characteristics of the dataset only. Generally, density or distance is used to provide an assessment about whether a data point is an inlier (normal) or outlier. This review is focused on unsupervised outlier detection.

### 3.4.1 Static Environment

Outlier detection in static environments has received a lot of attention in the areas of data mining and machine learning. Many algorithms have been designed for the unsupervised outlier detection of global and local outliers. The LOF is the foundation for local outlier detection. Many researchers have, however, addressed local outlier detection from different perspectives. This section reviews the local outlier detection algorithms in general and goes deeper into the most popular algorithms. Table 3.1 Summarizes the most-cited local outlier algorithms in a static environment. Figure 3.2 shows all of the reviewed papers included in this section.

#### 3.4.1.1 Local Outlier Factor (LOF)

The LOF algorithm is defined in [58] by using density-based methods. For each data point, the process of finding the LOF includes calculating the degree of outlying. The idea of a local outlier is introduced by the LOF. The key definitions for the LOF [58] are:

**Definition 1.** *k-distance of a data point  $p$ .*

The distance between the two data points  $p$  and  $o$  can be calculated by using a Euclidean  $n$ -dimensional space (Equation (1)).

$$d(p, o) = \sqrt{\sum_{i=1}^n (p_i - o_i)^2} \quad (1)$$

Let the dataset be a  $D$  and a positive integer is  $k$ . For a data point  $p$ , the *k-distance*  $p$  is the distance  $d(p, o)$  between  $p$  and the farthest neighbor data point (record)  $o$  ( $o \in D$ ) in the following conditions:

- (1) At the least,  $k$  data points (records)  $o' \in D \setminus \{p\}$  maintains that  $d(p, o') \leq d(p, o)$ ;
- (2) At the most,  $k-1$  data points (records)  $o' \in D \setminus \{p\}$  maintains that  $d(p, o') < d(p, o)$ .

Table 3.1. Summary of the most-cited local outlier algorithms in a static environment.

Authors and Year	Algorithm	Features	Time Complexity	Taxonomy	Remarks
Breunig et al., 2000 [58]	LOF	Better in spherical data	$O(n^2)$	Nearest neighbor based	Introduced local outlier detection, and it uses Euclidian distance and $k$ NN to estimate local density.
Tang et al., 2002 [67]	COF	Better in linear data	$O(n^2)$	Nearest neighbor based	Overcomes the linear distribution and it uses the chaining distance to find the local outlier.
Jin et al., 2006 [68]	INFLO	Reversing the data point's nearest neighbors	$O(n^2)$	Nearest neighbor based	Overcomes the issue of the data points within the boundaries of the clusters. Effective for data points that include clusters with diverse densities.
Kriegel et al., 2009 [69]	LoOP	Presumes "half-Gaussian" distribution of distances	$O(n^2)$	Nearest neighbor based	Estimates the local density by probabilistic set distance. Combines probability and statically approaches to provide outlier score.
Papadimitriou et al., 2003 [70]	LOCI	Presumes "half-Gaussian" distribution of quantity of data points density in the neighbors	$O(n^3)$	Nearest neighbor based	Uses the same process as LoOP; the difference is the amount of instance is used rather than the distance. Long computation time but does not need parameters.
Papadimitriou et al., 2003 [70]	aLOCI	Uses the quad trees to speed up counts	$O(NLdg + NL(dg + 2d))$	Nearest neighbor based	Simple approximation of density based on occupancy and depth. Overcomes the high time complexity in LOCI.
He et al., 2003 [71]	CBLOF	Uses a heuristic procedure for small and large clusters	$O(n^2)$	Clustering-based	Many parameters. Ineffective in detecting the local outliers. It takes into consideration the local variation of clusters.
Amer et al., 2012 [72]	LDCOF	Estimates the clusters' densities. Spherical distribution is presumed for the cluster members	$O(n^2)$	Clustering-based	Many parameters. Effective in detecting the local outliers. The threshold is included to determine whether or not the points are outliers.

References			[74]												[87]
			[73]												[86]
			[69]											[83]	[85]
	[58]	[67]	[68]	[75]	[76]	[70]	[77]	[71]	[72]	[78]	[79]	[80]	[81]	[82]	[84]
Year	2000	2002	2003	2004	2005	2006	2008	2009	2012	2013	2014	2016	2017	2018	2019

Figure 3.2. Summary of references used in the literature review for static data.

**Definition 2.** *k*-nearest neighbors of *p*.

Here, the meaning of *k*-Nearest Neighbors (*k*NN) of *p* is any data point *q* whose distance to the *p* data point is not greater than the *k*-distance(*p*). Those *k*-Nearest Neighbors of *q* form the so-called *k*-distance neighborhood of *p*, as described in Equation (2).

$$N_{k\text{-distance}(p)}(p) = \{ q \in D \setminus \{p\} \mid d(p, q) \leq k\text{-distance}(p) \} \quad (2)$$

**Definition 3.** *Reachability distance of p with respect to o*.

Let *k* be a positive integer. The reachability distance of a data point *p* with regard to the data point *o* is defined in Equation (3).

$$\text{reach-dist}_k(p, o) = \max \{ k\text{-distance}(o), d(p, o) \} \quad (3)$$

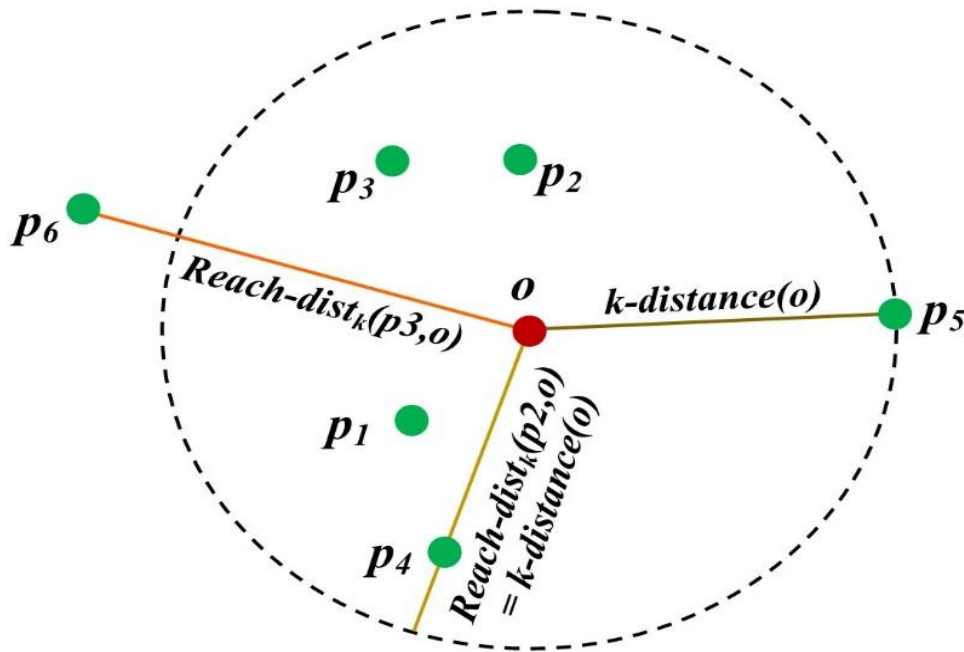


Figure 3.3. The reachability distance for different data points *p* with regard to *o*, when *k* equals 5.

Figure 3.3 shows an example of the reachability distance when the  $k$  value is 5. If the actual distance between a data point  $p4$  and data point  $o$  is shorter than the  $k$ -distance ( $o$ ), then the reachability distance of data point  $p4$  will be the  $k$ -distance ( $o$ ). On the other hand, if the actual distance between a data point  $p6$  and data point  $o$  is greater than the  $k$ -distance ( $o$ ), then the reachability distance of  $p6$  is the actual distance. Such a smoothing operation is used to reduce the statistical fluctuations of  $d(p,o)$  for each data point close to  $o$ . The smoothing strength is controlled by  $k$ .

**Definition 4.** *Local reachability density of  $p$ .*

In density-based clustering algorithms, two parameters are used for defining the notion of density: (1) *MinPts* for a minimum number of data points and (2) a volume. The authors of [53] used  $reach-dist_{MinPts}(p,o)$  for  $o \in N_{MinPts}(p)$  as a volume measurement. Thus, the local reachability density (*Lrd*) of data point  $p$  is defined in Equation (4).

$$Lrd_{MinPts}(p) = 1 / \left( \frac{\sum_{o \in N_{MinPts}(p)} reach-dist_{MinPts}(p,o)}{|MinPts(p)|} \right) \quad (4)$$

In Equation (4), the average reachability distance based on the *MinPts* number of nearest neighbors of data point  $p$  is first calculated. Its inversion then generates the local reachability density (*Lrd*) of data point  $p$ .

**Definition 5.** *LOF of  $p$ .*

With all the above-mentioned work, the LOF score of a data point  $p$  can be calculated through Equation (5).

$$LOF_{MinPts}(p) = \frac{\sum_{n \in N_{MinPts}(p)} \frac{Lrd_{MinPts}(o)}{Lrd_{MinPts}(p)}}{|N_{MinPts}(p)|} \quad (5)$$

Equation (5) calculates the average ratio of the local reachability density of data point  $p$  and the *MinPts*-nearest neighbors of data point  $p$ . Finally, an LOF score is assigned to each data point. To determine whether data point  $p$  is an outlier or not, the threshold score  $\theta$  is used. The strength of the

LOF is that it can identify the local density and determine local outliers. The weaknesses are that it requires a long execution time and is sensitive to the minimum points value.

#### 3.4.1.2 Connectivity-Based Outlier Factor (COF)

The COF algorithm [67] works like the LOF algorithm, except that the density estimation of the COF for the data points works differently. For the LOF, the  $k$ NN is determined by using the Euclidean distance, while the COF uses a different approach to determine the  $k$ NN, called the chaining distance. In particular, the LOF assumes the data points are distributed in a spherical form, but the COF assumes the data points have a linear distribution. In mathematics, the chaining distances are the minimum of the total sum of the distances linking all neighbors. The advantage of the COF is that in datasets that have linear correlation, the COF performs with greater accuracy than the LOF. The COF builds a set path to determine the outlier. The COF weakness is that it requires a longer execution time than the LOF.

#### 3.4.1.3 Local Correlation Integral (LOCI)

In the above-mentioned algorithms, selecting the  $k$  value is an important decision for the performance of an algorithm. However, the Local Correlation Integral (LOCI) uses the maximization approach to address the issue of choosing the  $k$  value [68]. The main idea is to use all possible  $k$  values for each data point and then take the top score. To implement this, the  $r$ -neighbor is defined, which uses the radius  $r$ . The radius  $r$  expands over time. Here, the estimation of local density is obtained by using a half-Gaussian distribution, similar to the LoOP, but the LOCI uses the quantity of data points in neighbors instead of using distances. Additionally, the estimation of local density in the LOCI is obtained by comparing two different sets of neighbors instead of the ratio of local densities. The parameter  $\alpha$  controls the ratio of different neighbors. A disadvantage of the LOCI algorithm is that it requires a lot of time for large datasets because the radius  $r$  has to be expanded from one data point to the next. However, the strength of LOCI is that it can locate the maximum value for all radii.



#### 3.4.1.4 Approximate Local Correlation Integral (aLOCI)

To address the issue of run time in the LOCI, the Approximate Local Correlation Integral (aLOCI) was proposed [68]. To speed up the counting of two neighbors, the aLOCI uses quad trees and places some constraints on  $\alpha$ . The count estimation will be accurate in the aLOCI when the data point is located in the middle of a cell of its quad tree. By contrast, the approximation might be poor if the data point is located at the border of a quad tree. Because of this, multiple quad trees are built with the hope that the approximate tree is perfect for every data point. However, the depth of the tree must be specified. The strength of this method is that it can speed up the process by using quad trees; its weakness is that the tree depth has a significant effect on the performance of the algorithm.

#### 3.4.1.5 Cluster-Based Local Outlier Factor (CBLOF)

The Cluster-Based Local Outlier Factor (CBLOF) works by using a cluster to determine the density area in the dataset [69]. It estimates the density for each cluster. First,  $k$ -means is used to cluster the dataset. Next, the CBLOF uses a heuristic procedure to split the resulting clusters into two categories: large and small. Finally, the calculation of the outlier score is accomplished by using the distance of each data point to the central data point of its cluster, multiplied by the data points that belong to its cluster. In a small cluster, the distance to the nearest large cluster is used. The advantage of CBLOF is that it uses the cluster method instead of the nearest neighbor method. Its disadvantage is that it is sensitive to the  $k$  parameter in the  $k$ -means clustering.

#### 3.4.1.6 Influenced Outlierness (INFLO)

The Influenced Outlierness (INFLO) algorithm [70] is applied when the data points include clusters with diverse densities that are near each other. The LOF algorithm fails to correctly score the data points within the boundaries of the clusters. The  $k$ -NN method is used in the INFLO algorithm. In addition, the set of reverse nearest neighbors, where data points are stored with existing data points, is

considered a neighbor. To calculate the INFLO score, both neighbor sets need to be combined. After this, the local reachability density and the score are calculated as in the LOF algorithm. The INFLO operation is illustrated in Figure 3.4, where the red data point has five nearest neighbors, as can be seen in the blue circle. The red data point will be detected as an outlier because the local density value of its four nearest neighbors is much higher. In the INFLO, the (green) data points are taken into consideration when the red data point is their neighbor. The INFLO algorithm calculates the score of outliers more accurately when the dataset contains multiple clusters of diverse densities that are near to each other. It can also detect outliers in different neighbor density distributions. The disadvantages are that it requires a long time to run, and it is only focused on local outliers.

#### 3.4.1.7 Local Outlier Probability (LoOP)

Similar to the LOF, the Local Outlier Probability (LoOP) uses the set of nearest neighbors to estimate the local density [71]. However, the LoOP calculates the local density differently. While the LOF detects the outlier data points using the score of an outlier, the LoOP detects them by using the probability of an outlier. The main idea is that the distance of a data point to its nearest neighbors follows the Gaussian distribution. Because the distances have positive values, the LoOP presumes a “half-Gaussian” distribution and uses its probabilistic set distance, which is considered the local density. The proportions of each data point are compared to the proportions of its nearest neighbors, which result in the local outlier detection score. Finally, the normalization and a Gaussian error function are implemented to convert the score to a probability. The advantage of LoOP is that it depends on the probability score. The disadvantages of LoOP are that it requires more time to execute and the probabilistic size of the data points may cause incorrect measurements.

#### 3.4.1.8 Local Density Cluster-Based Outlier Factor (LDCOF)

In the CBLOF algorithm, only the quantity of cluster members is used, and the density of the cluster is not considered. The Local Density Cluster-based Outlier Factor (LDCOF) resolves this issue

by taking into consideration an estimation of the cluster densities, where the spherical distribution is assumed for the cluster members [72]. First, the LDCOF applies k-means to cluster the dataset, and then it uses the CBLOF process of dividing the clusters into large and small clusters. Next, the LDCOF calculates the average

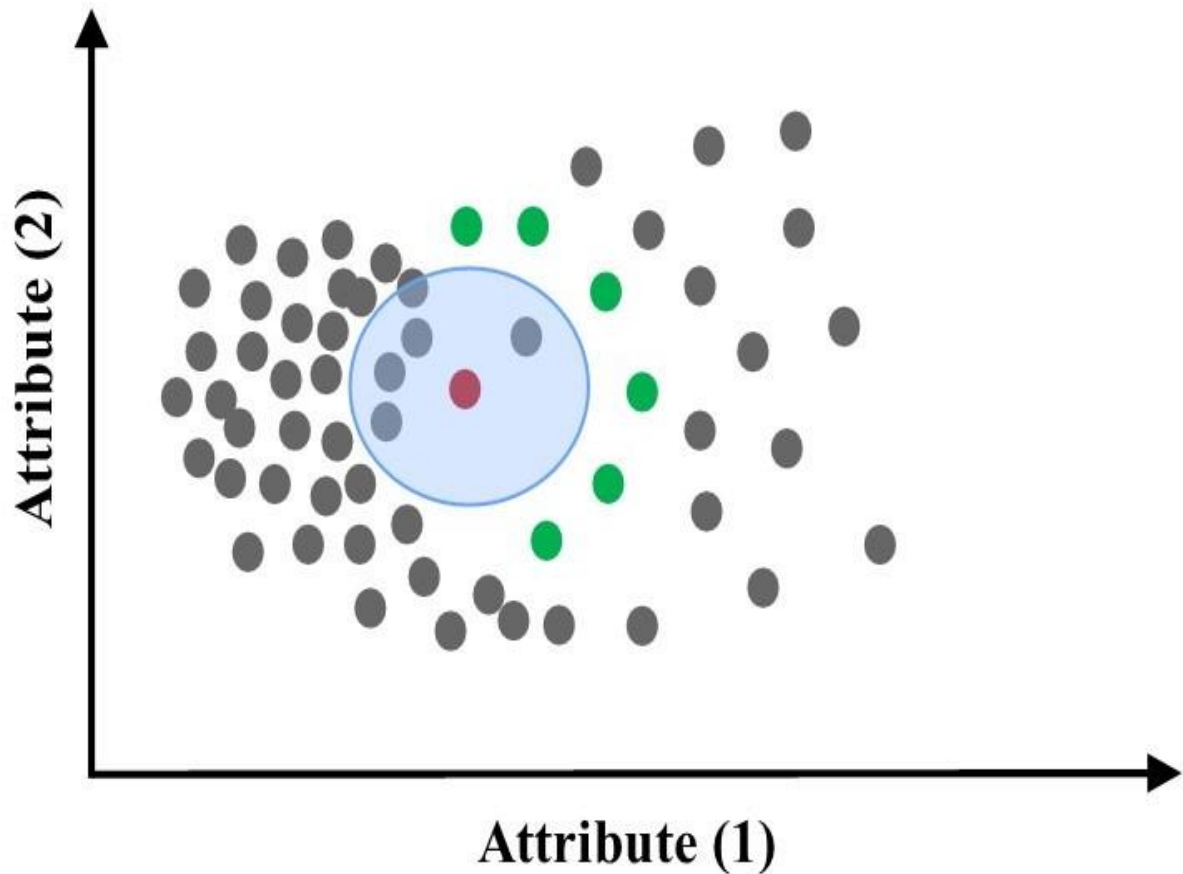


Figure 3.4. Comparison between the LOF and the INFLO. For the red data point, LOF will consider the data points in the blue area to be neighbors, which will result in a high outlier value. By contrast, the INFLO will take into account the green data points so that the value of the red data point will be more reasonable and will be more likely to be considered an outlier.

distance of the data points to the centroid for all clusters. The last step is to calculate the LDCOF value by dividing the distance of each data point to its centroid by the average distance of each point. A strength of the LDCOF is that it estimates cluster densities and presumes a spherical distribution. Its weakness is that it is sensitive to the  $k$  parameter.

### 3.4.1.9 Other Local Outlier Detection Algorithms

Chiu et al. [73] introduced three enhancement schemes, called the LOF', the LOF'', and the GridLOF. The LOF' provides a more intuitive meaning for local outlierness. The advantage of this method is that it introduces another version of the LOF for the minimum number of points, and it can also process large datasets by eliminating *rd* and *lrd*. Its disadvantage is that the minimum points value could be inaccurate depending on the computational results. The LOF'' handles cases that the LOF cannot properly handle by introducing a new calculation method to determine neighbors. The GridLOF improves the efficacy of outlier detection by pruning the inliers. The weaknesses of this method are that it requires a long run time due to its complexity and it is difficult to select its parameters.

Jiang et al. [74] created a novel approach to determine the outlierness score, which is called the generalized local outlier factor (GLOF). They also proposed the rule of  $(k \sigma)$  that does not require a threshold or previous knowledge about the number of outliers in a dataset. The strength of this method is that it does not need thresholds. The weakness of this method is that it depends on the  $k$  value. Ren et al. [75] proposed an algorithm, called the Relative Density Factor (RDF), which detects an outlier using *p-trees*. A data point  $pt$  that has a high RDF score is an outlier. The advantage of this method is that it has better scalability when data size is increased; however, it requires more computational time.

Lozano et al. [76] introduced two parallel algorithms: the first algorithm detects a distance-based outlier, which depends on nested loops, randomization, and a pruning rule; the second algorithm is the parallel LOF algorithm, which detects density-based local outliers. The advantage of these algorithms is the fast execution time. Fan et al. [77] proposed an algorithm called Resolution-based Outlier Factor (ROF). This algorithm solves some impediments, such as poor accuracy. The strength of ROF is that it has a better outlierness score because it uses a growing window. The weaknesses of this method are that ROF cannot rank the outliers properly, cannot deal with different density clusters, and it requires more storage. Momtaz et al. [78] improved the ROF algorithm [77] by developing a new algorithm that detects the top- $n$  outliers by assigning a score for each data point; it is called the Dynamic-Window Outlier Factor (DWOFF). The DWOFF improves the ROF by taking into consideration the

varying-density clusters. This improves the ranking of the outliers even when presenting the same outliers. Its strength is that it identifies a new measurement using *top-n*, which improves outlier detection, while its weakness is that it is sensitive to the parameter choices.

A new density-based local outlier concept based on uncertain data (UDLO) was proposed in [79]. To detect the outlier faster, they built an algorithm that calculates the density of a data point rather than calculating all  $k$ -neighbors in order to detect the outliers, as in the LOF. The strength of this algorithm is it does not need to calculate all the  $k$ -neighbors, but it focuses on Euclidean distance only. The Clustering-based Multivariate Gaussian Outlier Score algorithm (CMGOS) algorithm was proposed by Goldstein [80]. The estimation of local density for CMGOS [28] is achieved using the estimate of the multivariate Gaussian model, where the Mahalanobis distance is used as a basis for the calculation of the outlier score. As a first step, the CMGOS applies the  $k$ -means to cluster the dataset and divides them into large and small clusters. Then the covariance matrix is calculated for each cluster. Next, the CMGOS value is calculated by dividing the Mahalanobis distance for each data point to its closest cluster centroid through the distribution of a chi-square together with a certain confidence interval. A strength of this algorithm is that it uses the multivariate Gaussian model and the Mahalanobis distance to find the outlier score. The weaknesses of this algorithm are that it requires more  $k$  values and it is not suitable for large datasets.

Tang et al. [81] proposed a method for detecting outliers using local kernel density estimation (KDE). They introduced the Relative Density-based Outlier Score (RDOS) to calculate the local outlier score of data points, where the estimation of the density distribution in the position of a data point, using a local KDE, depends on the extended nearest neighbors of the data point. KDE has the advantage of being able to estimate the density around a data point, and RDOS can find the local outlierness of a data point in the local KDE. Vázquez et al. [82] proposed a novel algorithm, called Sparse Data Observers (SDO), to locate outliers based on a model for low-density data. The strength of SDO is that it reduces the complexity of the quadratic formula used as well as the computational time. Ning et al. [83] developed a method, called relative density-based outlier detection, which uses a new technique to

calculate the density of a data point's neighbors. The strength of this method is that it can handle the issues caused by low-density patterns. Su et al. [84] introduced an effective density-based scheme that depends on the local outlier detection approach. It is used for scattered data and is named E2DLOS. Instead of the LOF, they used the Local Deviation Coefficient (LDC) for their measurements. The advantage of this algorithm is that it improves computational efficiency.

Zhao et al. [85] developed a framework, the Locally Selective Combination in Parallel Outlier Ensembles (LSCP) that works by determining a local region (area) around a test data point based on its compatibility with its nearest neighbors in randomly selected feature subspaces. LSCP selects and combines the best performing detectors in that local area. The strength of this algorithm is that it can quantify the degree of the local outliers. A new methodology of tuning the hyper-parameters for the LOF algorithm was proposed in [86]. The strength of this method is that it can handle small and large datasets; its weakness is that the algorithm has to well-sample the training data. Yang et al. [87] developed a new local outlier detection algorithm called the Neighbor Entropy Local Outlier Factor (NELOF) that improves the Self-Organizing Feature Map (SOFM) and uses it to cluster the dataset.

#### 3.4.2 *Local Outlier Detection in Stream Environment*

Despite the contributions of the above research to local outlier detection, their focus has been on static environments rather than on stream environments, in which the data comes as a stream of points rather than a single, static dataset. The environments with streamed data have been much less studied. Thus, there has been growing interest in developing traditional local outlier detection algorithms to work in stream environments. This section reviews the research on local outlier detection algorithms in stream environments and focuses on the LOF-based approaches that contributed effectively to the calculation of the LOF score in these environments. Table 3.2 provides a summary of previous work on LOF in stream environments. Figure 3.5 shows all of the reviewed papers included in this section.

### 3.4.2.1 Overview of Local Outlier Detection Algorithms in Stream Environment

Authors of [88] proposed Incremental Local Outlier Factor (ILOF). The idea of ILOF is to update and calculate the score of LOF in a data stream to determine if the newly incoming data point is an inlier or outlier using landmark window. However, ILOF needs to store all the data points in memory in order to calculate the LOF score. Pokraiaic et al. [89] proposed an incremental COF (connectivity-based outlier factor) whose detection performance in a data stream is equivalent to that of the integrated static connectivity-based outlier factor (COF) algorithm. The contribution of the paper is that it shows that the number of data profiles that updates for every deletion or insertion does not in any way rely on the number of points found in a given set of data. Ren et al. [90] discussed the issue of data stream attributes in memory usage. They presented a new approach, Heterogeneous Data Streams Outlier Detection (HDSOD), to overcome issues in processing data streams. The strength of this approach is that it is based on a heterogeneous data stream that uses the partition-cluster approach for the data stream segments, after which the segment is stored in a reference as a cluster. The outlier result is computed based on the number of cluster references and the degree of representation.

An improved version of the incremental LOF (ILOF) algorithm is introduced by [91], which uses a sliding window. The sliding window allows for updating data profiles, particularly during the window after which the profiles are declared an inlier/outlier. The strength of this paper is that it demonstrates that a false-positive rate could be reduced without incurring extra computational expense. For the Multiple Instance (MI) setting, Wang et al. [92] proposed an incremental MI algorithm, Inc. I-MLOF, for detecting outliers. The algorithm is capable of achieving outlier detection results that are similar to those of I-MLOF, an original LOF extension for MI. This paper demonstrates that simple incremental algorithms can be used to produce outlier detection results that are similar to those of more complex algorithms. The focus of [93] is on local outliers, and hence, an incremental solution was introduced with the assumption that finite memory is available. The advantage of the paper is that it

shows that different outlier detection solutions are suitable for different application environments based on the available memory.

Kontaki et al. [94] presented an algorithm that is applied to the continuous monitoring of streams of data outliers. The basis for the algorithm is sliding windows. The strength of the algorithm is that memory requirements are reduced. The paper also contributed to the introduction of more flexible and efficient techniques for detecting anomalies in streams of data. Zhang et al. [95] presented a technique for detecting faults in streams of data that are highly dimensional and non-stationary. The technique presented is angle-based and can identify sub-space faults that are low-dimensional from high-dimensional dataset spaces. Salehi et al [96] proposed the Memory Efficient Incremental Local Outlier Factor (MILOF). The MILOF algorithm summarizes the old data by using  $k$ -means. However, this method usually performs badly because clustering old data points by using  $k$ -means does not retain the density. Hamlet et al. [97] proposed a new incremental algorithm that is a modification of the LoOP (Local Outlier Probabilities) algorithm, which is usually utilized for anomaly detection. The advantage of the Hamlet et al. algorithm is that it can detect outliers almost immediately. It reduces computational time meaning that low-resource machines can be used in incremental anomaly detection, even on large static sets of data. The disadvantage of this method is that it needs to be executed on the dataset before accepting incremental data points.

Siffer et al. [98] proposed a novel method for detecting outliers, particularly in the streaming of a univariate time series whose basis is the extreme value theory. An advantage of this method is that it does not make any distribution assumptions and it does not need a handset threshold. The objective of this paper was to develop a method where an anomaly can be detected in a more efficient manner. However, its weakness is the gap in terms of the multivariate case. Mu et al. [99] introduced an alternative technique as a solution to the problem of classifying new classes of data streams. The advantage of this work is that it demonstrates that it is possible to make use of random trees to find the solution to three key sub-problems: model updates, supervised learning, and unsupervised learning. The problem with this method is that it does not give true class labels. Ishimtsev et al. [100] considered a



model-free anomaly detection technique that uses a univariate time-series and generates a probabilistic abnormality score. The advantage of this work is that it demonstrated that simple methods, such as the Lazy Drifting Conformal Detector (LDCD), can be used for performing anomaly detection and achieve results similar to those of more complex anomaly detection techniques. The disadvantage of this work is that the procedure of LDCD, required for the suitable validity warranty, needs further research for the Numenta Anomaly Benchmark (NAB) corpus.

A Neuromorphic Anomaly Detection (AnRAD) framework was introduced in [101] that can perform probabilistic inferences. The strength of the framework is that it improves memory efficiency and computing performance during anomaly detection. The paper also describes the improved speed of incremental learning and hence improved anomaly detection efficiency and accuracy. Yao et al. [102] introduced an incremental approach for detecting a local outlier aimed to evaluate the local outlier in a given stream of data in a dynamic manner. The strength of the work is that local outliers can be detected more accurately with advanced  $k$  nearest neighbor-based techniques. The weakness of the work is that it requires long computational time. Munir et al. [103] presented a new approach for detecting an anomaly, which is based on deep learning. The approach is known as DeepAnT and is used on time-series data. The method can also be used in non-streaming cases. The advantage of the approach is that an anomaly can be detected in real-life cases in which labeling a relatively big data stream, as well as static data, is required. Na et al [104] proposed Density Summarization Incremental Local Outlier Factor (DILOF). The DILOF algorithm shows better performance than MILOF by summarizing the old data using gradient descent. However, gradient descent can get stuck in local minima.

Manzoor et al. [105] proposed a method for detecting an outlier that is density based. The method, known as xSTREAM, is suitable for more extreme streaming data. The paper demonstrates that by making use of state-of-the-art methods, such as the xSTREAM, an anomaly can be detected more accurately, regardless of the size of the dimensions of the noise. Yang et al. [106] proposed a rather fast algorithm for outlier detection in data streams. The advantage of their method is a reduction in the time required for anomaly detection because the algorithm minimizes the required calculations in the LOF

while still generating accurate outlier detection. However, it is based on a prediction model. Novel outlier semantics are proposed by Qin et al. [107], which deal with the shortcomings associated with the existing methods in light of modern high-velocity streams of data. The algorithm leverages KDE (kernel density estimation) to enhance the effectiveness of detecting local outliers from streaming data. The strength of this paper is that it shows how KDE can be leveraged to enhance the effectiveness of local outlier detection.

Kalliantzis et al. [108] focused on an outlier detection technique that is distributed and density based. The technique is applied to multi-dimensional streams of data. The strength of the work is that it establishes how the approximation computation method for LOCI (Local Correlation Integral) can be used to enhance the scalability and efficiency of detecting an anomaly, even when a largely distributed dataset is involved. Cai et al. [109] proposed an approach for detecting outliers that is two-phased and pattern-based and is known as Weighted Maximal Frequent Pattern-based Outlier (WMFP). The advantage of this method is that it shows that the impact of outliers, particularly from a weighted stream of data, can be detected more effectively. By accelerating the process of detecting outliers, the maximum frequent patterns are utilized rather than the frequent patterns. Novel methods for detecting and predicting outliers are introduced in [110]. The paper contributes to anomaly detection by showing that ensuring that an appropriate technique is used results in the achievement of accurate results, minimization of memory consumed, and the minimization of computational time.

A new approach for classifying data streams proposed in [111] is known as Evolving Micro-Clusters (EMC). The method learns the evolving micro-clusters in a dynamic manner in order to evaluate the evolution and concept drift. The advantage of this method is that it introduces the possibility of distinguishing evolution and concept drift from the distribution of noise. Alsini et al. [112] proposed a Grid Partition-based Local Outlier Factor (GP-LOF) algorithm for detecting a local outlier in a data stream. The GP-LOF has three phases: the first is the preprocessing phase where a sliding window is used to summarize the data points; the second is the processing phases that uses a grid method to split

the data points, and the third is the detection phase that detects the outliers using the LOF. The strength of the algorithm is that it summarizes the data stream.

### 3.5 Applications of Outlier Detection

#### 3.5.1 Intrusion Detection

Intrusion detection is a popular application of outlier detection. The purpose of its application is to monitor traffic networks. Outlier detection algorithms are used to identify intrusion attempts [113,114]. and programs usually inform the network manager of any suspicious violations and processes that have been observed. Intrusion Detection Systems can be split into two basic types: Network In-trusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS). Intrusion Detection Systems (IDS)

Table 3.2. Previous works summary for LOF in a stream environment.

Authors and Year	Algorithm	Method	Window Form	Time Complexity	S. Technique		Remark
					Opt	Clus	
Pokrajac et al., 2007 [88]	ILOF	Updating data when new data point $np$ is inserted.	Landmark window	$O(N \log N)$			Need to store all data points in the memory, which requires high memory complexity and high time complexity.
Salehi et al., 2016 [96]	MILOF	Summarizing the data by using $k$ -means.	Sliding Window	$O(N \log W_s)$		✓	Summarized data points using $k$ -means which cannot preserve the density of data and addressed the issue of high time complexity in ILOF.
Na et al., 2018 [104]	DILOF	Summarizing the data by using gradient descent.	Sliding Window	$O(N W_s)$		✓	Summarized data points using gradient descent. Addressed the issues of preserving the density of data points in MILOF. In contrast, it might be stuck in the local minima.

Note: Summarization (S), Optimization (Opt), Clustering (Clus).

References							[101]		[110]	
							[100]	[105]	[109]	
						[96]	[99]	[104]	[108]	
				[93]	[95]	[98]	[103]	[107]	[112]	
Year	[88]	[89]	[90]	[91]	[92]	[94]	[97]	[102]	[106]	[111]
	2007	2008	2009	2012	2015	2016	2017	2018	2019	2020

Figure 3.5. Summary of references used in the literature review for streaming data.

are hardware or software that observe the operation of the computer network and detect suspicious data or violations of network management policies. These devices NIDS observes the operations of exchanged data on the network, and continuously compares it with normal data. It can therefore determine suspicious data or violations perpetrated on a network. HIDS works as software on devices and observes any modification or deletion of files. It also observes the outgoing and incoming data from the network to detect any suspicious data transfer between the device and the network [115]. The challenge of intrusion detection applications is the processing of huge amounts of data streams. Therefore, these applications use more basic, but more rapid, outlier detection algorithms.

### 3.5.2 Fraud Detection

Fraud detection is another common application for outlier detection. Fraud detection is used in many industries and other sectors, such as banking, insurance, and law enforcement. It aims to detect and prevent fraud [116]. Fraud detection software analyzes data, such as financial transactions, to detect abnormalities and unusual activities [117,118]. The Internet has become a necessity of life for many people. This has vastly increased e-shopping, which has led to the growth of online payments using credit cards. Outlier detection algorithms are used to identify attempted fraud.

### 3.5.3 Medical Applications

Outlier detection is also employed in many important medical applications. Outlier detection helps doctors monitor patients' conditions when body sensors, such as electrocardiography (ECG), are used to detect critical situations [119,120]. Outlier detection algorithms are often used for analyzing medical images. For example, to detect abnormalities in a computed tomography scan (CT), doctors need to take into consideration the size, density, expansion, shape, structure, and location of suspicious tissue or structural part to compare the abnormality to normal tissue or structure. However, outlier detection algorithms in medical applications depend on image processing as a preprocessing stage.

## 3.6 Computational Complexity

The LOF and the related local outlier detection algorithms are nearest neighbor-based algorithms. The computational complexity of these algorithms is  $O(n^2)$ , except for LOCI. Therefore, the execution times of these algorithms are similar. The computational complexity of the LOCI algorithm is  $O(n^3)$ , which requires more execution time because of the repetition required for the radius expansion. Another version of the LOCI algorithm is the aLOCI, which is faster than the LOCI because the execution time is based on the number of quad trees. The nearest neighbor-based algorithms have better accuracy of outlier detection than cluster-based algorithms [121]. However, long computation times are the cost of these methods because of the long time spent calculating pairwise distances.

When comparing the cluster-based algorithms with nearest neighbor-based algorithms, the cluster-based algorithms are more efficient; for example, the  $k$ -means computational complexity is  $O(nki)$ , where  $n$  is the data point,  $k$  is the center of the cluster, and  $i$  is repetitions. By contrast, the nearest neighbor-base algorithms usually lead to quadratic computational complexity because they calculate the distances of all data points. The computational complexity of cluster-based algorithms, such as CBLOF, depends on the cluster algorithm, which is faster than  $O(n^2)$  when using  $k$ -means. Therefore, the clustering algorithm is the essential reason for the computational complexity.

The most important issue for the LOF and its extensions is application in a stream environment. Few works exist in this domain, but some of them show better execution times. The ILOF algorithm addresses the issue of the LOF in a stream environment by updating and calculating the score of the LOF when a new data point arrives by using a landmark window. The computational complexity of the ILOF is  $O(N \log N)$  and the memory usage complexity is  $O(Nk)$ . The MILOF algorithm solved the issue of unlimited memory in the ILOF by summarizing the data points using  $k$ -means clustering and sliding windows, which limits the memory requirement. The computational complexity of the MILOF is  $O(N \log Ws)$  and the memory usage complexity is  $O(Ws k)$ , where  $N$  is the size of the data stream and  $Ws$  is the window size. The DILOF algorithm overcomes the issue of unbounded memory requirements by summarizing the data points using gradient-descent and sliding windows. The computational complexity of the DILOF is  $O(N Ws)$  and the memory usage complexity is  $O(Ws k)$ . Finally, the MILOF and the DILOF solve the issue of the ILOF by summarizing and keeping only a small portion of the data points, which leads to reduced execution time.

### 3.7 Strengths and Weaknesses

#### 3.7.1 *Nearest Neighbor-Based Techniques*

Distance and similarity metrics can be determined by different approaches in the nearest neighbor-based outlier detection methods. The advantage of this approach is that it does not need an assumption for the data distribution and can be applied to different data types. However, it requires an appropriate distance calculation for the data. The Euclidean distance is the optimal approach for serving the outlier detection in continuous characteristics such as data streams [122]. The methods depend on two approaches:

- The use of techniques to measure the distance of the data point as for the outlier score.
- To identify the outlier score, the calculation of the relative density of each data point.

However, nearest neighbor-based techniques require a long computational time for big data.

### 3.7.2 *Clustering-Based Techniques*

In data mining, the clustering technique is a common alternative for clustering data with similar features [123,124]. Clustering is also an effective method for the study of outliers. In most of the clustering approaches, the primary assumption is that the normal data is often bound to density and significant clusters, where outliers can be separated into other classes [125]. The advantages of cluster approaches include the following:

- With the incremental model, it is simple to adjust.
- No oversight is required.
- Suitable for temporal data to detect outliers.
- Requires only a quick test step since the number of clusters needing comparison is typically small.

The disadvantages in the clustering-based techniques are the following:

- They depend strongly on the efficiency of the clustering algorithm for normal data points.
- The majority of approaches that identify outliers are cluster by-products and thus are not designed to perform well for detecting outliers.
- Several cluster approaches process each point to be distributed in some clusters. This could contribute to abnormalities in a large cluster, and techniques that work under the presumption that anomalies are included in each cluster may be viewed as normal data points.
- Some algorithms demand that each data point is allocated on a cluster. A wide cluster may be used for outliers and handled by methods that often conclude that outliers are isolated.
- Various approaches to the cluster are only applicable where outliers are not part of the main clusters.
- The measurement of the clustering algorithm is complicated.

### **3.8 Conclusion**

Outlier detection is an important task for many applications in many different domains. The aim of outlier detection is to find abnormal events that differ from normal events. This chapter reviews local outlier detection algorithms in both static and stream environments. More specifically, it addresses the challenges of local outlier detection in stream environments and compares the different methods of local outlier detection that have been proposed. Extra attention has been given to the LOF in stream environments. Based on the results of this review, the chapter also provides the strengths and weaknesses of the LOF in stream environments.



## Chapter 4: The Genetic-Based Incremental Local Outlier Factor (GILOF) Algorithm

"A Genetic-Based Incremental Local Outlier Factor Algorithm for Efficient Data Stream Processing." *the 4th International Conference on Compute and Data Analysis*, 2020 pp. 38-49, ACM.

### 4.1 Introduction

Outlier detection is a method in data mining, which has obtained a lot of interest in the machine learning domain because it is important to detect unusual activities for practical applications, such as the credit card fraud and intrusion detection. In fact, many algorithms have been developed and applied to detect outliers in big data for static environments. In contrast, streaming data is a form of big data that keeps growing indefinitely and, as such, a unique feature of streaming data is its velocity. As the data keeps growing, it cannot be stored as a whole in the computer memory [1]. Specific designs are needed for outlier detection in data stream processing.

Distance-based outlier detection algorithms are popular because they detect outliers without assuming an underlying distribution of data. Despite the popularity of distance-based techniques, they have poor accuracy in analysis of multi-density data. In contrast, density-based outlier detection techniques have demonstrated their ability to find outliers in multi-density data. Local Outlier Factor (LOF) is a commonly used density-based technique [58]. Nevertheless, LOF faces two challenging issues in real-world applications. First, it requires a lot of memory to store data points in the whole dataset and the distances between data points. Second, for any change in the dataset, LOF needs to be recalculated from the beginning for the whole dataset, which is inefficient.

Authors of [88] proposed an incremental technique for LOF called Incremental Local Outlier Factor (ILOF). This technique calculates the LOF in data streams, but it also needs the whole dataset to detect the outliers in a data stream. Moreover, ILOF cannot detect the sequence of outliers that represent a new type of outlier, such as outliers that represent a new type of attack. To solve these issues, the Memory efficient ILOF (MILOF) and Density summarizing ILOF (DILOF) algorithms were proposed. MILOF selects a few data points to store in memory by using  $k$ -means clustering of old data [96].

However, this approach often performs significantly worse for some datasets because clustering old data by using  $k$ -means does not maintain the density of data. Also, it cannot detect the sequences (series) of outliers in a dataset. Researchers in [104] proposed DILOF to detect outliers by summarizing the old data. They used gradient descent and a skipping scheme to detect sequences of outliers. So far, DILOF has the best performance among recently developed algorithms. However, the gradient descent can get stuck in local minima.

To further improve the accuracy and efficiency of outlier detection in data streams, in this chapter we propose a new algorithm called Genetic-based ILOF (GILOF), which is based on the Genetic Algorithm (GA). Like gradient descent, GAs can become stuck in local minima. However, because they are a population-based search technique and use a crossover operator to more widely explore the search space they are generally better than simple gradient descent for searching complex spaces with many local minima. Compared to the existing ILOF algorithms, GILOF has the following characteristics. First, GILOF is working under the limited memory. So, it is capable of handling the memory limitation by summarizing the data points. Second, it detects the outliers using ILOF. This new GA-based process is the major contribution of this chapter. In this research, GILOF was evaluated through a series of experiments using real-world datasets. The results show that the performance of GILOF is better than DILOF in several datasets.

The remainder of this chapter is organized as follows: The methodology is described in section 2. Section 3 shows the concept drift. Section 4 present the experiment. Section 5 presents and discusses the results of the experiments, and the conclusion of this chapter is presented in section 6.

## **4.2 The Methodology**

To address the limitation of DILOF and MILOF, we propose GILOF in this paper. The primary purpose of GILOF is to measure LOF under the following situation: A small part of the dataset is stored in memory, no previous knowledge of the data distribution exists, and outlier detection for the data point

$p_t$  should be done at the current time  $T$ . The algorithm has no previous knowledge about the future data points in detecting the current outliers.

Like DILOF, GILOF includes two phases: detection and summarization. During the detection phase, we used ILOF together with a skipping scheme [88,104]. During the summarization phase, we used the Genetic Density Summarization (GDS), which will be described in detail in the following text.

---

**Algorithm 1:** GILOF

---

**Input:** infinite data streams  $P=\{p_1, p_2, \dots, p_t, \dots\}$ ,  
maximum window size  $W$   
threshold of LOF scores  $\theta$   
population size  $PS$   
number of chromosomes  $NC$   
number of generations  $NG$

- 1  $X \leftarrow \{\}$  // is the data points in the memory
- 2  $X' \leftarrow \{\}$  // is the oldest 50% data points in the memory
- 3  $O \leftarrow \{\}$  // is the detected outliers
- 4 Skipping\_Scheme
- 5 **For each**  $p_t \in P$  **do**
- 6      $LOF_k(p_t) \leftarrow \text{ILOF}(p_t, O, \theta)$
- 7     **If**  $LOF_k(p_t) > 0$  **then**
- 8          $X \leftarrow X \cup \{p_t\}$
- 9         **If**  $|X| = W$  **then**
- 10              $Z \leftarrow \text{GDS}(X', PS, NC, NG)$
- 11             Delete the oldest 50%  $W$  data points in  $X$
- 12              $X' \leftarrow X' \cup Z$
- 13         **End**
- 14     **End**
- 15 **End**

---

The proposed GILOF algorithm is described in Algorithm 1, and it works as follows: In the beginning, it determines the maximum window size (number of points) as  $W$ . Then, the LOF threshold is used to detect the outliers depending on the threshold  $\theta$ , then applies the GDS for the summarization phase. After that, when the new data point arrives, the GILOF algorithm detects the outlier using the ILOF (lines 5-6 in Algorithm 1). GILOF will continue detecting outliers and computing the LOF score for each new data point until, the existing window reaches the  $W$  size. After that, GDS applies over the window to summarize the old 50%  $W$  data points in the window by selecting the best possible 25%  $W$  data points to represent the old 50%  $W$  data points (Figure 4.1). Then, the old 50%  $W$  data points will be

deleted from the window and the selected 25%  $W$  data points will be inserted to the window to be with the remaining 50%  $W$  (lines 9-13 in Algorithm 1). Those 75%  $W$  data points will be selected to join with other new incoming data points in the window. When the window is full again (100%  $W$ ), the GDS repeats the process. The GILOF video [168] displays a simulation of how the GDS algorithm works.

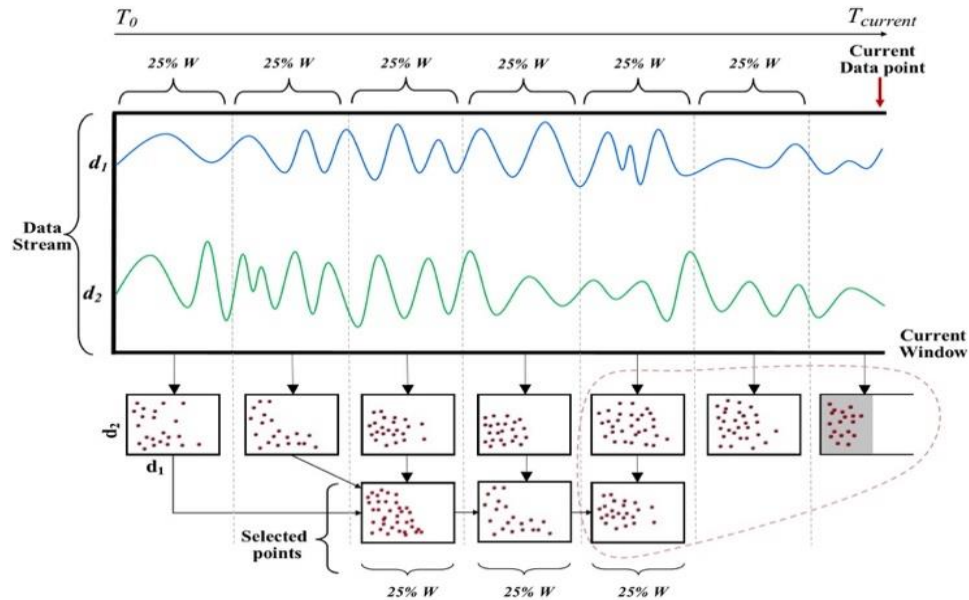


Figure 4.1. GILOF process for a data stream in two dimensional from time  $T_0$  to  $T_{current}$ .

#### 4.2.1 Detection Phase

There are two versions of Genetic based Incremental Local Outlier Factor (GILOF) algorithm. The first one is named GILOF which uses ILOF algorithm with skipping scheme to detect the local outlier. The second version is GILOF-NS which is not using skipping scheme. Below is an explanation of ILOF and skipping scheme procedures.

##### 4.2.1.1 Incremental Local Outlier Factor (ILOF)

The concept of ILOF is to calculate and update the LOF score based on each data point on the data stream, and thus to determine if a data point is an outlier or not [88]. The LOF algorithm has its drawback when processing a data stream, for which the dataset is dynamic and new data points  $np$

continuously appear. It requires an update of the  $k$ -distance, reachability distance, and  $Lrd$  for the LOF value based on  $np$ . For this reason, the insert method of ILOF deals with a new data point  $np$  in two steps. First, it calculates the reachability distance,  $Lrd$ , and LOF score of  $np$ . Second, it updates the  $k$ -distance, reachability distance,  $Lrd$ , and LOF score for the existing data points [88].

However, a main issue of ILOF is that it keeps all the old data points in the memory to calculate the LOF value for every streaming data point that arrives at different times. Thus, ILOF requires long-term and large memory usage.

#### 4.2.1.2 Skipping Scheme

In the detection phase of GILOF, the skipping scheme method keeps any new data point in the low density of the region, because it works together with ILOF to detect a sequence of outliers. Also, it ensures that if any new data point emerges with a new class, it will not affect the accuracy of outlier detection. The skipping scheme works by using the Euclidean distance in two steps. First, it calculates the distance between the data point  $p$  and the nearest neighbor to get the average of the distance  $\overline{d_1(p)}$ . Second, it calculates the distance between the last detected outlier  $o_i$  and the current data point  $p_i$  based on the condition of the return value. If the return value  $d(p_i, o_i)$  is less than the average of the distance  $\overline{d_1(p)}$ , then  $p_i$  is considered an outlier and the method returns true. Otherwise, the method will return false. For more details of the algorithm readers can refer to [104].

### 4.2.2 Summarization Phase

#### 4.2.2.1 Genetic Density Summarization (GDS)

In order to find the optimization, the GDS algorithm summarizes the data points in the existing window by selecting a few data points to minimize the density differences between the old 50%  $W$  data points and the new 25%  $W$  data points. The GDS algorithm works like the Non-parametric Density Summarization (NDS) algorithm in [104]. The NDS algorithm is a part of DILOF algorithm [104]. NDS

algorithm used the gradient descent method to minimize the density difference, whereas GDS uses GA to apply the minimization of the density difference. GA is hypothesized to be better than the gradient descent because it is able to find enough optimal solutions and avoids the local minima.

In Algorithm 2, the GDS algorithm starts by creating a population, where the individuals in the population contains chromosomes. After that, the Objective Function ( $Of$ ) is applied to evaluate all chromosomes. The  $Of$  is the sum of  $a_k(x'_n)$ .

$$Of = \sum_{n=1}^{50\%W} a_k(x'_n) \quad (6)$$

Where  $a_k(x'_n)$  is the density of  $x'_n$  according to its  $k^{\text{th}}$  nearest neighbors (lines 1-6 in Algorithm 2).

Then, for each generation, GDS first applies the selection method to the existing generation. Then GDS applies a crossover method for a pair of chromosomes. After that, the GDS applies the mutation method for pair of chromosomes too. When GA operations are completed for all generations, GDS will keep the best result (lines 7-12 in Algorithm 2). Following the above steps, the selected chromosomes are transformed into  $S$ . Then,  $S$  is projected into the binary domain, i.e., the best 25% of  $s_n$  in  $S$  is set to 1 and the remaining to 0. The GDS algorithm will select a data point  $x'_n$  in  $X'$  if its corresponding  $s_n$  in  $S$  equals to 1 (Figure 4.2). Finally, the selected data points will be output as  $Z$  (lines 13-20 in Algorithm 2).

### 4.3 Concept Drift

Concept drift refers to unexpected change occurring over time in the underlying data distribution and to changes in the collected data over time. Concept drift as part of machine learning and data mining aims to predict the target value or class value. The processes accuracy can be increased over time or finding the missing data value of certain variables [169, 170].

---

**Algorithm 2: GDS**


---

**Input:** set of data points  $X' = \{x'_1, x'_2, \dots, x'_{50\%W}\}$ ,  
population size  $PS$   
chromosomes  $C$   
number of chromosomes  $NC$   
number of generations  $NG$

**Output:** represented the selected data points  $Z$

- 1  $P \leftarrow \{\}$  populations
- 2 Create Population  $P$
- 3 **For each**  $p \in P$  **do**
- 4   Create chromosomes  $C$  in  $p$
- 5   Apply Objective (Fitness) Function to  $C$ .
- 6 **End**
- 7 **For each**  $G$  in  $NG$
- 8   Apply RWS selection to  $G$ .
- 9   Apply cross-over to  $G$ . (two-point random cross-over).
- 10   Apply mutation to  $G$ . (BDM mutation)
- 11   Keep the best results to next  $G$
- 12 **End**
- 13   Transmit the result (chromosome) to  $S$
- 14   Project  $S$  into a binary domain
- 15 **For**  $n=1:50\%W$  **do**
- 16   **If**  $s_n=1$  **then**
- 17      $Z \leftarrow Z \cup \{x'_n\}$
- 18   **End**
- 19 **End**
- 20 **Return**  $Z$

---

In our work, the streaming data occurs when a new data element is inserted into the dataset. ILOF attempts to determine if each new data element is an outlier or inlier by following two steps. First, it calculates the *reachability distance*,  $Lrd$ , and *LOF* score of incoming data point. Second, it updates the *k*-distance, *reachability distance*,  $Lrd$ , and *LOF* score for the existing data points. The aim of the skipping scheme is to predict the sequence of outliers. When a sequence of outlier data points is inserted into the window, the skipping scheme attempts to detect this sequence of outliers.

Because a genetic algorithm (GA) evolves over time, and learns from the data itself, when a change occurs, a GA is inherently capable of handling concept drift without a separate complex computation [151].

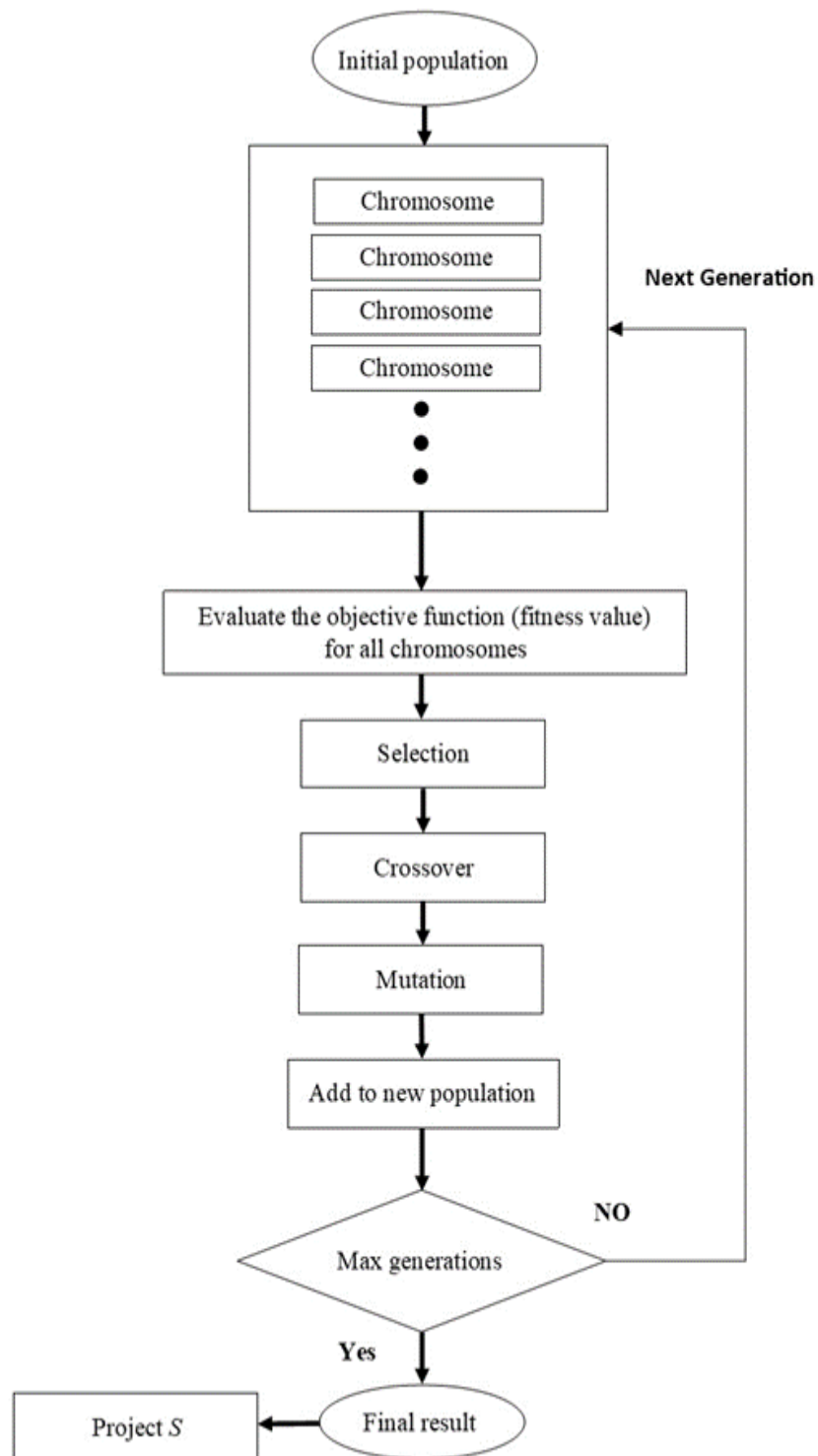


Figure 4.2. GDS flowchart.



## 4.4 Experiment

This section presents the experiment results of GILOF. We only compared the result with the state-of-the-art DILOF algorithm, as DILOF has already demonstrated advantage over MILOF. To make the comparison more comprehensive, we also tested GILOF and DILOF without the skipping scheme (GILOF\_NS and DILOF\_NS). The objective of the experiment is to answer the following two questions:

- 1) Do GILOF and GILOF\_NS perform better than DILOF and DILOF\_NS in accuracy of outlier detection?
- 2) Do GILOF and GILOF\_NS perform better than DILOF and DILOF\_NS in execution time?

### 4.4.1 Experiment Procedures

We compared our proposed algorithms GILOF and GILOF\_NS with DILOF and DILOF\_NS, which is the state-of-the-art competitors. We also listed the results of ILOF as a background for both GILOF and DILOF. The following metrics were used: 1) Accuracy of outlier detection and 2) Execution time. In particular, the Area Under the ROC Curve (AUC) method [171,172] was used for the first metric. All algorithms were implemented in C++ on a machine that has an Intel(R) Core(MT) i5-8250U CPU, 8GB RAM, 250 GB SSD hard disk, and Windows 10 (64-bit) operating system.

We tested GILOF over different window sizes with normalized and unnormalized real-world datasets as described in Table 2.1. The UCI Vowel and Pendigit datasets were from the Machine Learning database repository at UCI [146], and the KDD Cup99 datasets were from [147]. The Vowel dataset was modified to a data stream format such as in [104,173]. The 5 percent uniform noise was added to the Pendigit dataset, where the noised data points were considered as outliers [104]. In the KDD Cup99 datasets, we did the same settings of [174] where the network attacks were considered as outliers.

For DILOF, the hyperparameters were set as in [104], the  $\eta$  and  $\lambda$  values were equal to 0.3 and 0.001 respectively for each dataset. The hyperparameters of GILOF are fixed as follows: the population

size ( $PS$ ) and number of generations ( $NG$ ) were equal to 2 and 4. The selection type was RWS, two-point crossover was used at rate of 0.7, and BDM mutation was used at rate of 0.07 for each dataset [175]. Those values of the hyperparameters in GILOF were determined after a series of tests. Furthermore, memory consumption was tested for all algorithms and the results are available at [176].

To obtain better results, we took the average (AVG) results of ten iterations for all algorithms. The parameter of  $kNN$  for datasets in those algorithms was assigned as follows: for UCI Vowel,  $k$  was 19 for both unnormalized and normalized datasets in all algorithms, except that in ILOF it was 9 when the dataset was unnormalized and 7 when it was normalized. For UCI Pendigit,  $k$  was 9 for the unnormalized dataset and 18 for the normalized dataset in all the algorithms, except that in ILOF it was 55 for normalized dataset and 8 for unnormalized dataset. For KDD Cup 99 smtp and KDD Cup 99 http datasets,  $k$  was 8 for unnormalized and normalized datasets in all algorithms. We did not use ILOF to process KDD Cup 99 smtp and http datasets due to an unacceptable execution time.

GILOF, GILOF\_NS, DILOF, and DILOF\_NS all contained a summarization phase which was executed when the incoming data points reached window size  $W$ . According to the characteristics of datasets, we chose six standard window sizes for UCI Vowel and UCI Pendigit datasets,  $W = \{100, 120, 140, 160, 180, 200\}$ . For KDD Cup 99 smtp and http datasets, we chose four standard window sizes,  $W = \{100, 200, 300, 400\}$ . Also, we chose a large window size  $W = \{1000\}$  for all datasets to test the accuracy of outlier detection in high dimensional data.

## 4.5 Results Discussion

### 4.5.1 Accuracy of Outlier Detection

We tested the accuracy of GILOF, GILOF\_NS, ILOF, DILOF and DILOF\_NS in four datasets (Table 2.1). As mentioned before, the ILOF was not applied to KDD Cup 99 smtp and http due to the inadmissible execution time. Figures 4.4, 4.5, 4.8, 4.9, 4.12, 4.13, 4.16, and 4.17 present the accuracy results of all algorithms for unnormalized and normalized datasets, respectively.

For the UCI Vowel dataset, the GILOF accuracy results were higher than DILOF in all window

sizes (Figures 4.4; and 4.5, Tables 4.1 and 4.2). GILOF, GILOF\_NS and DILOF\_NS algorithms reached the accuracy of ILOF which stored the whole dataset in the memory when  $W = 200$ , while DILOF did not reach ILOF accuracy when the dataset was unnormalized. For the normalized UCI Vowel dataset, GILOF, GILOF\_NS and DILOF\_NS reached the accuracy of ILOF when  $W = 200$ . GILOF\_NS surpassed all algorithms and showed the best accuracy.

For the normalized UCI Pendigit dataset (Figure 4.9; and Table 4.3 and 4.4), GILOF showed better accuracy than DILOF in all  $W$  sizes. GILOF and GILOF\_NS reached the accuracy of ILOF when  $W \geq 140$ . In contrast, all algorithms showed poor results when the dataset was unnormalized (Figure 4.8).

For the unnormalized KDD Cup 99 smtp dataset (Figure 4.12; and Table 4.5), GILOF\_NS showed better accuracy when  $W = 200$  and other algorithms were close to GILOF\_NS. After that, DILOF dropped down when  $W = 300$  and  $400$ . In addition, DILOF\_NS surpassed all algorithms when  $W = 300$  and when  $W = 400$ , GILOF, GILOF\_NS and DILOF\_NS performed almost the same. For normalized KDD Cup 99 smtp (4.13), DILOF\_NS surpassed other algorithms when  $W \geq 200$ , while DILOF dropped down and showed a lower accuracy. GILOF and GILOF\_NS showed a little wobbling result; however, they were close to DILOF\_NS when  $W \geq 300$ .

For the unnormalized KDD Cup 99 http dataset that has sequence outliers (Figure 4.16; Table 4.7), GILOF\_NS and DILOF\_NS showed poor accuracy results compared to GILOF and DILOF, and the performance of DILOF was slightly better than GILOF. When the dataset was normalized (Figure 4.17; Table 4.8), GILOF\_NS and DILOF\_NS performed better, and they both were in the same level when  $W = 100$  and  $400$ , while DILOF\_NS was slightly better than GILOF\_NS in other window sizes. On another hand, the DILOF accuracy result was slightly better than GILOF, and they reached the same level when  $W = 400$ . In addition, our GitHub page provides the rest of the result tables for all algorithms [176].

#### 4.5.1.1 Accuracy of Outlier Detection in Large Window Size

We tested the accuracy of all algorithms in the large window size of  $W = 1000$ . Actually, the GILOF and GILOF\_NS were applicable to work in large window size better than DILOF and DILOF\_NS, especially in a dataset that has high dimensional data [177] (Figure 4.3). For UCI Vowel dataset, the GILOF and GILOF\_NS showed slightly better accuracy than DILOF and DILOF\_NS in the normalized and unnormalized dataset. For UCI Pendigit dataset, GILOF and GILOF\_NS showed better accuracy than DILOF and DILOF\_NS in the normalized dataset. However, all algorithms have poor results in the unnormalized dataset. In KDD Cup 99 smtp, GILOF and DILOF have the same accuracy and DILOF\_NS is slightly better than GILOF\_NS when the dataset is normalized. GILOF and DILOF have the same accuracy when the dataset is unnormalized, while DILOF\_NS was slightly better than GILOF\_NS. For KDD Cup 99 http, GILOF showed a slightly better accuracy than DILOF. GILOF\_NS and DILOF\_NS have almost the same accuracy when the dataset is normalized. When the dataset is unnormalized, DILOF showed slightly better accuracy than GILOF. GILOF\_NS and DILOF\_NS showed poor results.

#### 4.5.2 Execution Time

Figures 4.6, 4.7, 4.10, 4.11, 4.14, 4.15, 4.18, and 4.19 present the execution time results of GILOF, GILOF\_NS, ILOF, DILOF, and DILOF\_NS for the unnormalized and normalized datasets, respectively. In general, GILOF was slightly better in execution time than other algorithms in most standard window sizes. ILOF took approximately 20 seconds for the UCI Vowel dataset and 300 to 322 seconds for the UCI Pendigit dataset. The execution times of GILOF, GILOF\_NS, DILOF, and DILOF\_NS were close to each other, as the difference between them was very small. However, GILOF took 0.72 to 1.26 seconds on the UCI Vowel dataset and DILOF took 0.72 to 1.27 seconds (Figure 4.6 and 4.7). For the UCI Pendigit dataset GILOF took 1.94 to 3.46 seconds and DILOF took 1.87 to 3.48 seconds (Figure 4.10 and 4.11). For KDD Cup 99 smtp and KDD Cup 99 http, the ILOF was not executed due to inadmissible execution time. On the smtp dataset, GILOF took 46.4 to 173.5 seconds

and DILOF took 42.8 to 182.6 seconds (Figures 4.14 and 4.15). On the http dataset, GILOF took 265.8 to 1028 seconds and DILOF took 253.8 to 1073 seconds (Figures 4.18 and 4.19; and Tables 4.1 to 4.8). Further, GILOF\_NS and DILOF\_NS were slightly slower than GILOF and DILOF in execution time. Moreover, GILOF and GILOF\_NS have better execution time when  $W = 1000$  (Tables 4.1 to 4.8).

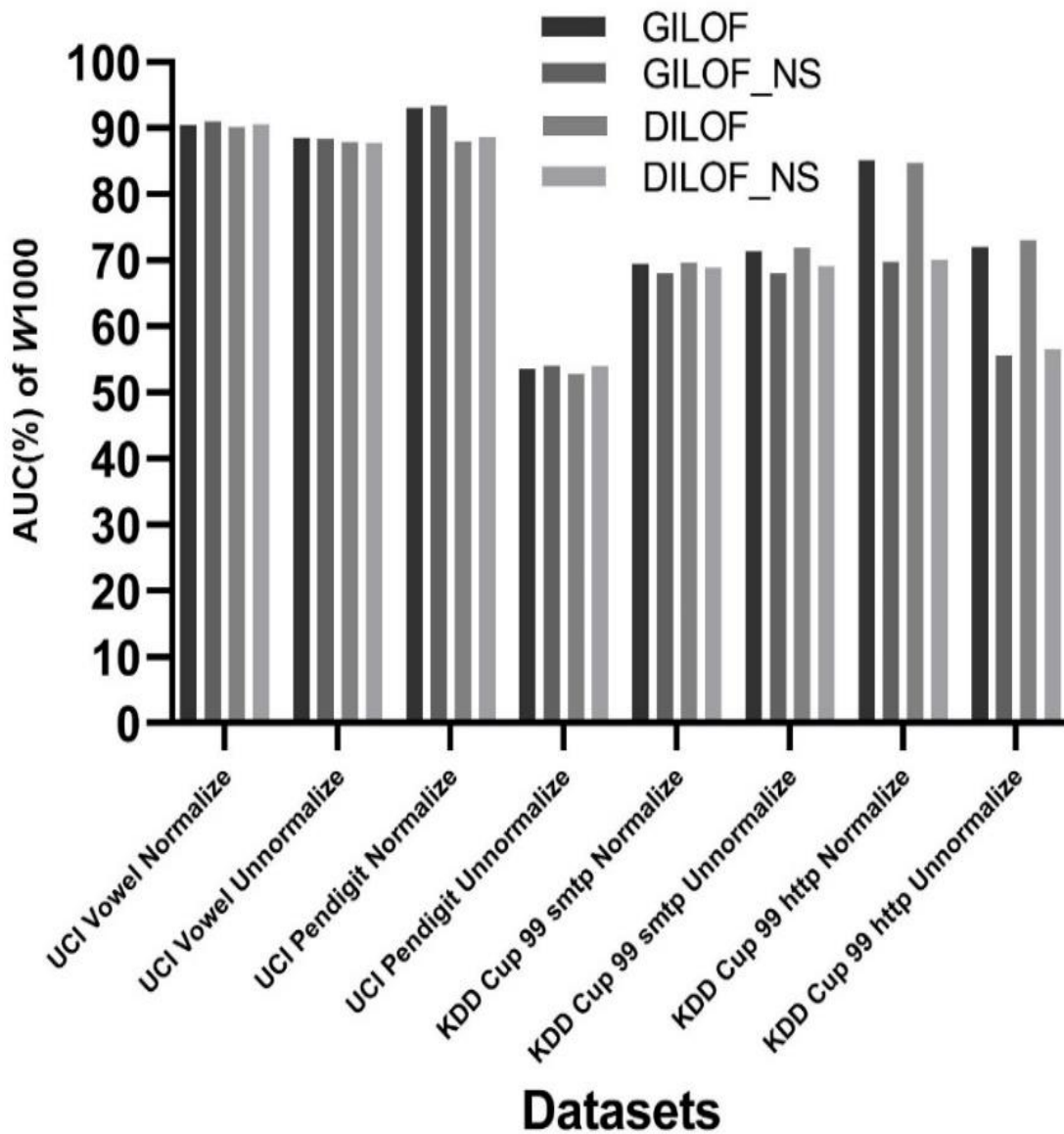


Figure 4.3. The accuracy of outlier detection for all datasets when  $W = 1000$ .

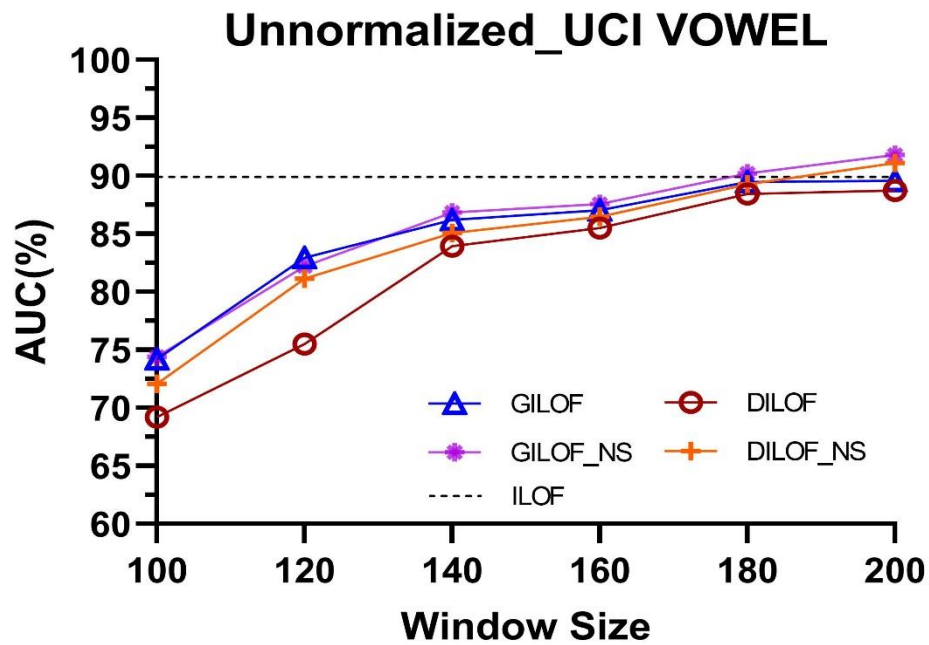


Figure 4.4 The comparisons of outlier detection accuracy for unnormalized UCI VOWEL datasets.

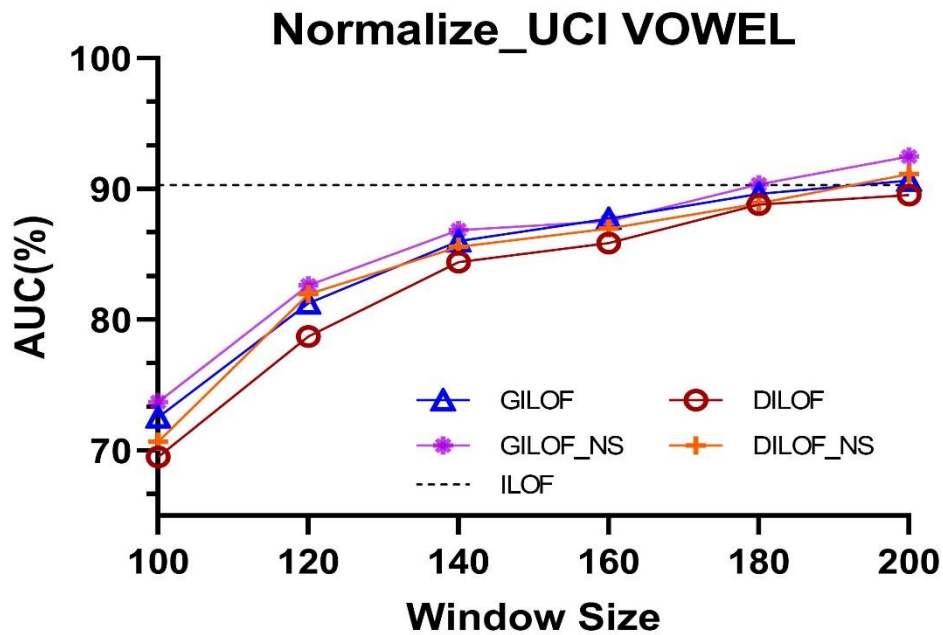


Figure 4.5 The comparisons of outlier detection accuracy for normalized UCI VOWEL datasets.

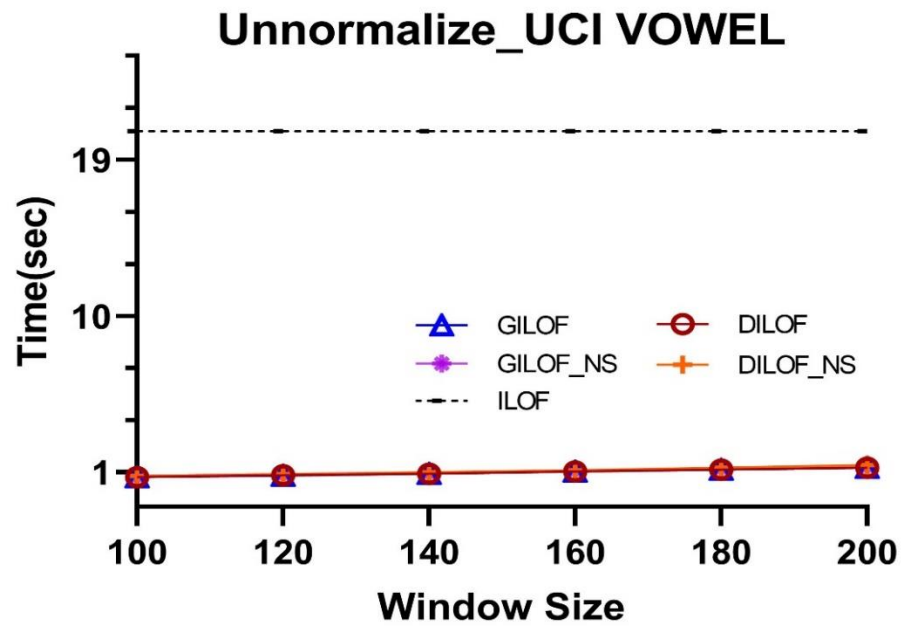


Figure 4.6 The comparisons of outlier detection execution time for unnormalized UCI VOWEL datasets.

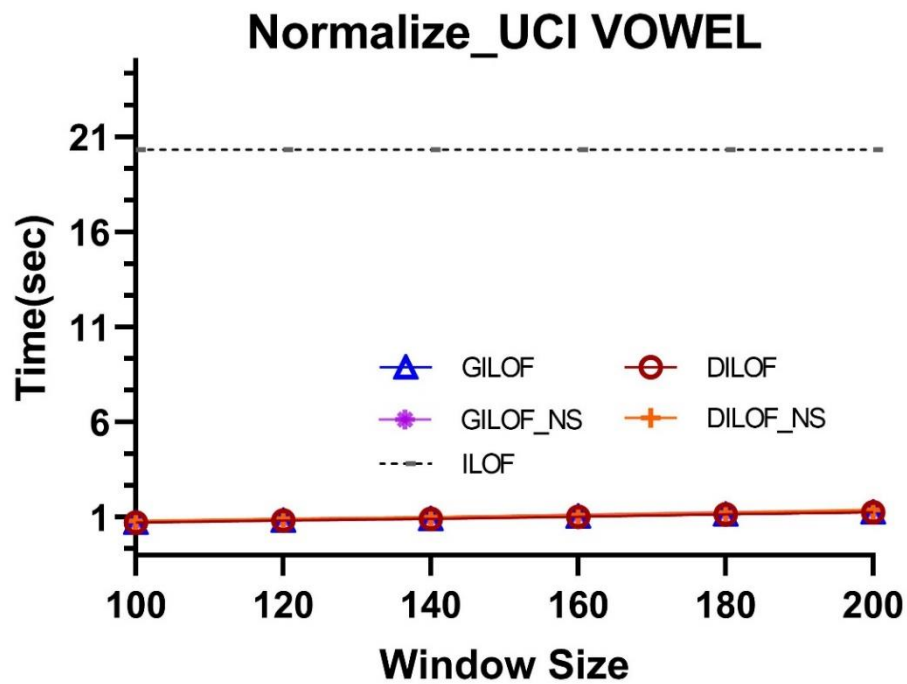


Figure 4.7 The comparisons of outlier detection execution time for normalized UCI VOWEL datasets.

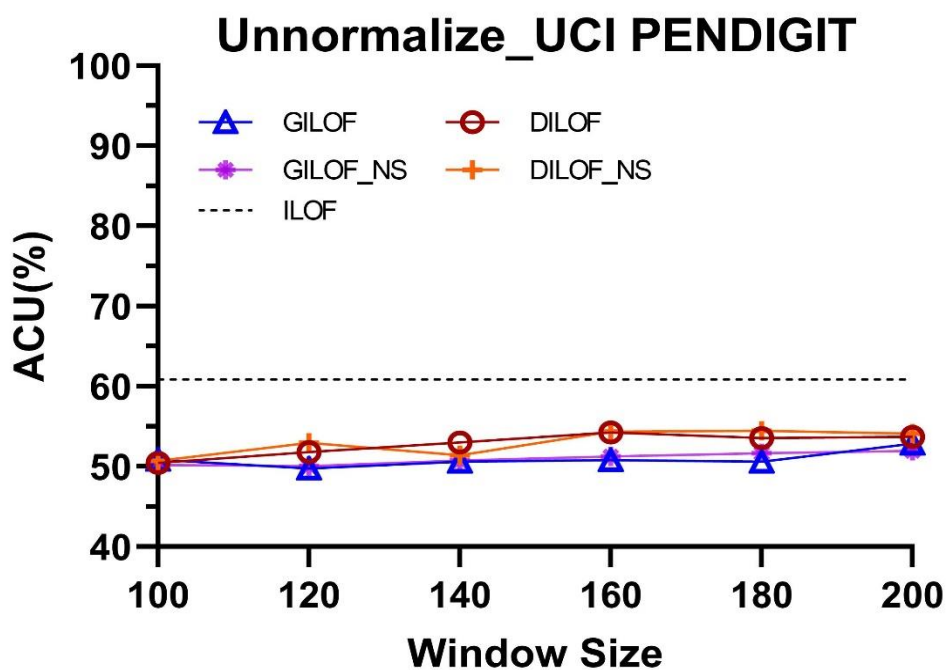


Figure 4.8. The comparisons of outlier detection accuracy for unnormalized UCI Pendigit datasets.

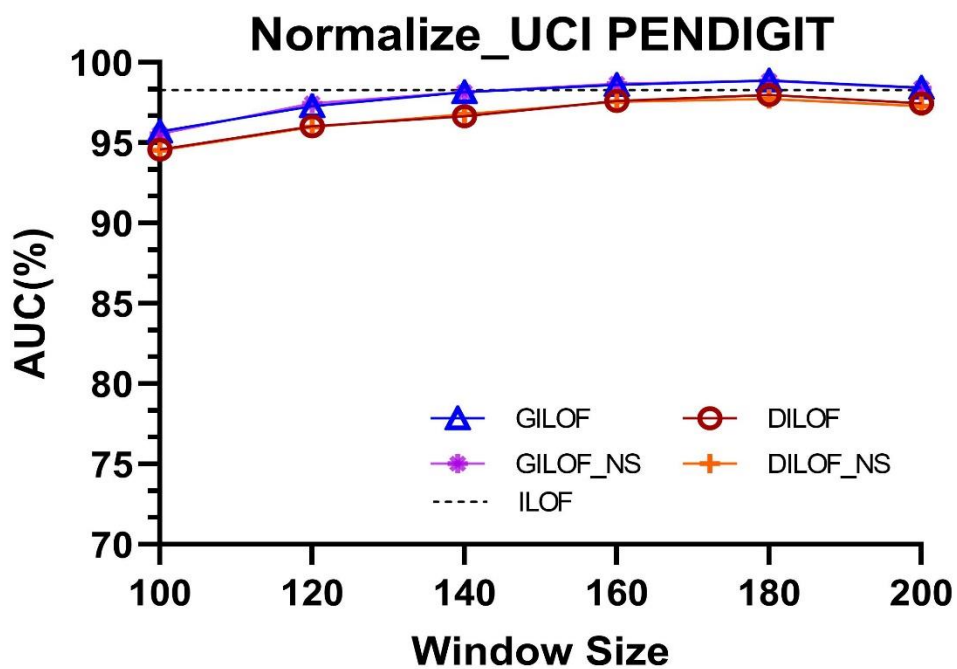


Figure 4.9. The comparisons of outlier detection accuracy for normalized UCI Pendigit datasets.



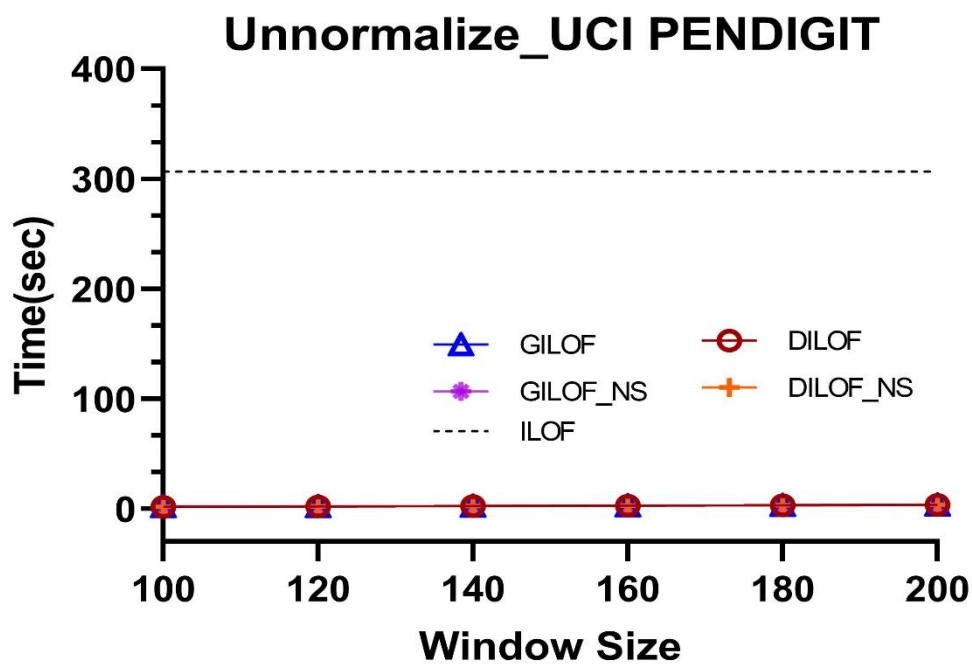


Figure 4.10. The comparisons of outlier detection execution time for unnormalized UCI Pendigit datasets.

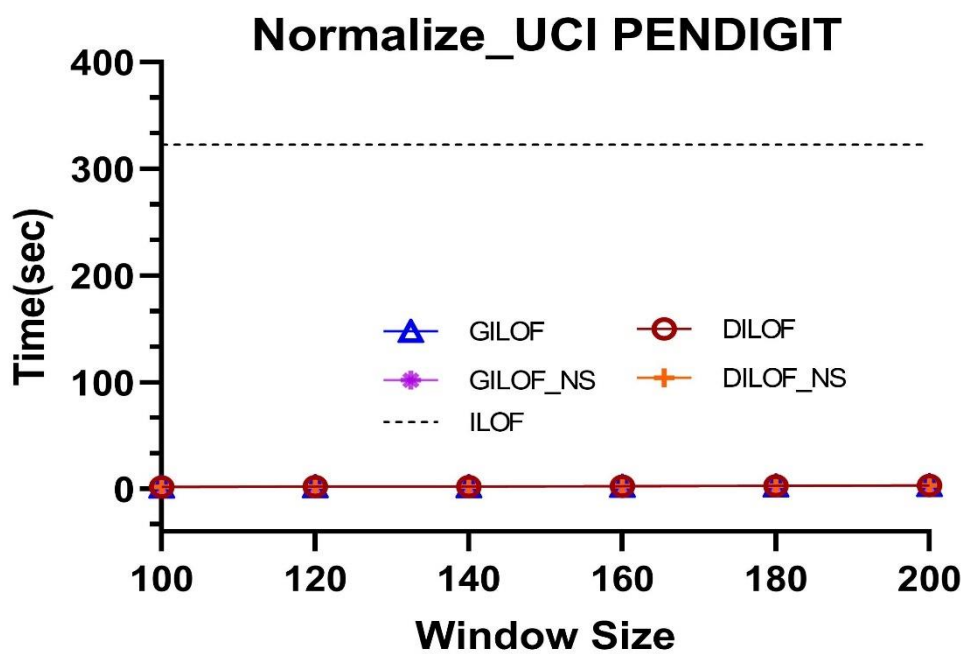


Figure 4.11. The comparisons of outlier detection execution time for normalized UCI Pendigit datasets

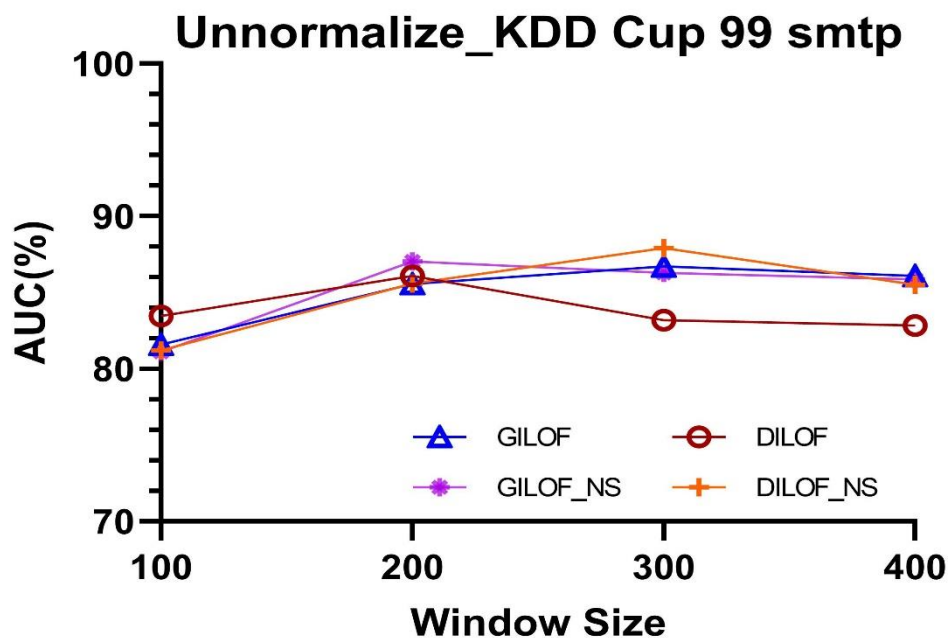


Figure 4.12. The comparisons of outlier detection accuracy for unnormalized UCI KDD Cup99 SMTP datasets.

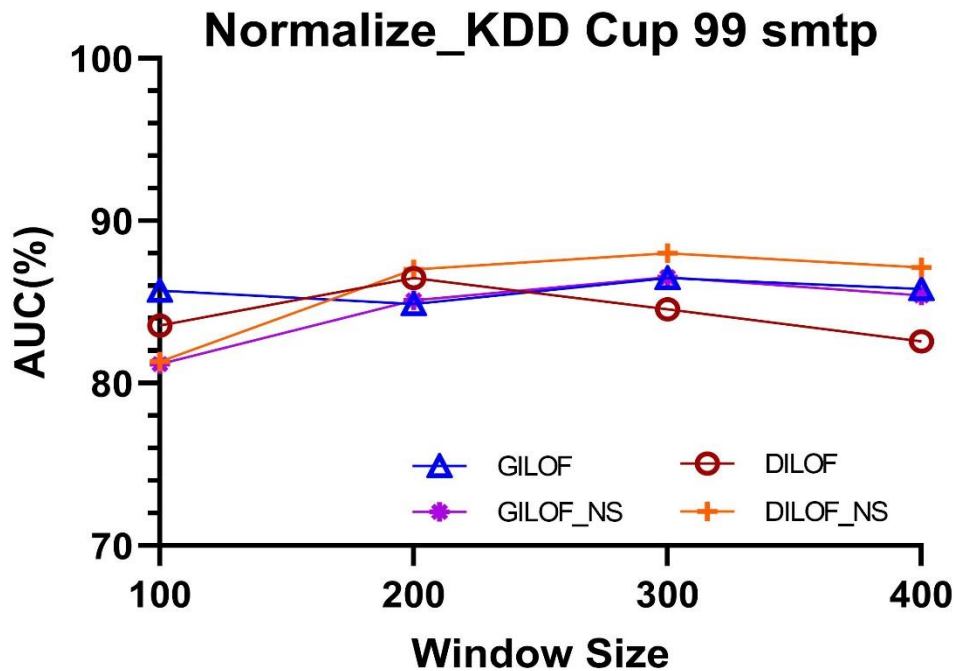


Figure 4.13. The comparisons of outlier detection accuracy for normalized UCI KDD Cup99 SMTP datasets.

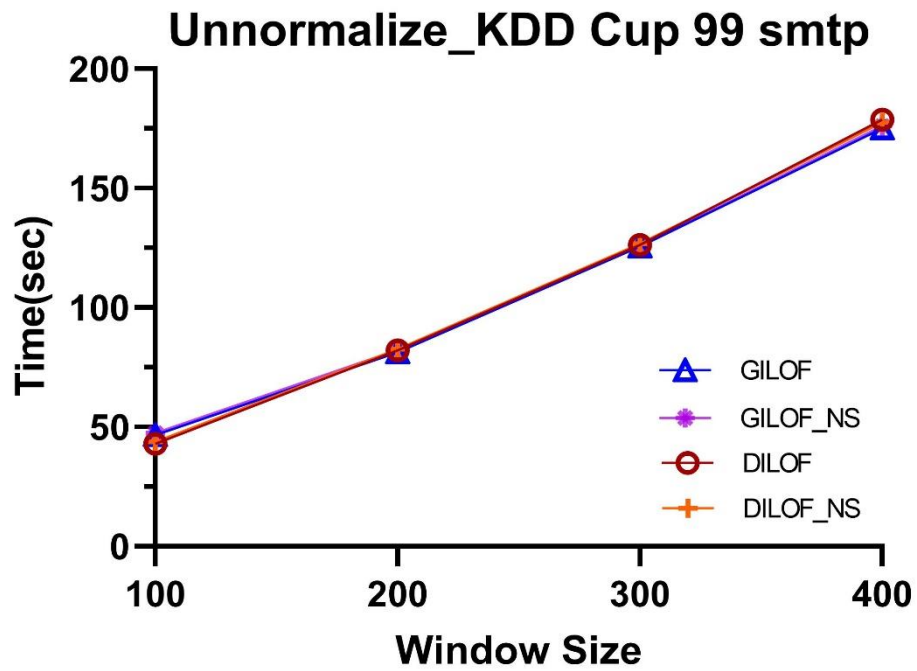


Figure 4.14. The comparisons of outlier detection execution time for unnormalized KDD Cup SMTP 99 dataset.

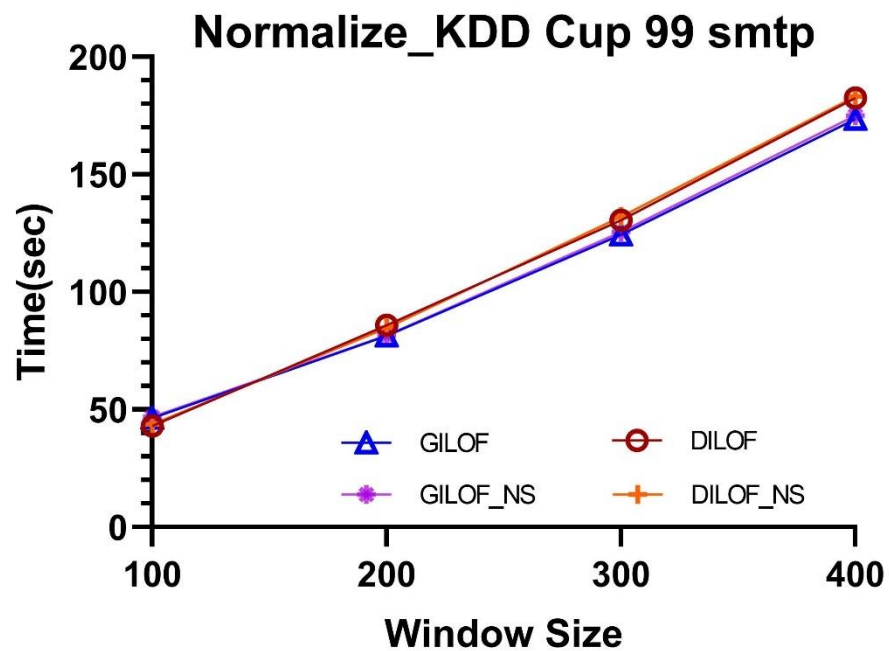


Figure 4.15. The comparisons of outlier detection execution time for normalized KDD Cup 99 SMTP dataset.

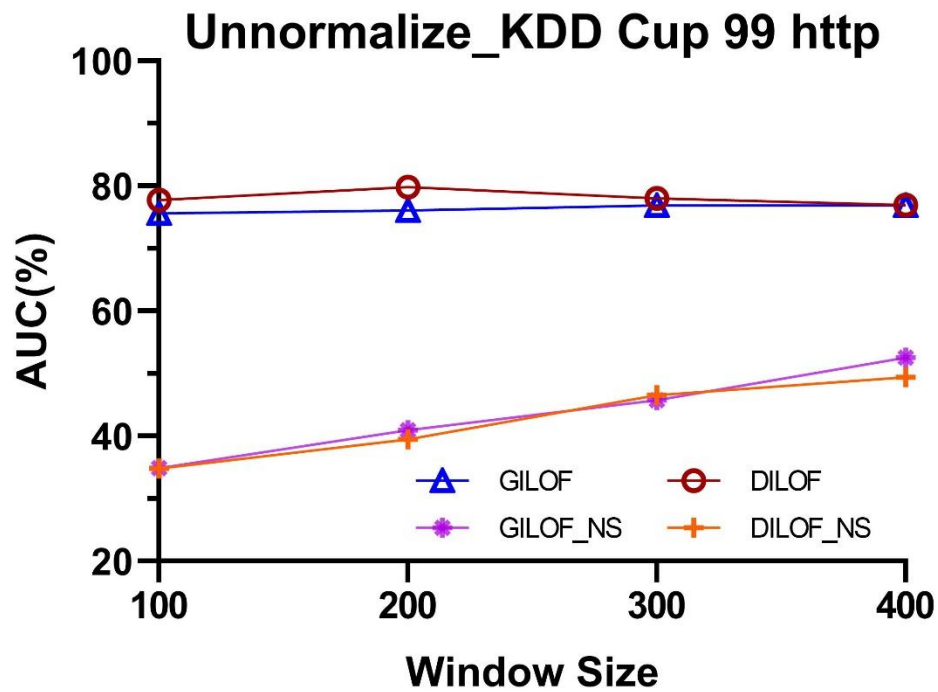


Figure 4.16. The comparisons of outlier detection accuracy for unnormalized UCI KDD Cup99 HTTP datasets.

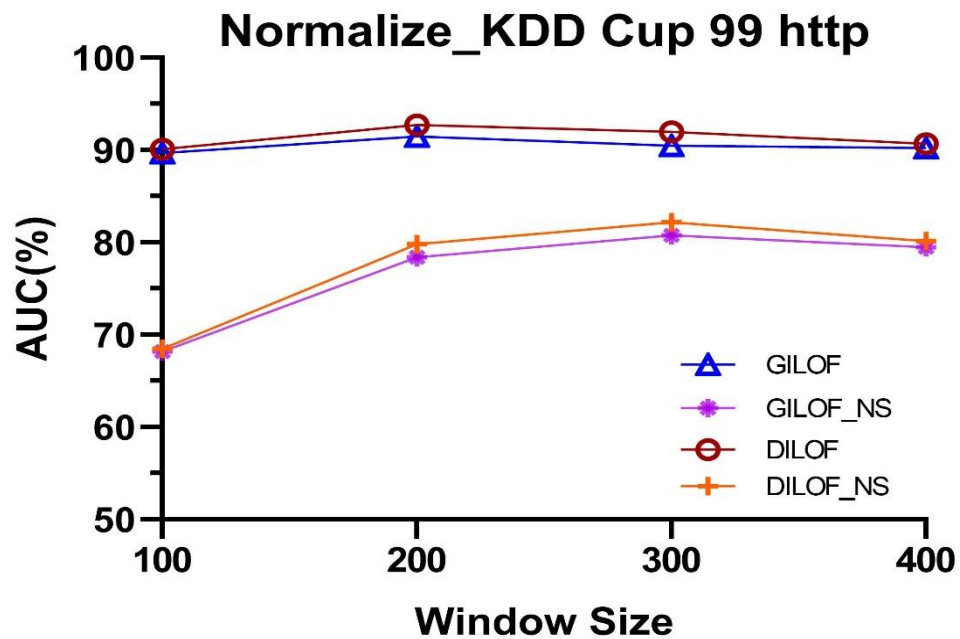


Figure 4.17. The comparisons of outlier detection accuracy for normalized UCI KDD Cup99 HTTP datasets.

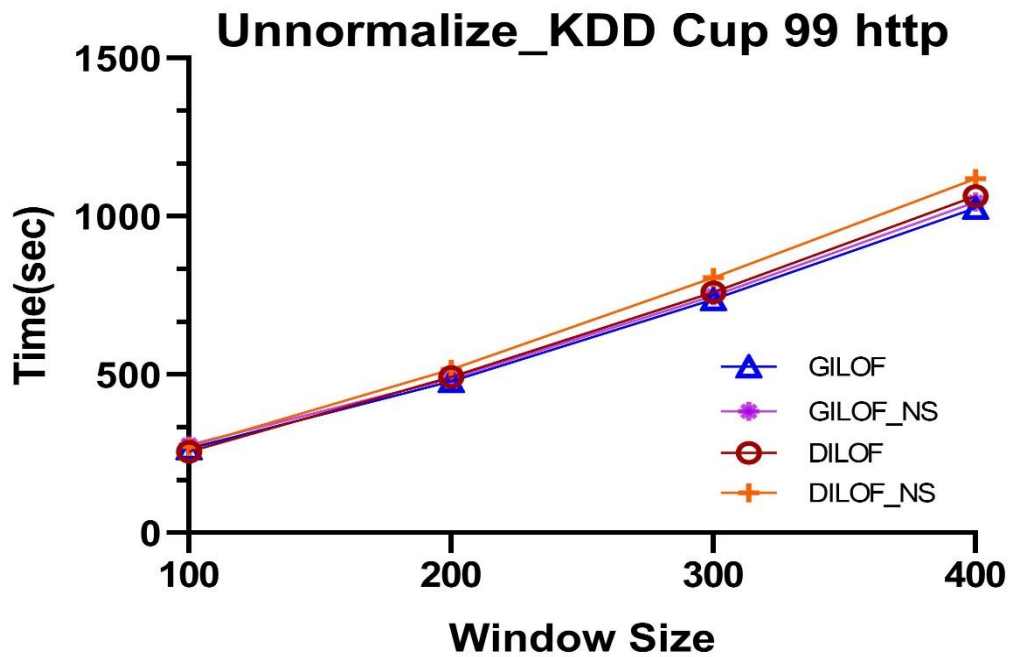


Figure 4.18. The comparisons of outlier detection execution time for unnormalized KDD Cup 99 HTTP dataset.

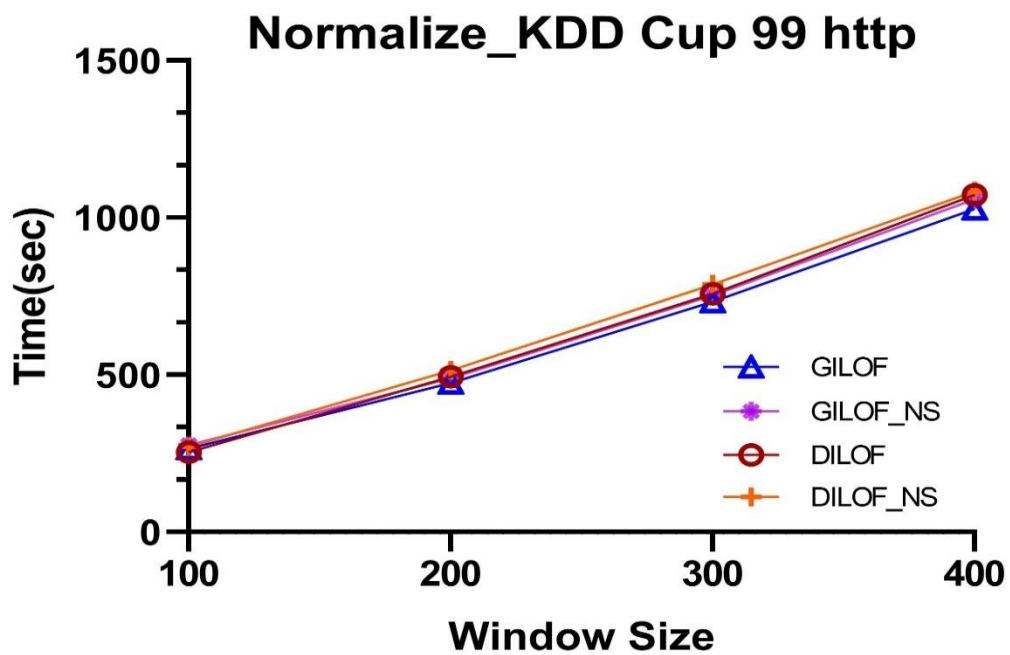


Figure 4.19. The comparisons of outlier detection execution time for normalized KDD Cup 99 HTTP dataset.

Table 4.1. The performance results of all algorithms for unnormalized UCI Vowel dataset.

<b>W Size</b>	Unnormalize UCI Vowel Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	74.21	0.721	74.41	0.787	69.21	0.720	72.09	0.786
<b>W120</b>	82.96	0.82	82.23	0.887	75.52	0.822	81.16	0.903
<b>W140</b>	86.23	0.918	86.84	0.993	83.95	0.922	85.07	1.007
<b>W160</b>	87.06	1.039	87.57	1.12	85.51	1.041	86.47	1.122
<b>W180</b>	89.47	1.157	90.23	1.24	88.46	1.156	89.27	1.26
<b>W200</b>	89.58	1.263	91.82	1.358	88.74	1.275	91.11	1.406
<b>W1000</b>	88.55	11.24	88.36	11.85	87.85	11.21	87.81	11.86

Table 4.2. The performance results of all algorithms for normalized UCI Vowel dataset.

<b>W Size</b>	Normalize UCI Vowel Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	72.54	0.729	73.7	0.792	69.51	0.713	70.66	0.804
<b>W120</b>	81.26	0.829	82.64	0.907	78.72	0.823	81.95	0.912
<b>W140</b>	86.02	0.917	86.87	1.005	84.4	0.919	85.57	1.005
<b>W160</b>	87.73	1.045	87.53	1.134	85.88	1.021	86.97	1.111
<b>W180</b>	89.62	1.16	90.36	1.259	88.82	1.149	88.89	1.25
<b>W200</b>	90.65	1.25	92.48	1.379	89.54	1.271	91.16	1.4
<b>W1000</b>	90.489	11.2	91.05	11.85	90.15	11.36	90.54	11.9

Table 4.3. The performance results of all algorithms for unnormalized UCI Pendigit dataset.

<b>W Size</b>	Unnormalize UCI Pendigit Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	50.86	1.94	50.21	1.95	50.96	1.874	50.66	1.868
<b>W120</b>	49.75	2.224	50.06	2.23	51	2.102	51.78	2.168
<b>W140</b>	50.63	2.45	50.77	2.46	52.27	2.422	51.11	2.432
<b>W160</b>	50.81	2.81	51.27	2.81	52.33	2.809	52.44	2.763
<b>W180</b>	50.64	3.14	51.69	3.14	52.17	3.13	52.87	3.148
<b>W200</b>	52.86	3.46	51.97	3.45	52.85	3.485	52.62	3.453
<b>W1000</b>	53.52	44.39	54.04	43.94	52.81	44.73	53.92	44.14

Table 4.4. The performance results of all algorithms for normalized UCI Pendigit dataset.

W Size	Normalize UCI Pendigit Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	AUC%	TIME	AUC%	TIME	AUC%	TIME	AUC%	TIME
<b>W100</b>	95.7	1.955	95.56	1.95	94.59	1.899	94.53	1.904
<b>W120</b>	97.29	2.241	97.47	2.23	96.05	2.186	95.98	2.205
<b>W140</b>	98.17	2.474	98.15	2.47	96.66	2.492	96.8	2.491
<b>W160</b>	98.61	2.827	98.69	2.81	97.61	2.822	97.57	2.812
<b>W180</b>	98.89	3.144	98.87	3.13	97.98	3.145	97.74	3.135
<b>W200</b>	98.43	3.463	98.41	3.44	97.46	3.494	97.28	3.513
<b>W1000</b>	93.07	44.47	93.34	43.84	87.92	44.6	88.66	44.324

Table 4.5. The performance results of all algorithms for unnormalized KDD Cup 99 SMTP dataset.

W Size	Unnormalize KDD Cup 99 SMTP Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	AUC%	TIME	AUC%	TIME	AUC%	TIME	AUC%	TIME
<b>W100</b>	81.59	46.5	81.11	47.38	83.45	42.87	81.22	43.68
<b>W200</b>	85.57	81.21	87.04	82.2	86.08	82.03	85.6	82.51
<b>W300</b>	86.7	125.5	86.28	125.7	83.19	126.2	87.92	126.7
<b>W400</b>	86.09	175.	85.87	176	82.84	178.72	85.51	177.6
<b>W1000</b>	71.47	1351.4	68.04	1352.8	71.66	1367	69.11	1367.

Table 4.6. The performance results of all algorithms for normalized KDD Cup 99 SMTP dataset.

W Size	Normalize KDD Cup 99 SMTP Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	AUC%	TIME	AUC%	TIME	AUC%	TIME	AUC%	TIME
<b>W100</b>	85.7	46.45	81.18	46.96	83.57	43.05	81.35	43.7
<b>W200</b>	84.88	81.36	85.12	81.62	86.48	85.95	87.00	84.72
<b>W300</b>	86.48	124.4	86.53	125.5	84.57	130.6	88.02	132.1
<b>W400</b>	85.82	173.52	85.42	175.1	82.59	182.6	87.14	183.2
<b>W1000</b>	69.48	1351.1	68.2	1357.9	69.49	1369.8	68.83	1371.2

Table 4.7. The performance results of all algorithms for unnormalized KDD Cup 99 HTTP dataset.

W Size	Unnormalize KDD Cup 99 HTTP Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	AUC%	TIME	AUC%	TIME	AUC%	TIME	AUC%	TIME
<b>W100</b>	75.62	267.11	34.89	277.89	77.75	256.38	34.82	272.94
<b>W200</b>	76.08	479.25	40.94	487.96	79.83	492.95	39.51	516.72
<b>W300</b>	76.89	738.37	45.74	748.99	78.02	760.73	46.563	806.55
<b>W400</b>	76.91	1026.7	52.59	1046	76.98	1064.8	49.42	1118.8
<b>W1000</b>	72.08	8052.8	55.655	8140	73.02	8142.6	56.571	8278.2

Table 4.8. The performance results of all algorithms for normalized KDD Cup 99 HTTP dataset

W Size	Normalize KDD Cup 99 HTTP Dataset							
	GILOF		GILOF_NS		DILOF		DILOF_NS	
	AUC%	TIME	AUC%	TIME	AUC%	TIME	AUC%	TIME
<b>W100</b>	89.68	265.87	68.17	276.5	90.11	253.88	68.47	271.41
<b>W200</b>	91.48	473.84	78.39	489.89	92.72	494.62	79.81	514.27
<b>W300</b>	90.48	733.53	80.78	753.19	91.96	758.46	82.17	787.89
<b>W400</b>	90.24	1028.5	79.47	1059.3	90.69	1073	80.14	1084.1
<b>W1000</b>	85.12	7978.4	69.85	8112.9	84.61	8126.4	70.04	8271.7

## 4.6 Conclusion

The aim of the proposed GILOF algorithm is to further improve the accuracy and efficiency of the LOF algorithm in data stream processing and mining. LOF requires a large memory to store all the data points and the distances between data points. Moreover, LOF recalculates the whole dataset again for any new change. In this chapter, a new local outlier detection algorithm in data stream is discussed in detail.

Among the previous works, DILOF has the best performance so far. The GILOF algorithm developed in our work is based on the genetic algorithm, which addresses the limitation of DILOF. GILOF has been evaluated together with DILOF over several real-world datasets. The results showed



that GILOF is better than DILOF in accuracy of outlier detection and execution time for most of the datasets.

## Chapter 5: Improving the Algorithm of Genetic-based Incremental Local Outlier Factor for Network Intrusion Detection

"Improving the Efficiency of Genetic based Incremental Local Outlier Factor Algorithm for Network Intrusion Detection" Forthcoming in *Proceedings of the 4th International Conference on Applied Cognitive Computing*, 2020. Springer

### 5.1 Introduction

Outlier detection is an important process in big data, which has received a lot of attention in the area of machine learning. The reason is that it is important to detect suspicious items and unusual activities. Outlier detection has a significant impact on many applications such as detecting fraud transactions for credit cards and network intrusion detection [127].

Local outlier factor (LOF) [58] is the most common local outlier (anomaly) detection algorithm, it is a nearest-neighbor-based algorithm [28]. LOF has become popular because it can detect an outlier without any previous knowledge about the data distribution. Also, it can detect outliers in data that has heterogeneous densities [178, 179]. However, LOF is designed for static data, which means the size of data is fixed. LOF needs very large memory because it requires storage of the whole dataset with the distances value in memory. Therefore, LOF cannot be applied to a data stream, where the size of data is increasing continuously [1, 7, 180]. To overcome this issue, the Incremental Local Outlier Factor (ILOF) has been developed that can handle data streams [88]. Nevertheless, ILOF also needs to store all the data in the memory. Moreover, it cannot detect the sequence of outliers in the data stream such as unexpected surges of outliers in several streaming segments. However, it is important to detect the sequence of outliers in a data stream; if they are not detected, many problems may occur, such as network attacks and sensor malfunctions. For instance, the KDD Cup 99 http dataset has several outlier sequences. This dataset contains a simulation of normal data with attack traffic on an IP scale in computer networks for testing intrusion detection systems.

To solve the issues with ILOF, several algorithms have been proposed that are Genetic-based Incremental Local Outlier Factor (GILOF) [181], Memory Efficient Incremental Local Outlier Factor (MILOF) [96], Density summarizing Incremental Local Outlier Factor (DILOF) [104]. The GILOF

demonstrates an efficient performance in the state-of-the-art LOF algorithms in data streams. Our work aims to further improve the performance of the GILOF algorithm by proposing a new calculation method for LOF, called Local Outlier Factor by Reachability distance (LOFR). In LOF, the resulting score is based on the local reachability density while in LOFR the resulting score is based on the reachability distance. The newly proposed algorithm is Genetic-based Incremental Local Outlier Factor by Reachability distance (GILOFR). GILOFR includes two stages: 1) the detection stage; and, 2) the summarization stage. The detection stage detects the outliers and updates the old data information in the memory (window) and determines whether a new data point is an inlier or outlier. The summarization stage summarizes the old data in order to reduce the memory consumption and takes into account the old data density. Thus, GILOFR is able to work in limited memory. This new calculation method (LOFR) is the main contribution of this research paper. In this research, GILOFR was evaluated via a series of experiments by using real-world datasets. The results show that the performance of GILOFR is better than GILOF on several of the datasets.

The remainder of this chapter is as follows: The network intrusion detection system (NIDS) is introduced in Section 2; Section 3 describes the methodology; Section 4 presents the experiments; Section 5 presents and discusses the results of the experiments; and Section 6 presents the conclusion of chapter five.

## **5.2 Network Intrusion Detection System (NIDS)**

The basic purpose of using data mining techniques for intrusion detection in networks is to detect security violations in information systems. To detect an intrusion in networks, data mining processes and analyzes a massive amount of data [182]. Data mining and the knowledge extracted from big data has become more efficient with the new developments in machine learning [183]. Many new algorithms in data mining have been developed that can be applied to datasets to discover clusters, profiles, factors, relationships, predictions and patterns. Nowadays, data mining algorithms are widely used in different domains, such as business, marketing, laboratory research, weather forecasting, and

network intrusion detection [184]. Our work is related to the Network Intrusion Detection System (NIDS). In order to test the effectiveness of the proposed algorithm in NIDS, we used the KDD Cup 99 HTTP service dataset, which is a subset of the KDD 99 Cup dataset [184]. This HTTP dataset has surprising surges of outliers in several streaming segments [185].

The intrusion detection system is classified by two categories: misuse detection and outlier (anomaly) detection. In outlier detection, it aims to recognize the suspicious items and unusual activity of the network or host. The NIDS monitors and tests the activity in the network, which provides the necessary security for the network of a system [186]. For instance, it detects malicious activities like a denial-of-service attack (DoS) or a network traffic attack. The LOF algorithm has been used in different models and methods for NIDS [187, 188, 86, 189, 104]. Despite the researchers' interest in outlier detection, they focus more on global outlier detection in a data stream rather than the local outliers, which have been less studied [93]. Therefore, it is difficult to use this traditional LOF algorithm in data streams for NIDS. However, the ILOF algorithm addressed the limitation of the LOFs in data streams, and other algorithms, such as GILOF, addressed the limitation of time complexity in the ILOFs by summarizing big data streams. The aim of our proposed algorithm GILOFR is to achieve an efficient performance for outlier detection in data streams for NIDS.

### **5.3 The Methodology**

#### *5.3.1 Local Outlier Factor by Reachability Distance (LOFR)*

In this section, we introduce a new calculation method for LOF, which is named LOFR. In fact, LOFR is like LOF [58], except the LOFR does not use the local reachability density when calculating its score. The score of LOFR is based on the Reachability distance of the data point  $p$  and its nearest neighbors. Below are the key definitions of LOFR; note that the first three definitions are the same as for the LOF [58].

Definition 1: calculating the  $k$ -dist of the data point  $p$

The distance between two data points  $p$  and  $o$  in a Euclidean is calculated using Equation 1.

$$d(p, o) = \sqrt{\sum_{i=1}^n (p_i - o_i)^2} \quad (1)$$

Let  $D$  be a dataset. For the data point  $p$ , the  $k$ -dist( $p$ ) is the distance between ( $p$ ) and the farthest neighbor point  $o$  ( $o \in D$ ) with the following conditions:

$k$  is a positive integer, 1) with regard to at least  $k$  data points  $o' \in D \setminus \{p\}$  it maintains that  $d(p, o') \leq d(p, o)$ , and 2) with regard to at most  $k-1$  data points  $o' \in D \setminus \{p\}$  it maintains that  $d(p, o') < d(p, o)$  [58].

Definition 2: calculating the  $k$ -nearest neighbor of  $p$

In this situation, the  $k$ -nearest neighbors of  $p$  are each a data point  $q$  whose distance to  $p$  is equal or smaller than the  $k$ -dist( $p$ ). The  $k$ -nearest neighbor is described in Equation 2.

$$N_{k\text{-distance}(p)}(p) = \{ q \in D \setminus \{p\} | d(p, q) \leq k\text{-dist}(p) \} \quad (2)$$

Definition 3: Calculating the Reachability distance ( $Rd$ ) of  $p$  in relation to  $o$

The  $Rd$  between two data points such as  $p$  with regard to  $o$  is defined in Equation 3.

$$\text{reach-dist}_k(p, o) = \max \{ k\text{-dist}(o), d(p, o) \} \quad (3)$$

Some examples of  $Rd$  when  $k$ -nearest neighbor = 5 are represented in Figure 5.1. When the distance between two data points (e.g.,  $p_2$ ) and  $o$  is not larger than the  $k$ -dist( $o$ ), then the  $k$ -dist( $o$ ) is the  $Rd$ . On the other hand, when the distance between two data points (e.g.,  $p_3$ ) and  $o$  is larger than the  $k$ -dist( $o$ ), the actual distance between them is the  $Rd$ .

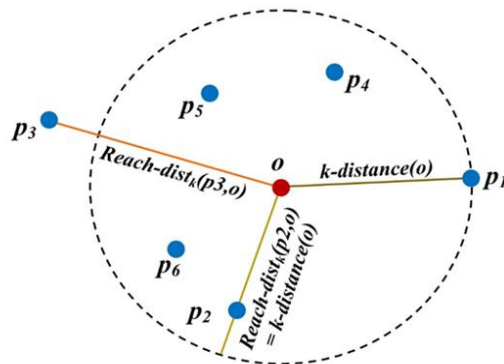


Figure 5.1. The reachability distance for different data points ( $p$ ) with regard to  $o$ , when  $k$  equals 5.

Definition 4: LOFR score of  $p$

After all previous steps, the score of LOFR is calculated by the following Equation 4.

$$LOFR_k(p) = \sum_{o \in N_k(p)} \frac{Rd_k(p)}{\left(\frac{Rd_k(o)}{k}\right)} \quad (4)$$

LOFR takes the Reachability distance ( $Rd$ ) of data point ( $p$ ) and divides it by the average  $Rd$  of its neighbors. This method of calculation can provide a little lower “outlierness” score than LOF.

### 5.3.2 Genetic-based Incremental Local Outlier Factor by Reachability Distance (GILOFR)

The main objective of GILOFR is to find the outlierness score in the following instances: 1) when detecting the local outlier with no prior knowledge about the data distribution; 2) when only a small part from the running dataset will be stored in memory; 3) when detecting that the outlier of data point ( $p$ ) has to be finished at current time  $T$ ; and, 4) the algorithm does not have any prior knowledge about future incoming data points when it detects the current outliers.

The designed GILOFR has two stages: detection and summarization. In the detection stage, the ILOFR and skipping scheme are used together to detect an outlier. Note: ILOFR is like ILOF [88], except the ILOFR uses the new calculation that is mentioned above. In the summarization stage, the Genetic Density Summarization (GDS) is applied to summarize the old data points in memory (the window). Algorithm 1 is the GILOFR, which works in the following steps: 1) it determines the window ( $W$ ) size, where the  $W$  size equals the amount of data points; 2) it uses the LOFR threshold  $\theta$  to detect outliers; and, 3) during the summarizing stage, GDS is applied to summarize the data points. Then, when the new data point  $np$  occurs, GILOFR detects the outliers by using ILOFR along with the skipping scheme (lines 4-6 in Algorithm 1) [181, 104]. GILOFR will keep detecting outliers and calculating the score of LOFR for every new data point until the present window reaches the  $W$  size. Thereafter, the GDS algorithm is applied on the window to summarize the 50% of older data points in the window through choosing 25% of data points from that older 50%. After that, the 50% of older data points will

be removed from the window and the chosen 25% will be transferred to the window; this 25% of data points will be combined with the remaining data points in the window (lines 9-13 in Algorithm 1). At this point, the window contains 75% of data points. This 75% of data points will be chosen to combine with the new streaming data points. This process is repeated when the window becomes full again (Figure 4.1).

---

Algorithm 1: GILOFR

---

Inputs: unlimited data streams  $P=\{p_1, p_2, \dots, p_t, \dots\}$ ,  
 The max window size  $W$   
 The threshold scores  $\theta$  of LOFR  
 The population size  $PS$   
 The number of chromosomes  $NC$   
 The number of generations  $NG$

- 1  $X \leftarrow \{\}$  // is the data points in the memory
- 2  $X' \leftarrow \{\}$  // is the oldest 50% data points in the memory
- 3  $O \leftarrow \{\}$  // is the detected outliers
- 4 Skipping\_Scheme
- 5 For each  $p_t \in P$  do
- 6      $LOFR_k(p_t) \leftarrow ILOFR(p_t, o, O, \theta)$
- 7     If  $LOFR_k(p_t) > 0$  then
- 8          $X \leftarrow X \cup \{p_t\}$
- 9         If  $|X| = W$  then
- 10              $Z \leftarrow GDS(X', PS, NC, NG)$
- 11             Delete the oldest 50%  $W$  data points in  $X$
- 12              $X' \leftarrow X' \cup Z$
- 13         End
- 14     End
- 15 End

---

### 5.3.2.1 Genetic Density Summarization (GDS)

GDS depends on the Genetic Algorithm (GA). A GA is a search algorithm in the evolutionary computation field, which is based on biological evolution [155, 163]. GA includes several components, Population, Chromosome (or individual), Objective (or fitness) function (Of), Selection, Crossover, and Mutation [157, 190]. The chromosomes are an array of numerical or binary values that represent the genes evolving and are known as candidate solutions. A population contains a set of chromosomes, which are set randomly as the initial population. The objective function aims to calculate the fitness

value for every chromosome in the population. The goal of selection is to find two chromosomes that acquired above-average fitness values. There are many types of selection, which include roulette wheel selection (RWS), linear rank-based selection (RNK), tournament selection (TNT), stochastic universal sampling selection (SUS), and linear rank-based selection with selective pressure (RSP) [159]. Crossover combines the characteristics of the two chromosomes based on a cut-off point to produce new chromosomes. Crossover has several types, as well, such as one-point, two-point, and uniform crossover [164]. Mutation aims to preserve the population diversity and prevent the population from becoming stuck in the local minima. There are also several types of mutation, such as single-point mutation, uniform mutation, and boundary mutation (BDM) [166].

To find the optimal data points, the GDS algorithm is going to summarize 50% of old data points in the window to balance the density differences between the chosen old 50% and the new candidates, which are 25% of data points. The GDS algorithm begins by producing a population, where the population contains random chromosomes assigned with data points. Then, the Objective Function ( $Of$ ) is executed to evaluate each chromosome. The  $Of$  is the sum of  $a_k(x'_n)$ .

$$Of = \sum_{n=1}^{50\%W} a_k(x'_n) \quad (5)$$

Where  $a_k(x'_n)$  is the density of  $x'_n$  according to its  $k^{th}$  nearest neighbors.  $x'_n$  is the 50% of old data points in the window.

After that, in every generation, GDS executes the selection operation on the present generation. Then GDS executes a crossover operation on two chromosomes. Thereafter, the GDS executes the mutation operation on two chromosomes as well. When GA operations are completed for each generation, GDS stores the best results. After the above steps, the chosen chromosomes are converted into  $S$ . After that,  $S$  is projected into the binary domain, and the better 25% of  $s_n$  in  $S$  is adjusted to 1, and the rest to 0. The GDS is going to choose a data point  $x'_n$  in  $X'$  if its corresponding  $s_n$  in  $S$  equals 1 (Figure 4.2). Finally, the chosen data points will be presented as  $Z$ , which is 25% of the new data points. For more detail, refer to [181].



### 5.3.2.2 Skipping Scheme

The purpose of the Skipping scheme is to keep data points in the dataset in a lower density of the region. Therefore, when the new data points emerge into a new class, that data point will have no effect on the outlier detection accuracy. Because of the benefit of the skipping scheme in outlier detection, it combines with ILOFR to detect any long sequence (series) of outliers [104].

Let  $a \in A$ , the skipping scheme works by using Euclidean distance in two steps: 1) the average distance  $\overline{d_1(A)}$  is computed according to the distance between the data point  $a$  with its first nearest neighbor  $d_1(a)$ ; and, 2) the distance between a new data point ( $p$ ) and the latest detected outlier  $lo$  is calculated. If the distance between  $p$  to  $lo$  is less than the distance between  $p$  to  $\overline{d_1(A)}$ , the data point  $p$  is considered an outlier [104]. Otherwise, the data point  $p$  will be an inlier.

## 5.4 Empirical Experiment

In this section, the experiment results for GILOFR are presented. We compared the GILOFR results with the results of GILOF and we compared both algorithms without the skipping scheme (GILOFR\_NS, GILOF\_NS). The metrics that are used in the experiment are the execution time and the accuracy of outlier detection. For the accuracy of outlier detection, we used the Area Under the ROC Curve (AUC) [171, 172]. All algorithms in the experiment were implemented in C++ on a machine that has an Intel(R) Core(MT) i5-8250U CPU, 250 GB SSD hard disk, 8GB RAM, and Windows 10 (64-bit).

The GILOFR is tested with different window sizes on unnormalized and normalized datasets, described in Table 2.1. Those datasets are available in the Machine Learning database repository [146] and [147]. The Vowel dataset is modified to the format of the data stream, such as that in [181,173]. The five percent of uniform noise is added to the Pendigit dataset, so any noised data point will be considered as an outlier [104]. For the KDD Cup99 datasets, the same settings as in [174] are used, where the network attacks are the outliers.

The hyperparameters of GILOF were set as in [181], the population size ( $PS$ ) is 2, number of generations ( $NG$ ) is 4, and the selection type is  $RWS$ . The crossover used a *two-point* crossover at a rate of 0.7, and mutation used a  $BDM$  mutation at a rate of 0.07 for each dataset [175]. For the GILOFR algorithm, we used the same hyperparameters of GILOF, except the number of generations ( $NG$ ) was changed to 2. The  $k$ -nearest neighbors for each dataset were determined as the following for all algorithms: for the Vowel dataset  $k = 19$ ; Pendigit dataset  $k = 18$ ; KDD Cup 99 SMTP dataset  $k = 8$  when the dataset is normalized and  $k = 9$  when the dataset is unnormalized for GILOFR and GILOFR\_NS while  $k = 8$  for GILOF and GILOF\_NS; for the KDD Cup 99 HTTP dataset  $k = 9$  when the dataset is normalized and  $k = 8$  when the dataset is unnormalized in all algorithms. In addition, we determined 10 window sizes for all datasets,  $W = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$ .

## 5.5 Experiment Results

### 5.5.1 Accuracy of Outlier Detection

The accuracy of all algorithms was tested by applying them to several real-world datasets (Table 2.1). Figures 5.2 to 5.9 show the AUC of all algorithms for the four datasets. In the unnormalized Vowel dataset, all algorithms were close to each other, but GILOFR\_NS was better in most  $W$  sizes and demonstrated the highest accuracy result (95.3%). In the normalized dataset, GILOFR\_NS was also a little better in most  $W$  sizes and demonstrated the highest accuracy result (96.2%). Tables 5.1 and 5.2.

In the unnormalized Pendigit dataset, all algorithms showed poor results for accuracy. By contrast, for the normalized dataset all algorithms were close to each other when  $W \leq 400$ , but GILOF and GILOF\_NS were better when  $W \geq 500$ , and they demonstrated the highest accuracy result (98.7%). Tables 5.3 and 5.4.

In the unnormalized KDD Cup 99 SMTP dataset, all algorithms were in competition when  $W \leq 500$ ; therefore, there is a specific algorithm that surpasses others in each  $W$  size. GILOFR and GILOFR\_NS showed better results and GILOFR\_NS demonstrated the highest accuracy result (88%) in  $W = 300$ . In the normalized KDD Cup99 SMTP dataset, all algorithms were in competition. Every

algorithm showed better accuracy in specific  $W$  sizes, while GILOFR\_NS demonstrated the highest accuracy result (89%) in  $W = 200$ . Tables 5.5 and 5.6.

In the unnormalized KDD Cup 99 HTTP dataset, GILOFR showed better results for accuracy in all  $W$  sizes, except when  $W = 700$ . However, GILOFR demonstrated the highest accuracy result (79.3%). Moreover, GILOFR\_NS and GILOF\_NS showed poor results for accuracy in most  $W$  sizes, while GILOFR\_NS demonstrated a moderate result for accuracy in  $W = 800$  (65.3%). For the normalized KDD Cup99 HTTP dataset, GILOFR showed better results for accuracy in all  $W$  sizes except when  $W = 900$ . However, GILOFR demonstrated the highest results for accuracy (93.6%). GILOFR\_NS and GILOF\_NS became better when the dataset was normalized, in which case GILOFR\_NS surpasses GILOF\_NS in all  $W$  sizes. Tables 5.7 and 5.8.

Finally, the experiment results prove that our new method of calculation can improve the GILOF algorithm, where GILOFR and GILOFR\_NS show better results for accuracy than GILOF and GILOF\_NS in several datasets, especially in the UCI Vowel and the KDD Cup99 HTTP datasets.

### 5.5.2 Execution time

In Figures 5.10 to 5.17, the results for the execution time are presented for all algorithms in the unnormalized and normalized datasets, respectively. In the experiments, execution times are measured by seconds. Generally, all algorithms are close to each other in most  $W$  sizes. In the UCI Vowel dataset, GILOFR took 0.73 to 12.3 seconds and GILOF took 0.75 to 12.4 seconds. In the UCI Pendigit dataset, GILOFR took 1.96 to 47.19 seconds and GILOF took 2.05 to 47.12 seconds. In KDD Cup99 SMTP dataset, GILOFR took 45.09 to 1442.1 seconds and GILOF took 46.2 to 1420.4 seconds. In KDD Cup99 HTTP dataset, GILOFR took 269.7 to 8436.5 seconds and GILOF took 274.5 to 8356.6 seconds. Moreover, GILOFR\_NS and GILOF\_NS were very close to GILOFR and GILOF in execution time for most  $W$  sizes. Tables 5.1 to 5.8.

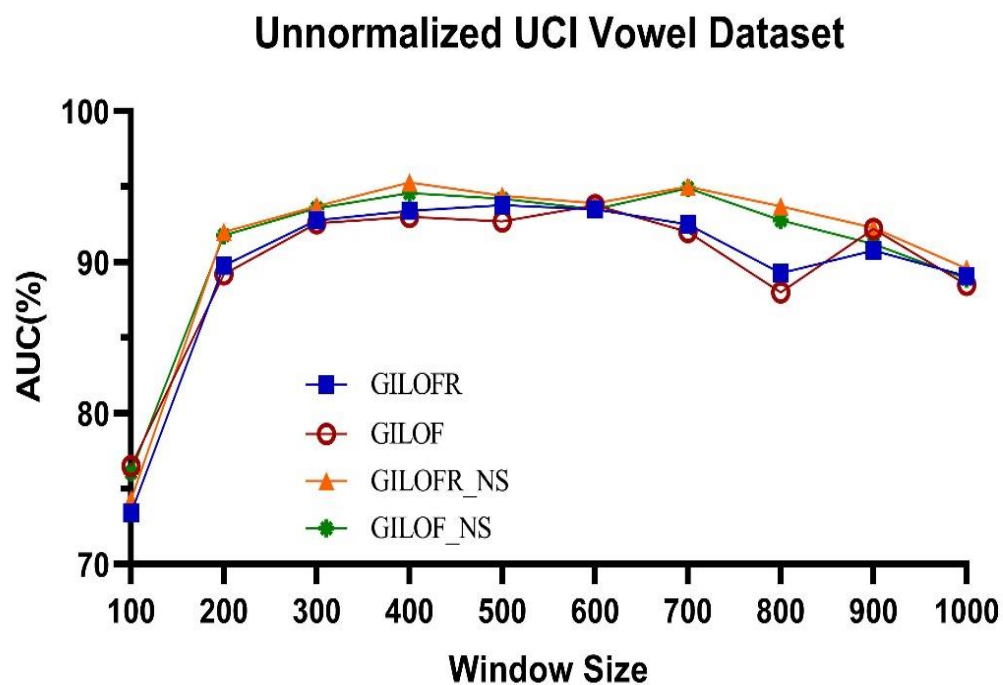


Figure 5.2. An accuracy comparison of outlier detection for unnormalized UCI Vowel dataset.

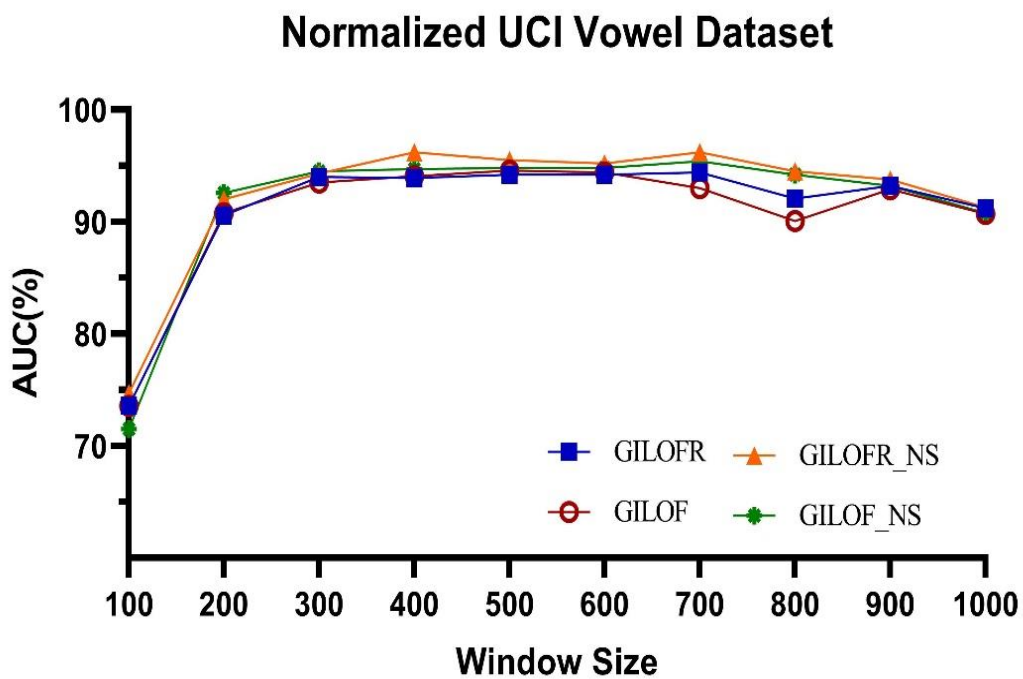


Figure 5.3. An accuracy comparison of outlier detection for normalized UCI Vowel dataset.

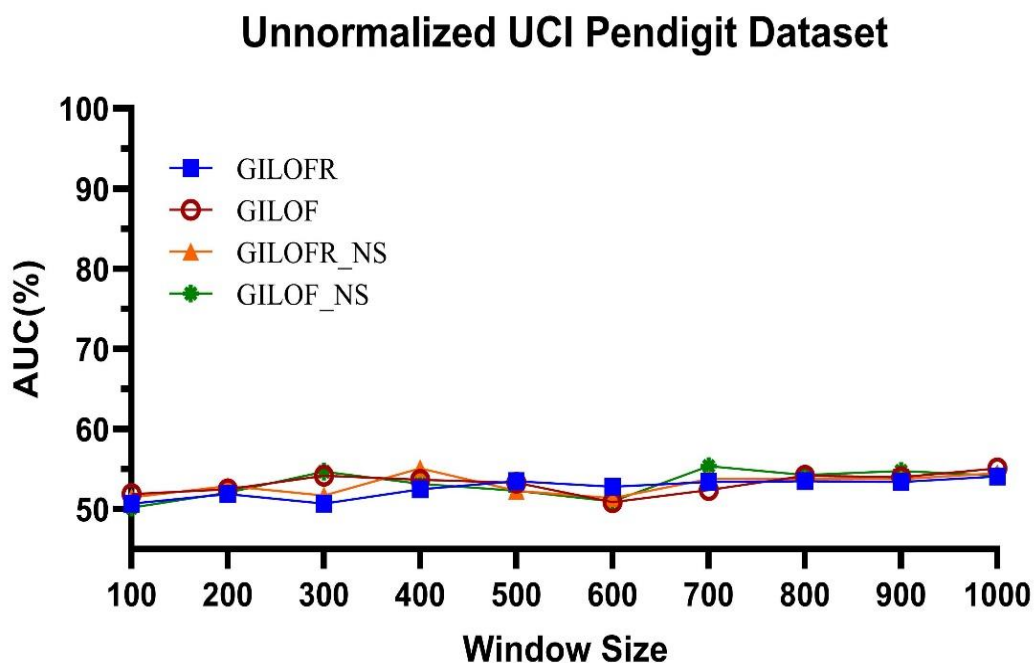


Figure 5.4. An accuracy comparison of outlier detection for unnormalized UCI Pendigit dataset.

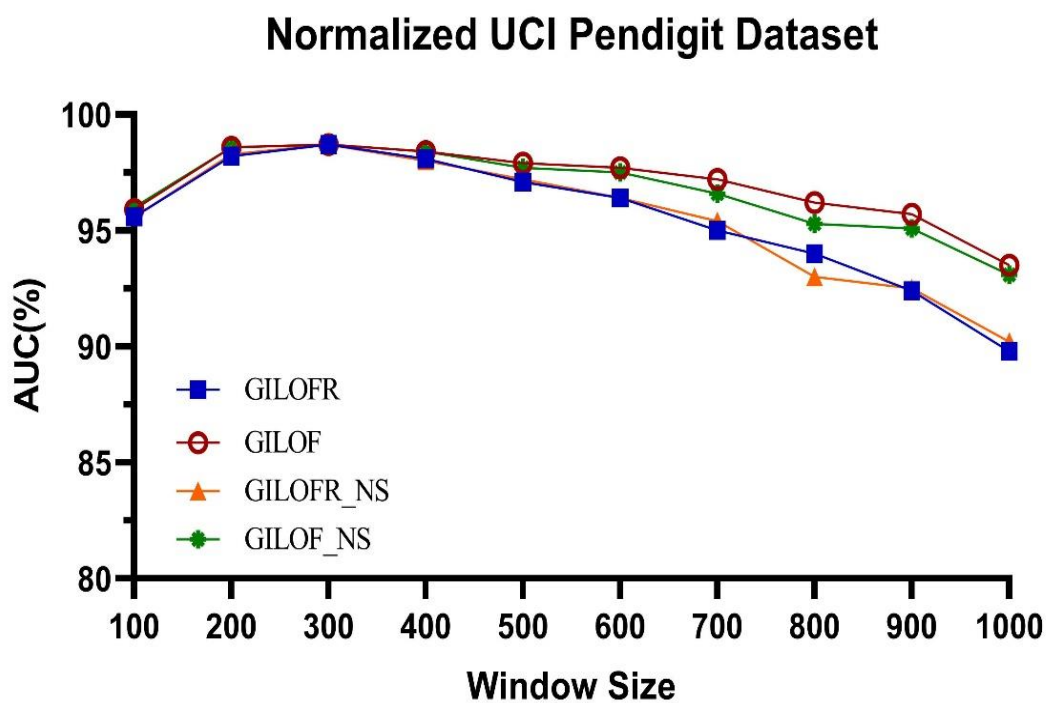


Figure 5.5. An accuracy comparison of outlier detection for normalized UCI Pendigit dataset.

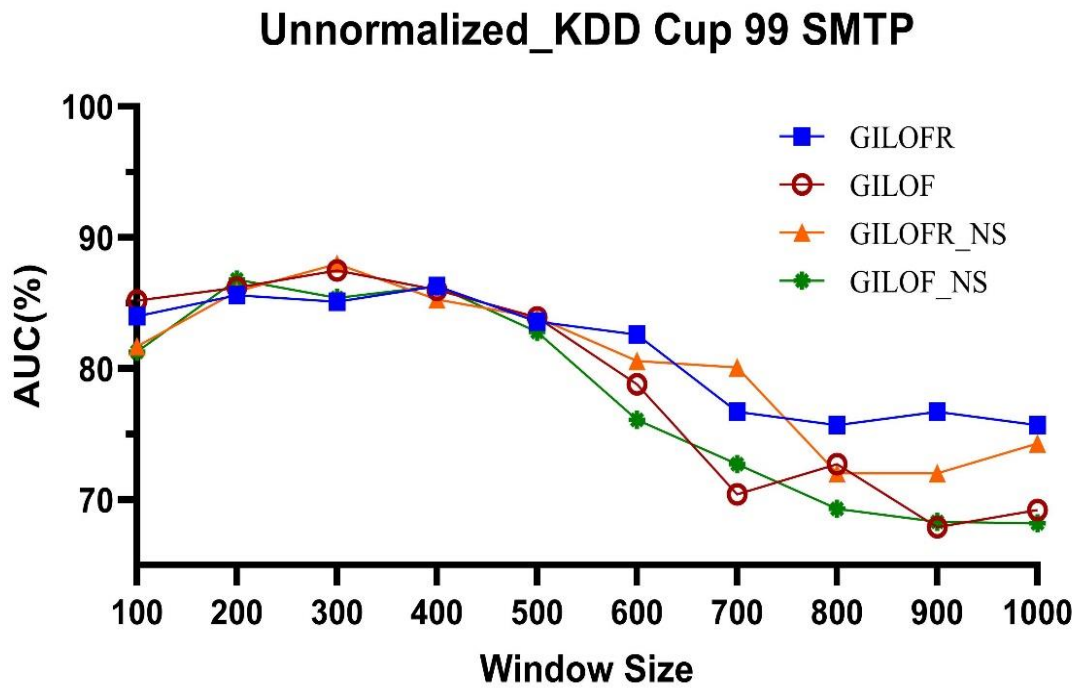


Figure 5.6. An accuracy comparison of outlier detection for unnormalized KDD Cup 99 SMTP dataset.

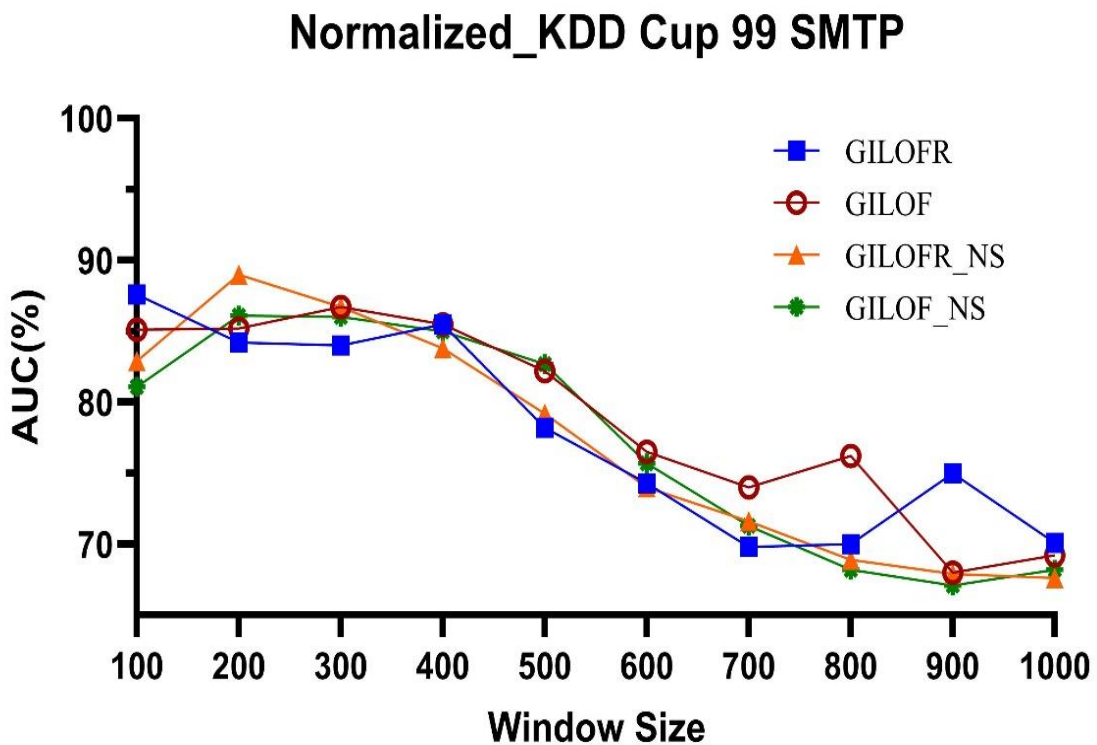


Figure 5.7. An accuracy comparison of outlier detection for normalized KDD Cup 99 SMTP dataset.

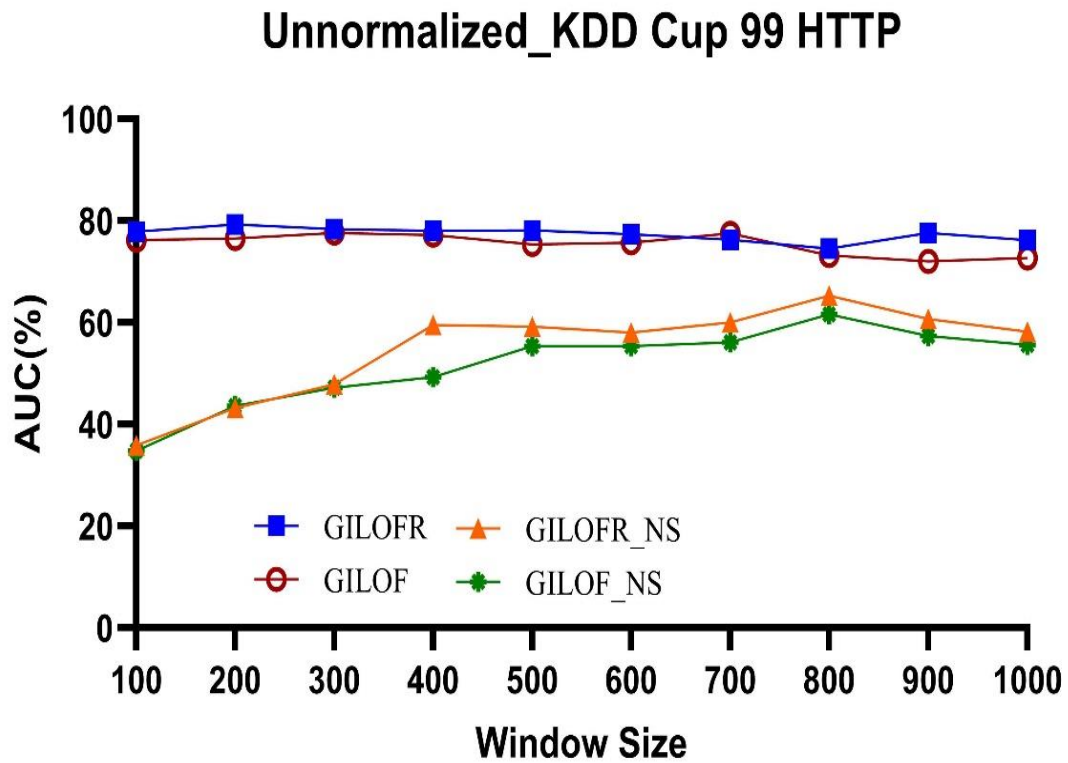


Figure 5.8. An accuracy comparison of outlier detection for unnormalized KDD Cup 99 HTTP dataset.

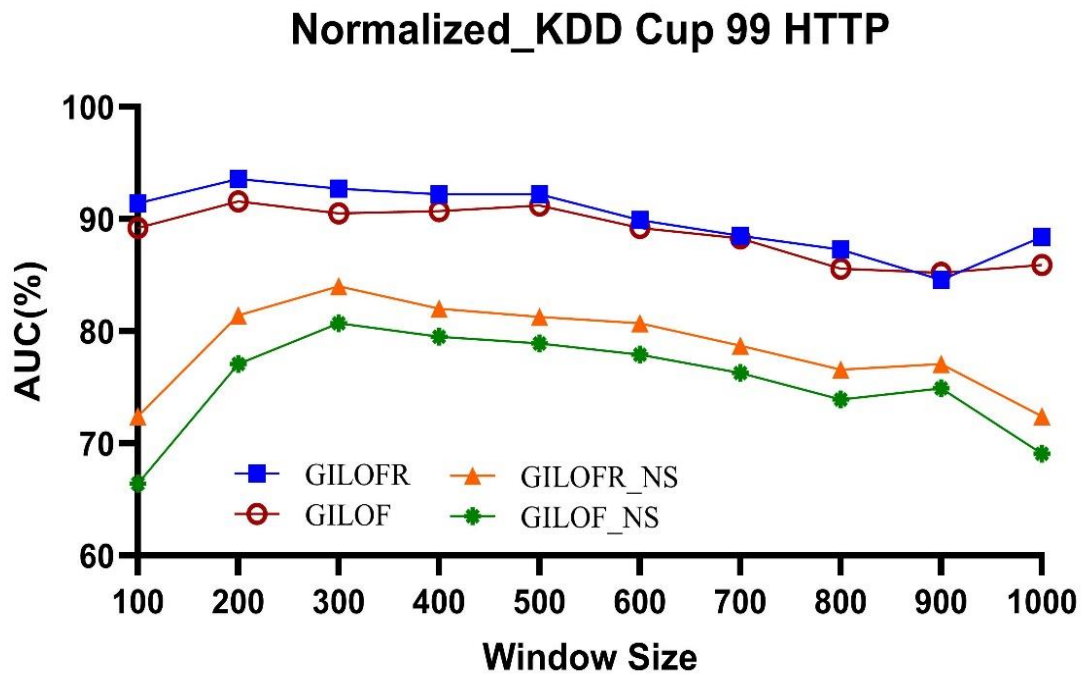


Figure 5.9. An accuracy comparison of outlier detection for normalized KDD Cup 99 HTTP dataset.

### Unnormalized UCI Vowel Dataset

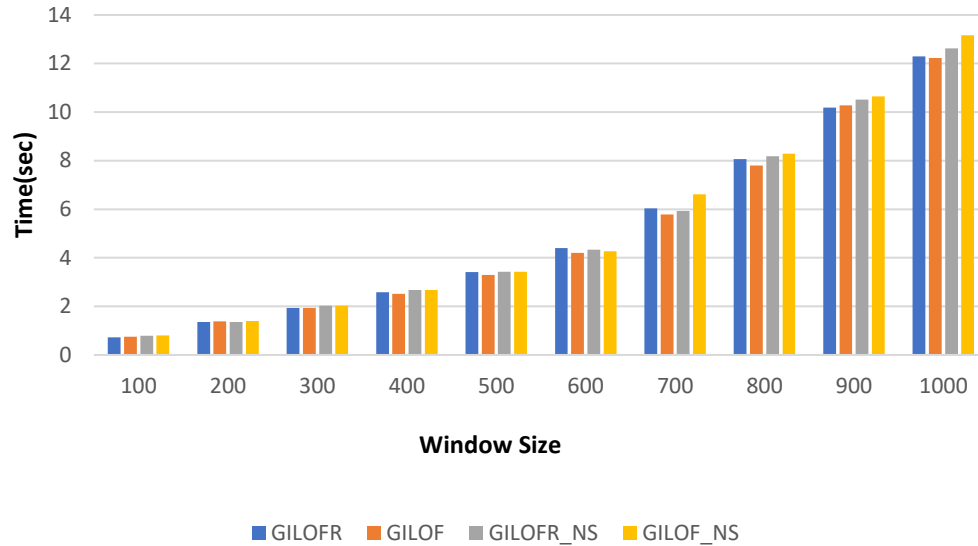


Figure 5.10. The execution time comparison of outlier detection for unnormalized UCI Vowel dataset.

### Normalized UCI Vowel Dataset

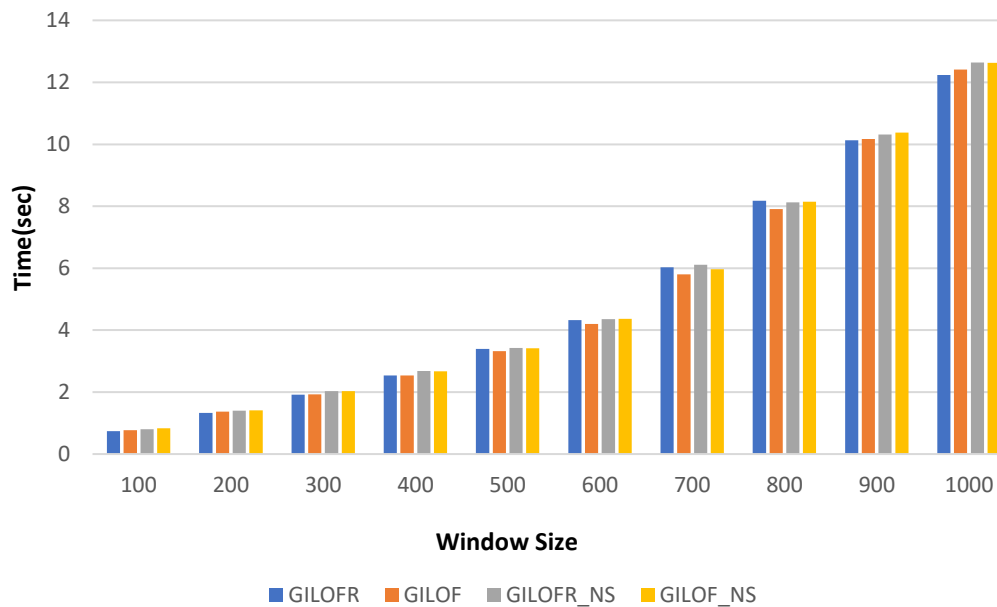


Figure 5.11. The execution time comparison of outlier detection for normalized UCI Vowel dataset.



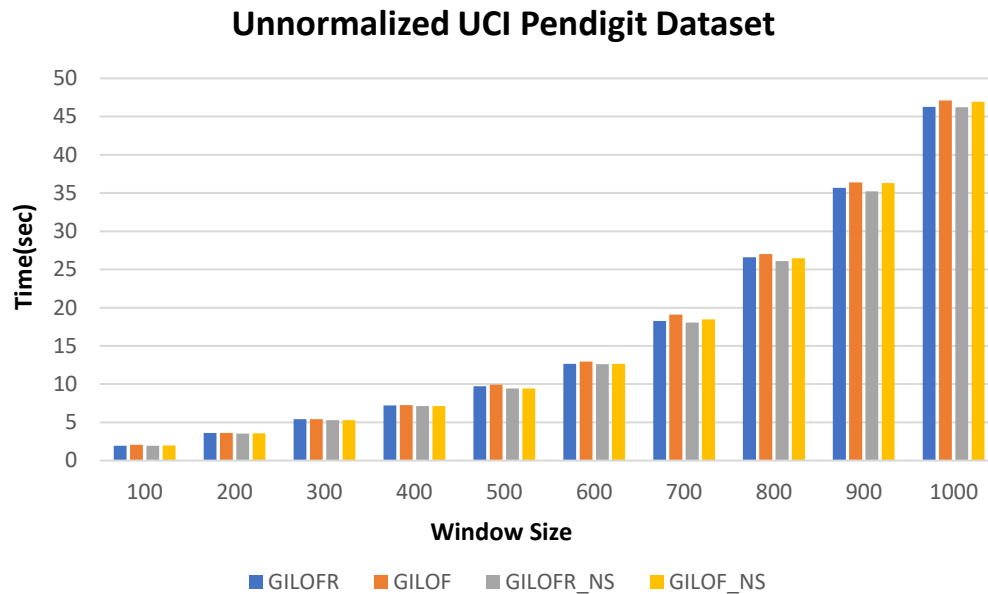


Figure 5.12. The execution time comparison of outlier detection for unnormalized UCI Pendigit dataset

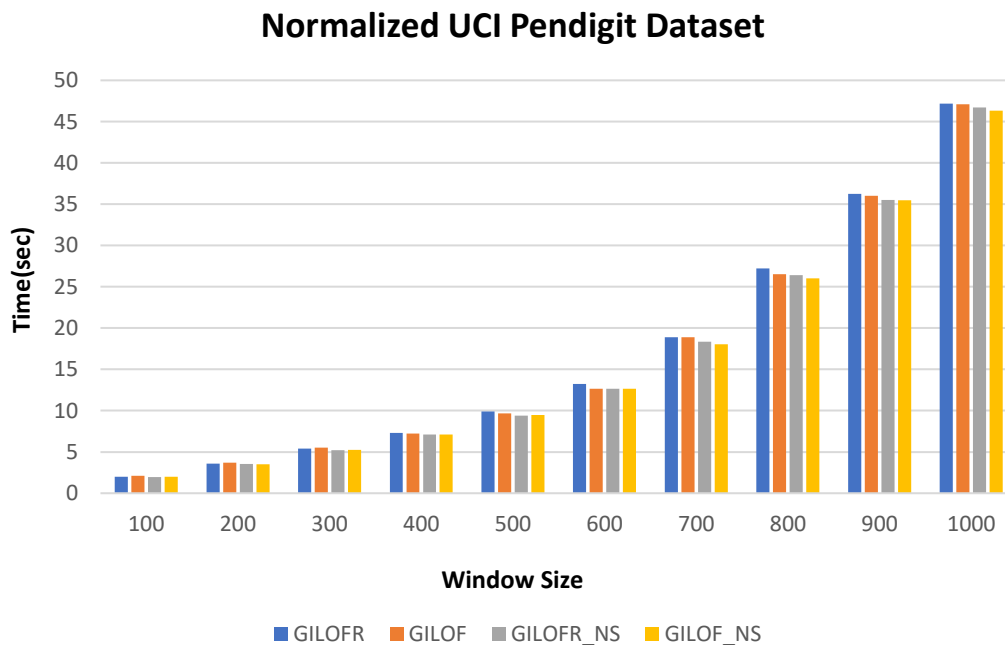


Figure 5.13. The execution time comparison of outlier detection for normalized UCI Pendigit dataset

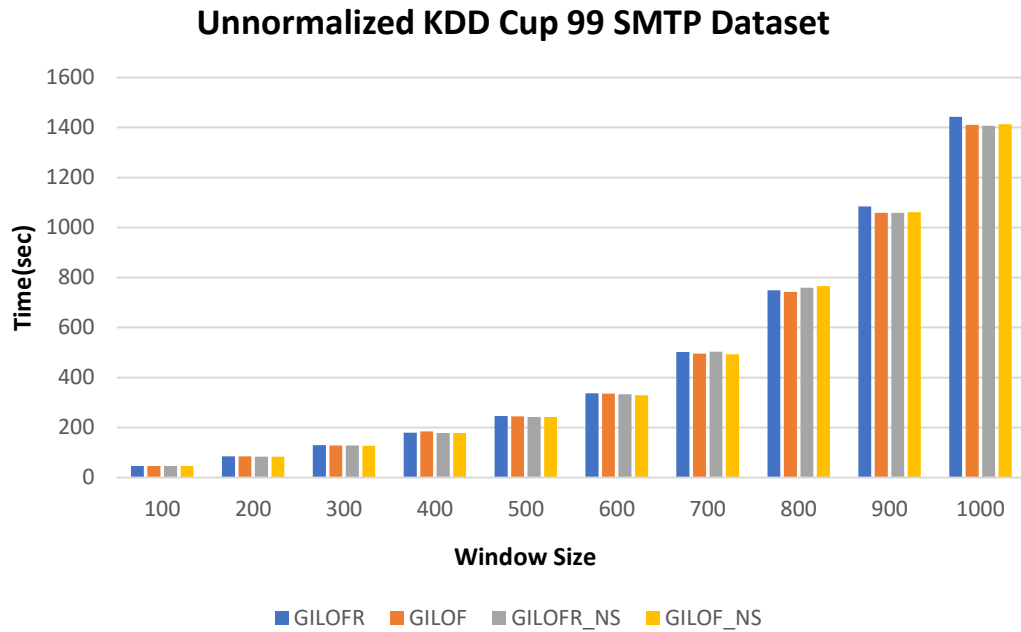


Figure 5.14. The execution time comparison of outlier detection for unnormalized KDD Cup99 SMTP dataset.

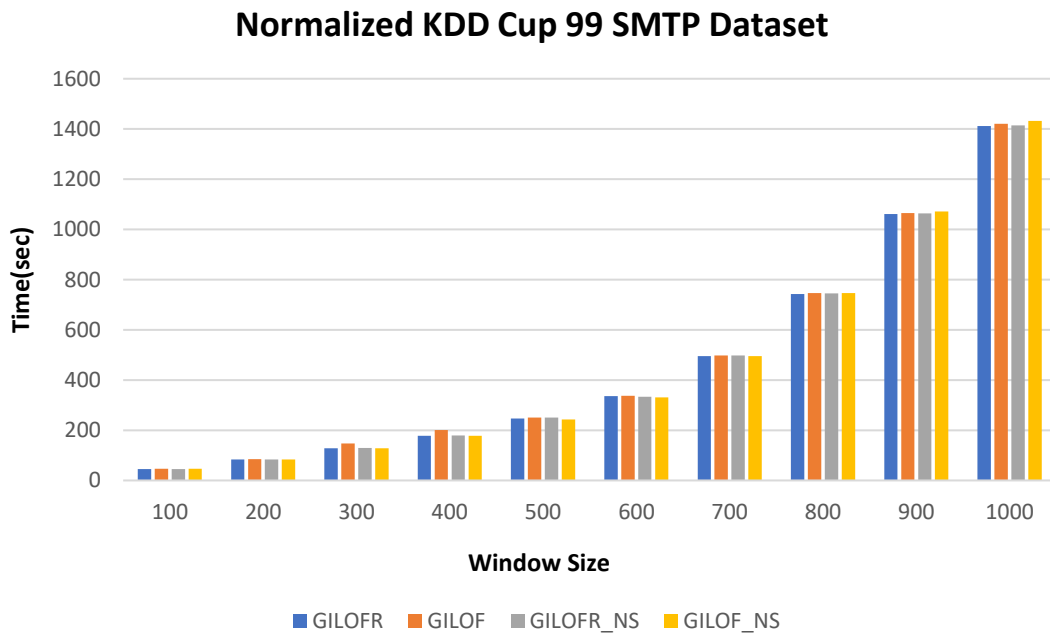


Figure 5.15. The execution time comparison of outlier detection for normalized KDD Cup99 SMTP dataset.

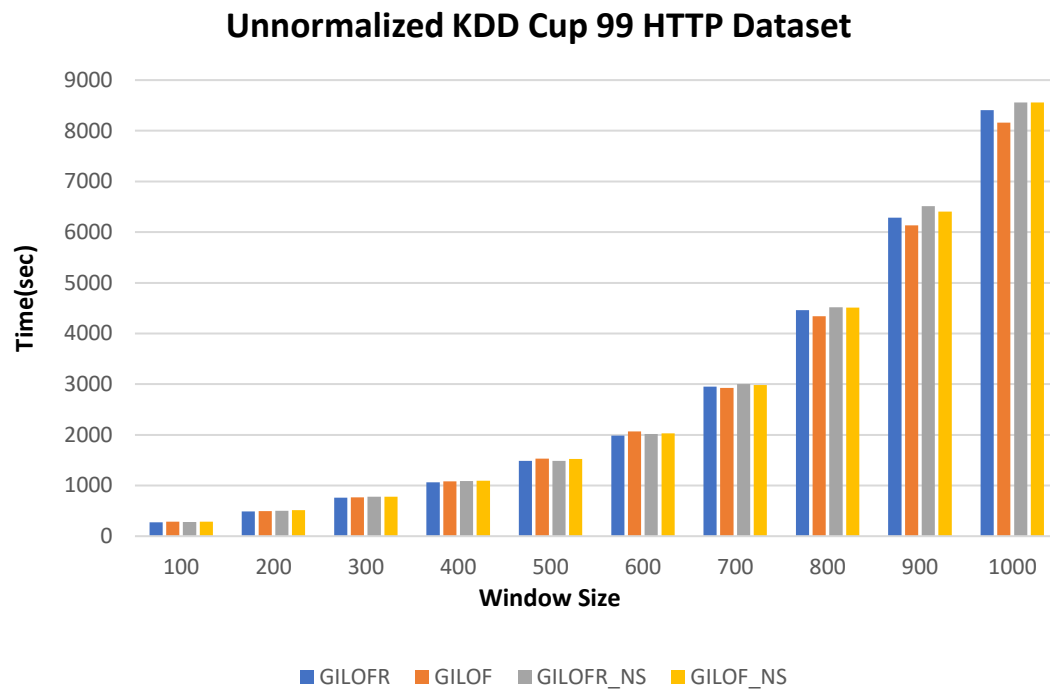


Figure 5.16. The execution time comparison of outlier detection for unnormalized KDD Cup99 HTTP dataset.

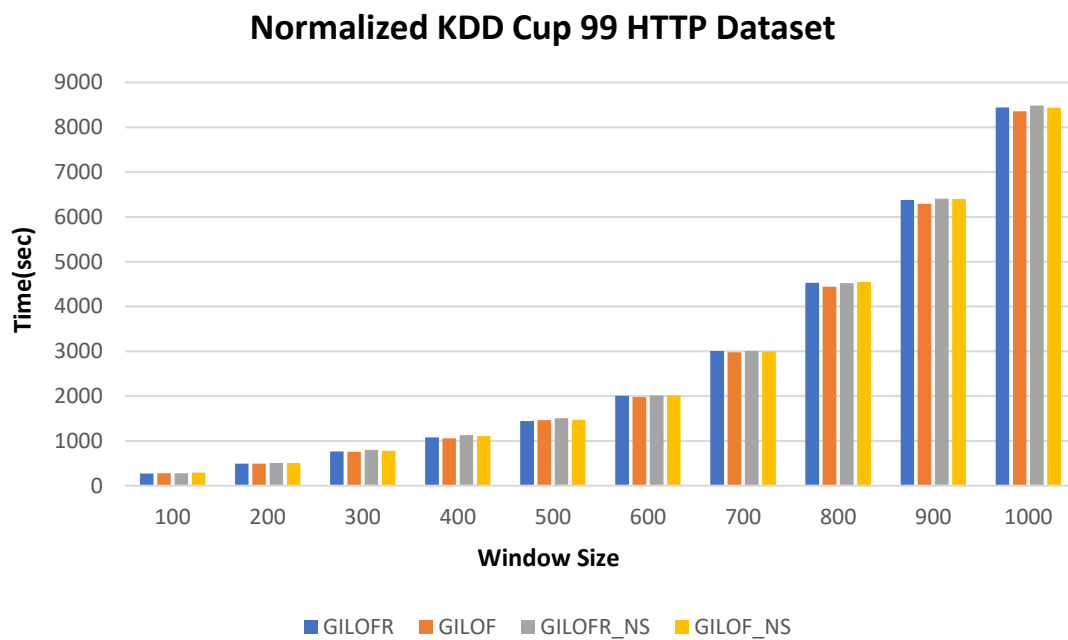


Figure 5.17. The execution time comparison of outlier detection for normalized KDD Cup99 HTTP dataset.

Table 5.1. The performance results of GILOFR and GILOF algorithms for unnormalized UCI Vowel dataset.

<i>W Size</i>	Unnormalize UCI Vowel Dataset							
	GILOFR		GILOFR_NS		GILOF		GILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	73.4	0.73	74.3	0.79	76.5	0.75	76.1	0.8
<b>W200</b>	89.8	1.36	92	1.36	89.2	1.38	91.8	1.4
<b>W300</b>	92.8	1.94	93.7	2.03	92.6	1.94	93.6	2.03
<b>W400</b>	93.4	2.58	95.3	2.68	93	2.52	94.6	2.67
<b>W500</b>	93.8	3.41	94.4	3.42	92.7	3.3	94.2	3.42
<b>W600</b>	93.5	4.4	93.9	4.33	93.8	4.21	93.5	4.27
<b>W700</b>	92.5	6.04	95	5.93	92	5.78	95.9	6.61
<b>W800</b>	89.3	8.07	93.7	8.19	88	7.8	92.8	8.29
<b>W900</b>	90.8	10.18	92.3	10.51	92.2	10.28	91.2	10.65
<b>W1000</b>	89.1	12.3	89.6	12.63	88.5	12.23	88.9	13.17

Table 5.2. The performance results of GILOFR and GILOF algorithms for normalized UCI Vowel dataset.

<i>W Size</i>	Normalize UCI Vowel Dataset							
	GILOFR		GILOFR_NS		GILOF		GILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	73.6	0.74	74.7	0.798	73.6	0.77	71.5	0.83
<b>W200</b>	90.6	1.33	92	1.4	90.8	1.37	92.6	1.41
<b>W300</b>	94	1.92	94.3	2.03	93.5	1.93	94.5	2.03
<b>W400</b>	93.9	2.54	96.2	2.68	94.1	2.54	94.7	2.67
<b>W500</b>	94.2	3.39	95.5	3.42	94.6	3.32	94.8	3.41
<b>W600</b>	94.2	4.32	95.2	4.35	94.4	4.2	94.8	4.36
<b>W700</b>	94.4	6.03	96.2	6.11	93	5.8	95.4	5.97
<b>W800</b>	92.1	8.18	94.5	8.13	90.1	7.91	94.2	8.15
<b>W900</b>	93.2	10.13	93.8	10.31	92.9	10.17	93.2	10.38
<b>W1000</b>	91.2	12.24	91.3	12.64	90.7	12.41	90.8	12.63

Table 5.3. The performance results of GILOFR and GILOF algorithms for unnormalized UCI Pendigit dataset.

<b>W Size</b>	<b>Unnormalize UCI Pendigit Dataset</b>							
	<b>GILOFR</b>		<b>GILOFR_NS</b>		<b>GILOF</b>		<b>GILOF_NS</b>	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	50.7	1.96	51.5	1.93	51.9	2.05	50.2	1.98
<b>W200</b>	51.9	3.61	52.9	3.53	52.5	3.61	52.1	3.56
<b>W300</b>	50.7	5.43	51.7	5.28	54.2	5.43	54.7	5.28
<b>W400</b>	52.5	7.2	55.1	7.14	53.7	7.24	53.2	7.11
<b>W500</b>	53.5	9.72	52.3	9.43	53.3	9.93	52.3	9.44
<b>W600</b>	52.8	12.64	51.4	12.62	50.9	12.95	51	12.65
<b>W700</b>	53.4	18.25	53.8	18.07	52.4	19.1	55.4	18.49
<b>W800</b>	53.5	26.6	53.8	26.08	54.2	27.01	54.3	26.48
<b>W900</b>	53.4	35.68	53.8	35.23	54	36.38	54.8	36.31
<b>W1000</b>	54.1	46.28	54.2	46.21	55.1	47.12	54.2	46.96

Table 5.4. The performance results of GILOFR and GILOF algorithms for normalized UCI Pendigit dataset.

<b>W Size</b>	<b>Normalize UCI Pendigit Dataset</b>							
	<b>GILOFR</b>		<b>GILOFR_NS</b>		<b>GILOF</b>		<b>GILOF_NS</b>	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	95.6	1.99	95.6	1.94	95.9	2.1	96	1.99
<b>W200</b>	98.2	3.6	98.3	3.53	98.6	3.71	98.6	3.52
<b>W300</b>	98.7	5.39	98.7	5.21	98.7	5.52	98.7	5.26
<b>W400</b>	98.1	7.32	98	7.13	98.4	7.22	98.4	7.11
<b>W500</b>	97.1	9.92	97.2	9.41	97.9	9.68	97.7	9.46
<b>W600</b>	96.4	13.22	96.4	12.66	97.7	12.66	97.5	12.64
<b>W700</b>	95	18.9	95.4	18.35	97.2	18.89	96.6	18.05
<b>W800</b>	94	27.24	93	26.41	96.2	26.51	95.3	26.04
<b>W900</b>	92.4	36.27	92.5	35.53	95.7	36.01	95.1	35.49
<b>W1000</b>	89.8	47.19	90.2	46.72	93.5	47.12	93.1	46.31

Table 5.5. The performance results of GILOFR and GILOF algorithms for unnormalized KDD Cup 99 SMTP dataset.

<b>W Size</b>	Unnormalize KDD Cup 99 SMTP Dataset							
	GILOFR		GILOFR_NS		GILOF		GILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	84	46.2	81.7	46.5	85.2	47.1	81.3	46.7
<b>W200</b>	85.6	84.9	85.9	83.4	86.2	84.5	86.8	83.5
<b>W300</b>	85.1	129.8	88	128.5	87.5	128	85.4	127.4
<b>W400</b>	86.3	179.4	85.3	178.8	86	184.5	86.3	177.8
<b>W500</b>	83.6	246.1	83.9	242.9	83.9	244.9	82.8	242.8
<b>W600</b>	82.6	336.4	80.6	333.8	78.8	335.4	76.1	329.7
<b>W700</b>	76.7	502.1	80.1	503.3	70.4	495.7	72.7	493.5
<b>W800</b>	75.7	749	72	759.8	72.7	742.6	69.3	765.1
<b>W900</b>	76.7	1084.9	72	1059	67.9	1058.6	68.3	1061.3
<b>W1000</b>	75.7	1442.1	74.3	1406.6	69.2	1410.8	68.2	1413.6

Table 5.6. The performance results of GILOFR and GILOF algorithms for normalized KDD Cup 99 SMTP dataset.

<b>W Size</b>	Normalize KDD Cup 99 SMTP Dataset							
	GILOFR		GILOFR_NS		GILOF		GILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	87.6	45.09	82.9	45.6	85.1	46.2	81.1	46.8
<b>W200</b>	84.2	83.7	89	83.6	85.2	85.2	86.1	83.8
<b>W300</b>	84	128.4	86.7	129	86.7	147.2	86	127.8
<b>W400</b>	85.5	178.3	83.8	179.1	85.5	201.3	85	177.6
<b>W500</b>	78.2	247.2	79.2	250.5	82.2	250.6	82.7	242.9
<b>W600</b>	74.3	335.5	74	333.5	76.5	337.2	75.7	330.2
<b>W700</b>	69.8	495.3	71.6	497.4	74	497.4	71.3	495.1
<b>W800</b>	70	742.3	68.9	745.7	76.2	746.8	68.2	746.3
<b>W900</b>	75	1060.6	67.9	1063.7	68	1065.1	67.1	1071.8
<b>W1000</b>	70.1	1411.5	67.6	1414.8	69.2	1420.4	68.2	1432.4

Table 5.7. The performance results of GILOFR and GILOF algorithms for unnormalized KDD Cup 99 HTTP dataset.

<b>W Size</b>	Unnormalize KDD Cup 99 HTTP Dataset							
	GILOFR		GILOFR_NS		GILOF		GILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	77.9	270.9	35.9	280.2	76.2	283.3	34.8	284.4
<b>W200</b>	79.3	489.7	43.2	503.4	76.5	496.8	43.6	513.6
<b>W300</b>	78.4	760.5	47.9	781.1	77.6	766	47.2	781.1
<b>W400</b>	78	1062.7	59.5	1090.8	77.2	1080.6	49.3	1094.1
<b>W500</b>	78.1	1485.3	59.2	1488.4	75.4	1533.1	55.4	1522.5
<b>W600</b>	77.4	1986.4	58	2018.8	75.7	2065.7	55.4	2027.7
<b>W700</b>	76.3	2948.6	60	3001.4	77.5	2924.3	56.1	2985.5
<b>W800</b>	74.5	4459.8	65.3	4516.3	73.2	4339.8	61.6	4507.9
<b>W900</b>	77.6	6284	60.7	6514.2	72	6135.9	57.4	6402.4
<b>W1000</b>	76.2	8410	58.2	8555.8	72.7	8158.3	55.6	8560.5

Table 5.8. The performance results of GILOFR and GILOF algorithms for normalized KDD Cup 99 HTTP dataset.

<b>W Size</b>	Normalize KDD Cup 99 HTTP Dataset							
	GILOFR		GILOFR_NS		GILOF		GILOF_NS	
	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>	<b>AUC%</b>	<b>TIME</b>
<b>W100</b>	91.4	269.7	72.4	280.4	89.2	274.5	66.4	288.4
<b>W200</b>	93.6	488.5	81.4	503.1	91.6	490.1	77.1	506.9
<b>W300</b>	92.7	760.7	84	795.1	90.5	757	80.7	780.1
<b>W400</b>	92.2	1075.2	82	1130.1	90.7	1058.8	79.5	1103.4
<b>W500</b>	92.2	1444.3	81.3	1506.5	91.2	1465.8	78.9	1472.7
<b>W600</b>	89.9	2003	80.7	2013.1	89.2	1976.2	77.9	2011.1
<b>W700</b>	88.5	3005.9	78.7	3002	88.3	2975.1	76.3	2989.1
<b>W800</b>	87.3	4523.9	76.6	4520.9	85.6	4440.7	73.9	4545.2
<b>W900</b>	84.6	6372.5	77.1	6403.3	85.2	6289.1	74.9	6398.8
<b>W1000</b>	88.4	8436.5	72.4	8479.5	85.9	8356.6	69.1	8431.5

## 5.6 Conclusion

The objective of the proposed GILOFR algorithm is to further improve the accuracy of outlier detection in the GILOF algorithm for data stream mining. Like GILOF, GILOFR addresses the limitation of the LOF algorithm in a data stream, but GILOFR further improves GILOF. Accordingly, GILOFR does not need to recalculate all previous steps when a new data point occurs. Additionally, it does not require the storage of all the data points in memory, because it can work under the limited memory. Our new calculation of LOF, which is called LOFR, has a positive impact on the GILOF algorithm and leads to more accurate results. The outcomes of experiments demonstrate that GILOFR and GILOFR\_NS are better than GILOF and GILOF\_NS for the accuracy of outlier detection in real-world datasets that have sequence of outliers and high-density region i.e. data points are very close to each other. Specifically, this is true for the KDD Cup99 HTTP dataset, which includes a simulation of normal data, with abnormal data as attack traffic, on an IP scale in computer networks for testing intrusion detection systems.



## Chapter 6: Synthesis and Conclusions

"A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams." *Big Data Cognitive Computing*. 2020, 5, 1. MDPI. <https://doi.org/10.3390/bdcc5010001>

"An Efficient Local Outlier Factor for Data Stream Processing: A Case Study" Forthcoming in *the 7th International Conference on Computational Science and Computational Intelligence 2020*, IEEE.

### 6.1 A Summary of the Key Research Activities and Outcomes

Outlier detection is a statistical procedure that aims to find suspicious events or items that are different from the normal form of a dataset. It has drawn considerable interest in the field of data mining and machine learning. Outlier detection is important in many applications, including fraud detection in credit card transactions and network intrusion detection. There are two general types of outlier detection: global and local. Global outliers fall outside the normal range for an entire dataset, whereas local outliers may fall within the normal range for the entire dataset, but outside the normal range for the surrounding data points. This paper addresses local outlier detection. The best-known technique for local outlier detection is the Local Outlier Factor (LOF), a density-based technique. LOF algorithm cannot be applied directly to data streams, which are an important type of big data. In general, local outlier detection algorithms for data streams are still deficient and better algorithms need to be developed that can effectively analyze the high velocity of data streams to detect local outliers. The issue with LOF is that it needs to store the whole dataset with its distances' results in memory. In addition, it needs to start from the beginning and recalculate all processes if any change happens in the dataset, such as inserting a new data point. This dissertation provide two solution methods for local outlier detection in data stream. In addition, it presents a literature review of local outlier detection algorithms in static and stream environments, with an emphasis on LOF algorithms. It collects and categorizes existing local outlier detection algorithms and analyzes their characteristics.

To address the limitation of the LOF in data streams, new methods should be developed. The primary purpose of any new method would be to measure the LOF score in all of the following circumstances: (1) keeping only a small part of the dataset in the computer memory; (2) the algorithm has no prior knowledge about the data distribution; (3) for an incoming data point  $pt$ , the algorithm

should verify whether it is an outlier or inlier at the current time  $T$ ; and (4) when the algorithm detects the current outlier, it has no prior knowledge about future data points.

To overcome the challenges of the LOF in stream environments, we have designed a new methodology to detect local outliers. This methodology contains two phases: (1) the detection phase and (2) the summarization phase. For the detection phase, the ILOF is used with a skipping scheme [88,104]. For the summarization phase, the Genetic Density Summarization algorithm (GDS), based on the genetic algorithm (GA), is used to summarize the dataset. The framework of our methodology, named Genetic-based Incremental Local Outlier Factor (GILOF) [181], works as follows: First, the maximum size of the window ( $W$ ) is determined as  $W\text{-size}$ . After that, the threshold of the LOF is applied to detect the local outliers, relying on the threshold  $\theta$ , then using the GDS to summarize the data points. Thereafter, the GILOF uses the ILOF together with a skipping scheme to detect the local outlier when an incoming data point arrives. It is worth noting that the skipping scheme is used in order to detect the sequence of outliers, when the sequence of outliers is outlier data points that are trying to build a new class. The GILOF continues to detect the outlier data points and to compute the LOF values for every new data point until the window reaches the  $W\text{-size}$ . When the window becomes full, the GDS algorithm is applied to the window in order to summarize  $50\%W$  of the older data points in the window; it does so by choosing  $25\%W$  of the fittest data points to represent the  $50\%W$  of the older data points. After this, the GILOF deletes the older  $50\%W$  of the data points from the window and the selected  $25\%W$  of data points is transferred to the window and merges with the remaining  $50\%W$ . These  $75\%W$  of data points joins with the newly incoming data points in the window. When the window reaches the  $W\text{-size}$  again, the GDS repeats the same process. The video in [168] displays a simulation of the GILOF system process. Figure 6.1 shows the overall design and workflow for the methodology.

Local Outlier Factor by Reachability Distance (LOFR) is similar to the LOF, but the LOFR has a different calculation method, in which it does not need to apply the local reachability density [191]. To calculate the score of outlierness, the LOFR uses  $k$ -distance,  $k$ -nearest neighbor, and Reachability distance  $Rd$ . Subsequently, the Reachability distance  $Rd$  of data point  $p$  will be divided by the average

$Rd$  of the data point  $p$  neighbors. This new calculation method for local outlier detection can produce a lower “outlierness” score than the LOF. The LOFR can produce a more accurate outlierness score in various datasets. The LOFR score is calculated by using Equation (6).

$$LOFR_k(p) = \sum_{o \in N_k(p)} \frac{Rd_k(p)}{\left(\frac{Rd_k(o)}{k}\right)} \quad (6)$$

The GILOF algorithm is discussed extensively in [181]. By trying to enhance the effectiveness of the GILOF algorithm, we propose another calculation method for the LOF, which is named LOFR. The newly adapted algorithm in the data stream is named Genetic-based Incremental Local Outlier Factor by Reachability Distance (GILOFR). The GILOFR algorithm is also extensively discussed in [191].

## 6.2 Summary of answers to research questions

In this dissertation, six research questions have been addressed. This section aims to summarize the answers of the research questions, which are already mentioned in chapter 1. However, the research questions are listed below again, to facilitate the process of linking questions and answers to the reader.

The research questions of this dissertation are:

- 1) How can the GILOF algorithm solve the issue of memory consumption?
- 2) How does the GILOF algorithm detect the LOF in a data stream?
- 3) How does the genetic algorithm retain the shape of the density of data points?
- 4) Can a skipping scheme make a difference in the accuracy of local outlier detection and why?
- 5) Do GILOF and GILOF\_NS perform better than DILOF and DILOF\_NS when considering the accuracy of outlier detection?
- 6) Do GILOF and GILOF\_NS perform better than DILOF and DILOF\_NS when considering execution time?

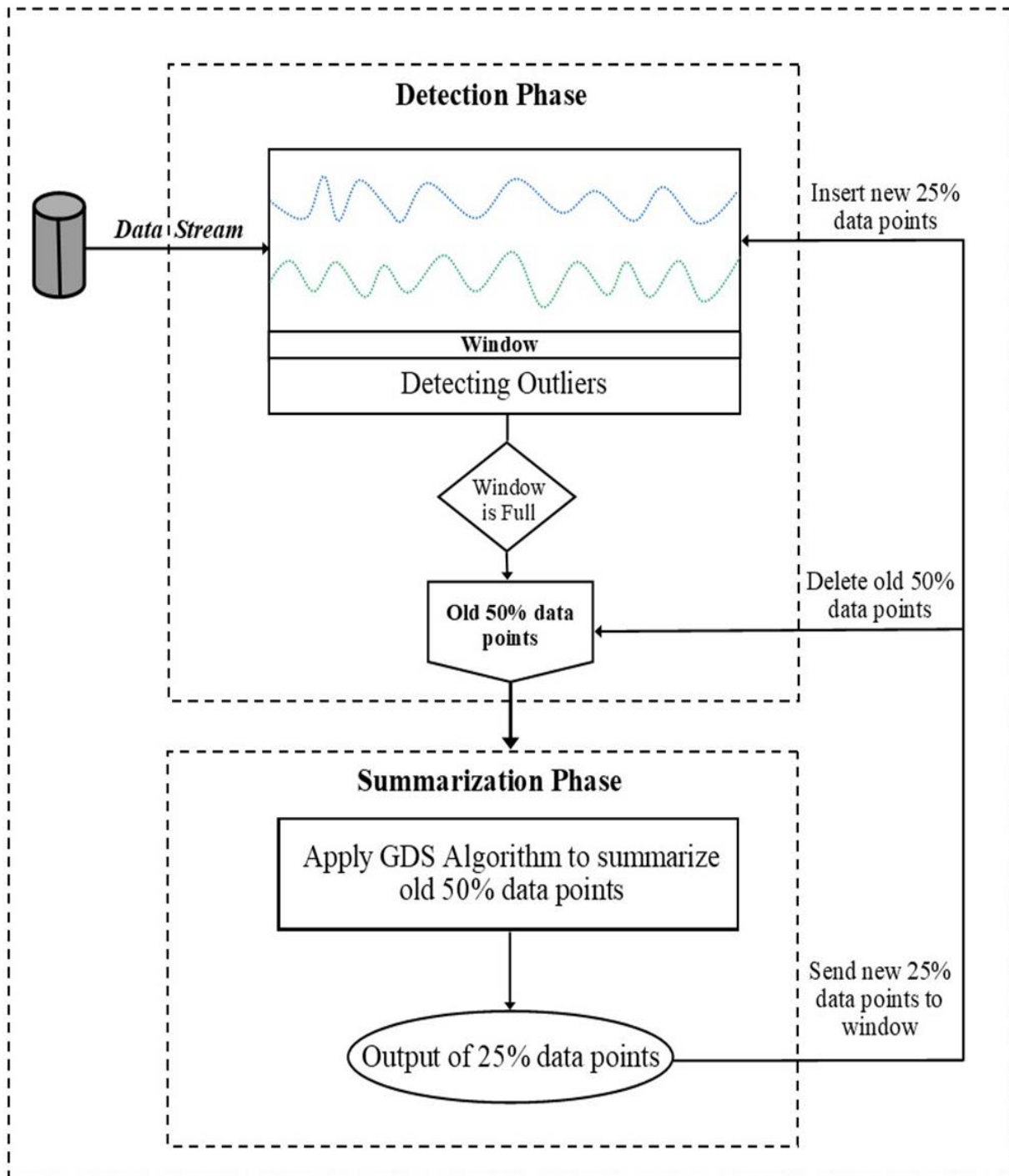


Figure 6.1. The overall design and workflow for the methodology.

To answer the first question, we used limited memory by summarizing the dataset and determining a specific window size by using sliding window. The second question was solved by using ILOF and ILOFR algorithms. In the third question, we used two strategies, first, we summarized the old 50% of data points and kept the rest. Second, we summarized the old 50% of data points by using the genetic algorithm. For the fourth question, we developed two versions of the GILOF algorithm, one with a skipping scheme, which is called GILOF and another one without a skipping scheme, which is called GILOF\_NS. The goal of that is to show the different results between them. In addition, the reason for the impact of the skipping scheme is that it calculates the average distance of data points in the window, and also it calculates the distance between the new inserted data point and the latest detected outlier. If the distance between the new inserted data point and the last detected outlier is less than the distance between the new inserted data point and the average distance of data points, the new data point is considered an outlier. For the questions 5 and 6, GILOF, GILOF\_NS, and their next improvement, GILOFR and GILOFR\_NS, show better results in accuracy of outlier detection and execution time for the most datasets. the results comparison of all algorithms is extensively discussed in the chapters 4 and 5. Table 6.1 shows the direction to the research question answers. In addition, figure 6.2 shows the outline of the dissertation chapters.

Table 6.1. The direction to the research question answers.

No.	Research Questions	Chapters	Pages
1	How can the GILOF algorithm solve the issue of memory consumption?	Ch4	Pg53
2	How does the GILOF algorithm detect the LOF in a data stream?	Ch2 & Ch5	Pg14 & Pg79
3	How does the genetic algorithm retain the shape of the density of data points?	Ch4 & Ch5	Pg56 & Pg81
4	Can a skipping scheme make a difference in the accuracy of local outlier detection and why?	Ch4	Pg56
5	Do GILOF and GILOF_NS perform better than DILOF and DILOF_NS when considering the accuracy of outlier detection?	Ch4 & Ch5	Pg61 & Pg85
6	Do GILOF and GILOF_NS perform better than DILOF and DILOF_NS when considering execution time?	Ch4 & Ch5	Pg63 & Pg86

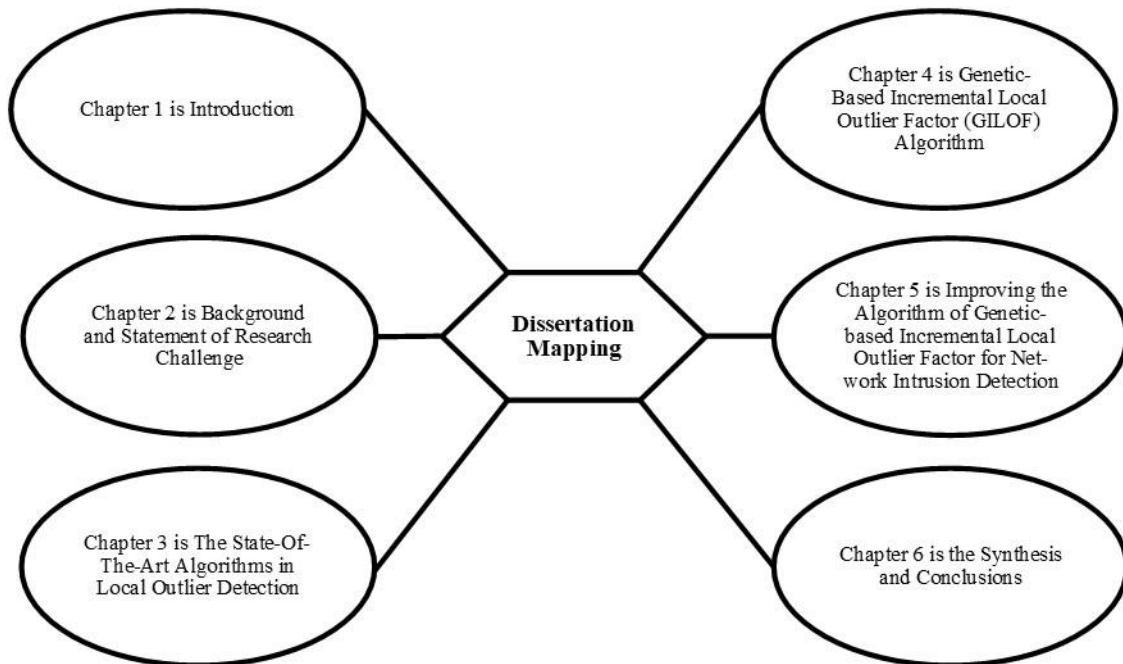


Figure 6.2. The dissertation division map.

### 6.3 Main Conclusions

In the big data era, outlier detection is a very important step in many applications, such as network intrusion detection systems and decision support systems. The objective of outlier detection is to detect suspicious items and unusual activities. For example, in practice, analyzing the dataset to extract information without removing the outlier's data will lead to inaccurate information, which will result in wrong decisions. Recently, outlier detection has gained a lot of attention from researchers, especially regarding data streams. This dissertation proposed a new possibility for local outlier detection in data streams by developing two methods, which are called the Genetic Based-Incremental Local Outlier factor (GILOF) and the Local Outlier Factor by Reachability Distance (LOFR). As mentioned above, the GA is assumed to be better than the gradient-descent because the genetic algorithm is a population-based search technique and can jump the local minima by the aid of its crossover and mutation operators for a more wide search of spaces. The GILOF algorithm is already compared with the DILOF algorithm [104] and the results are extensively discussed in [181]. By trying to improve the

efficiency of the GILOF algorithm, we developed another calculation method for the LOF, which is called LOFR. The new algorithm is named GILOFR. The GILOFR algorithm was compared with the GILOF algorithm; the outcomes are also extensively discussed in [191].

For future work, the other traditional local outlier detection algorithms, such as the COF, LoOP, LOCI, and the INFLO, etc., can be adapted to work in a data stream. To execute these traditional algorithms in a data stream, the mechanisms of the above-mentioned methods, such as the GILOF and DILOF algorithms, should be applied. The LOFR was also applied to another algorithm that is called Grid Partition-Based Local Outlier Factor and it showed slight improvement in some datasets [112]. this new calculation method of the LOFR can be applied in the DILOF algorithm instead of the LOF, which may lead to more accurate results. Finally, this dissertation addressed specific issues and challenges of the LOF in stream environments and provided new methods to improve the efficiency of local outlier detection in data streams. It also recommend new works of local outlier detection in data streams.

## 6.4 List of Publications

During my study, I published one book chapter for Encyclopedia of Big Data. I participated in the UI Research Computing and Data Science Symposium by presenting (poster) the research idea about summarizing big data using a genetic algorithm for local outlier detection. On the other hand, I invited to review for IEEE Access Journal. In addition, I gained two Travel Awards from GPSA at University of Idaho. The following is the list of publications:

### 6.4.1 As a first author

[1] Paper title: Data Storage

Description: Book chapter in the encyclopedia of big data. published online by Springer.

Reference: Alghushairy, O. and Ma, X., 2019. Data Storage. *Encyclopedia of Big Data*; Schintler, L., McNeely, C., Eds.

- [2] Paper title: A Genetic-Based Incremental Local Outlier Factor Algorithm for Efficient Data Stream Processing.

Description: Conference paper in the 4<sup>th</sup> international conference in compute and data analysis. Silicon Valley CA USA March, 2020. published online by ACM.

Reference: Alghushairy, O., Alsini, R., Ma, X. and Soule, T., 2020, March. A Genetic-Based Incremental Local Outlier Factor Algorithm for Efficient Data Stream Processing. In *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis* (pp. 38-49).

- [3] Paper title: Improving the Efficiency of Genetic based Incremental Local Outlier Factor Algorithm for Network Intrusion Detection.

Description: Conference paper in of the 4<sup>th</sup> International Conference on Applied Cognitive Computing, Las Vegas, NV, USA. published online by Springer.

Reference: Alghushairy, O., Alsini, R., Ma, X. and Soule, T., 2020, July. Improving the Efficiency of Genetic based Incremental Local Outlier Factor Algorithm for Network Intrusion Detection. In *Proceedings of the 4th International Conference on Applied Cognitive Computing, Las Vegas, NV, USA* (pp. 27-30).

- [4] Paper title: A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams.

Description: Journal paper in the Journal of Big Data and Cognitive Computing. published online by MDPI.

Reference: Alghushairy, O., Alsini, R., Soule, T. and Ma, X., 2021. A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams. *Big Data and Cognitive Computing*, 5(1), p.1.

- [5] Paper title: An Efficient Local Outlier Factor for Data Stream Processing: A Case Study.

Description: Conference paper in the 7<sup>th</sup> International Conference on Computational Science and Computational Intelligence. published online by IEEE.

Reference: Alghushairy, O., Alsini, R and Ma, X., 2020, December. An Efficient Local Outlier Factor for Data Stream Processing: A Case Study. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, Las Vegas, NV, USA*.



#### 6.4.2 As Co-Author

- [6] Paper title: A Grid Partition-based Local Outlier Factor for Data Stream Processing.

Description: Conference paper in the 4<sup>th</sup> International Conference on Applied Cognitive Computing, Las Vegas, NV, USA. published online by Springer.

Reference: Alsini, R., Alghushairy, O., Ma, X. and Soule, T., 2020, July. A Grid Partition-based Local Outlier Factor for Data Stream Processing. *In Proceedings of the 4th International Conference on Applied Cognitive Computing, Las Vegas, NV, USA.*

- [7] Paper title: A Grid Partition-Based Local Outlier Factor by Reachability Distance for Data Stream Processing.

Description: Conference paper in the 7<sup>th</sup> International Conference on Computational Science and Computational Intelligence. published online by IEEE.

Reference: Alsini, R., Alghushairy, O., Ma, X. and Soule, T., 2020, December. A Grid Partition-Based Local Outlier Factor by Reachability Distance for Data Stream Processing. *In 2020 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE., Las Vegas, NV, USA.*

- [8] Paper title: Local Outlier Detection Techniques in Real-World Big Data Processing: A Literature Review.

Description: Journal paper that **submitted** to the Journal of Algorithm. Will published online by MDPI.

## References

1. Sadik, S. and Gruenwald, L., 2014. Research issues in outlier detection for data streams. *Acm Sigkdd Explorations Newsletter*, 15(1), pp.33-40.
2. Tellis, V.M. and D'Souza, D.J., 2018, March. Detecting Anomalies in Data Stream Using Efficient Techniques: A Review. In *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCT)* (pp. 296-298). IEEE.
3. Thakkar, P., Vala, J. and Prajapati, V., 2016. Survey on outlier detection in data stream. *Int. J. Comput. Appl*, 136, pp.13-16.
4. Souiden, I., Brahmi, Z. and Toumi, H., 2016, December. A survey on outlier detection in the context of stream mining: review of existing approaches and recommendations. In *International Conference on Intelligent Systems Design and Applications* (pp. 372-383). Springer, Cham.
5. Alghushairy, O., Alsini, R and Ma, X., 2020, December. An Efficient Local Outlier Factor for Data Stream Processing: A Case Study. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE.
6. García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J.M. and Herrera, F., 2016. Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1), p.9.
7. Alghushairy, O. and Ma, X., 2019. Data Storage. *Encyclopedia of Big Data; Schintler, L., McNeely, C., Eds.*
8. Margara, A. and Rabl, T., 2019. Definition of Data Streams. *Encyclopedia of Big Data Technologies*, pp. 1-4.
9. Kreml, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M. and Stefanowski, J., 2014. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1), pp.1-10.
10. Younas. M., 2019, Jun. Research challenges of big data. *Service Oriented Comput. Appl.*, vol. 13, pp. 105-107.
11. Minelli, M., Chambers, M. and Dhiraj, A., 2013. *Big data, big analytics: emerging business intelligence and analytic trends for today's businesses* (Vol. 578). John Wiley & Sons.
12. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E. and Bouwman, J., 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, 3(1), pp.1-9.
13. Ma,X. 2018. Metadata. *Encyclopedia of Big Data; Schintler, L., McNeely, C., Eds. Cham: Springer.* [https://doi.org/10.1007/978-3-319-32001-4\\_135-1](https://doi.org/10.1007/978-3-319-32001-4_135-1).
14. Savage, T.M. and Vogel, K.E., 2014. *An introduction to digital multimedia*. Jones & Bartlett Publishers.
15. Strohbach, M., Daubert, J., Ravkin, H. and Lischka, M., 2016. Big data storage. In *New horizons for a data-driven economy* (pp. 119-141). Springer, Cham.
16. Boukerche, A., Zheng, L. and Alfandi, O., 2020. Outlier Detection: Methods, Models, and Classification. *ACM Computing Surveys*, 53(3).

17. Cios, K.J., Pedrycz, W. and Swiniarski, R.W., 1998. Data mining and knowledge discovery. In *Data mining methods for knowledge discovery* (pp. 1-26). Springer, Boston, MA.
18. Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M. and Herrera, F., 2017. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239, pp.39-57.
19. Kumar, V., 2005. Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems Online*, 6(10).
20. Spence, C., Parra, L. and Sajda, P., 2001, December. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Proceedings IEEE workshop on mathematical methods in biomedical image analysis (MMBIA 2001)* (pp. 3-10). IEEE.
21. Fujimaki, R., Yairi, T. and Machida, K., 2005, August. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 401-410).
22. Knox, E.M. and Ng, R.T., 1998, August. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the international conference on very large data bases* (pp. 392-403). Citeseer.
23. Patcha, A. and Park, J.M., 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12), pp.3448-3470.
24. Snyder, D., 2001. On-line intrusion detection using sequences of system calls. Master's Thesis, Department of Computer Science, Florida State University, Tallahassee, FL, USA.
25. Markou, M. and Singh, S., 2003. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12), pp.2481-2497.
26. Markou, M. and Singh, S., 2003. Novelty detection: a review—part 2: neural network based approaches. *Signal processing*, 83(12), pp.2499-2521.
27. Hodge, V. and Austin, J., 2004. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2), pp.85-126.
28. Goldstein, M. and Uchida, S., 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4), p.e0152173.
29. Wang, H., Bah, M.J. and Hammad, M., 2019. Progress in outlier detection techniques: A survey. *IEEE Access*, 7, pp.107964-108000.
30. Park, C.H., 2019. Outlier and anomaly pattern detection on data streams. *The Journal of Supercomputing*, 75(9), pp.6118-6128.
31. Pimentel, M.A., Clifton, D.A., Clifton, L. and Tarassenko, L., 2014. A review of novelty detection. *Signal Processing*, 99, pp.215-249.
32. Chauhan, P. and Shukla, M., 2015, March. A review on outlier detection techniques on data stream by using different approaches of K-Means algorithm. In *2015 International Conference on Advances in Computer Engineering and Applications* (pp. 580-585). IEEE.
33. Salehi, M. and Rashidi, L., 2018. A Survey on Anomaly detection in Evolving Data: [with Application to Forest Fire Risk Prediction]. *ACM SIGKDD Explorations Newsletter*, 20(1), pp.13-23.

34. Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), pp.1-58.
35. Domingues, R., Filippone, M., Michiardi, P. and Zouaoui, J., 2018. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74, pp.406-421.
36. Safaei, M., Asadi, S., Driss, M., Boulila, W., Alsaeedi, A., Chizari, H., Abdullah, R. and Safaei, M., 2020. A Systematic Literature Review on Outlier Detection in Wireless Sensor Networks. *Symmetry*, 12(3), p.328.
37. Barnett, V. and Lewis, T., 1984. Outliers in statistical data. *osd*.
38. Zimek, A. and Filzmoser, P., 2018. There and back again: Outlier detection between statistical reasoning and data mining algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(6), p.e1280.
39. Eskin, E., 2000. Anomaly detection over noisy data using learned probability distributions. in *Proc. 17th Int. Conf. Mach. Learn.*
40. *Maximum Likelihood Estimation*, 2015. Available online: [https://en.wikipedia.org/w/index.php?title=Maximum\\_likelihood\\_estimation&oldid=857905834](https://en.wikipedia.org/w/index.php?title=Maximum_likelihood_estimation&oldid=857905834) (accessed on date 30 November 2020).
41. Yang, X., Latecki, L.J. and Pokrajac, D., 2009, April. Outlier detection with globally optimal exemplar-based GMM. In *Proceedings of the 2009 SIAM International Conference on Data Mining* (pp. 145-154). Society for Industrial and Applied Mathematics.
42. Tang, X.M., Yuan, R.X. and Chen, J., 2015. Outlier detection in energy disaggregation using subspace learning and Gaussian mixture model. *Int. J. Control Autom.*, 8(8), pp.161-170.
43. Zhang, J., 2013. Advancements of outlier detection: A survey. *ICST Transactions on Scalable Information Systems*, 13(1), pp.1-26.
44. Satman, M.H., 2013. A new algorithm for detecting outliers in linear regression. *International Journal of statistics and Probability*, 2(3), p.101.
45. Park, C.M. and Jeon, J., 2015, August. Regression-based outlier detection of sensor measurements using independent variable synthesis. In *International Conference on Data Science* (pp. 78-86). Springer, Cham.
46. Pavlidou, M. and Zioutas, G., 2014. Kernel density outlier detector. In *Topics in Nonparametric Statistics* (pp. 241-250). Springer, New York, NY.
47. Latecki, L.J., Lazarevic, A. and Pokrajac, D., 2007, July. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 61-75). Springer, Berlin, Heidelberg.
48. Gao, J., Hu, W., Zhang, Z.M., Zhang, X. and Wu, O., 2011, May. RKOF: robust kernel-based local outlier detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 270-283). Springer, Berlin, Heidelberg.
49. Samparathi, V.K. and Verma, H.K., 2010. Outlier detection of data in wireless sensor networks using kernel density estimation. *International Journal of Computer Applications*, 5(7), pp.28-32.

50. Edgeworth, F.Y., 1887. Xli. on discordant observations. *The london, edinburgh, and dublin philosophical magazine and journal of science*, 23(143), pp.364-375.
51. Rousseeuw, P.J. and Leroy, A.M., 2005. *Robust regression and outlier detection* (Vol. 589). John Wiley & sons.
52. Hawkins, D.M., 1980. *Identification of outliers* (Vol. 11). London: Chapman and Hall.
53. Clarke, K.R. and Warwick, R.M., 1994. An approach to statistical analysis and interpretation. *Change in marine communities*, 2, pp.117-143.
54. Bakar, Z.A., Mohamad, R., Ahmad, A. and Deris, M.M., 2006, June. A comparative study for outlier detection techniques in data mining. In *2006 IEEE conference on cybernetics and intelligent systems* (pp. 1-6). IEEE.
55. Aggarwal, C.C., 2015. Outlier analysis. In *Data mining* (pp. 237-263). Springer, Cham.
56. Meng, F., Yuan, G., Lv, S., Wang, Z. and Xia, S., 2019. An overview on trajectory outlier detection. *Artificial Intelligence Review*, 52(4), pp.2437-2456.
57. Gökalp, E., Güngör, O. and Boz, Y., 2008. Evaluation of different outlier detection methods for GPS networks. *Sensors*, 8(11), pp.7344-7358.
58. Breunig, M.M., Kriegel, H.P., Ng, R.T. and Sander, J., 2000, May. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (pp. 93-104).
59. Joshi, M.V., Agarwal, R.C. and Kumar, V., 2001, May. Mining needle in a haystack: classifying rare classes via two-phase rule induction. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data* (pp. 91-102).
60. Joshi, M.V., Agarwal, R.C. and Kumar, V., 2002, July. Predicting rare classes: Can boosting make any weak learner strong?. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 297-306).
61. Chawla, N.V., Japkowicz, N. and Kotcz, A., 2004. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1), pp.1-6.
62. Sen, P.C., Hajra, M. and Ghosh, M., 2020. Supervised classification algorithms in machine learning: A survey and review. In *Emerging Technology in Modelling and Graphics* (pp. 99-111). Springer, Singapore.
63. Salzberg, S.L., 1994. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993.
64. Mehrotra, K., Mohan, C.K. and Ranka, S., 1997. *Elements of artificial neural networks*. MIT press.
65. Moya, M.M. and Hush, D.R., 1996. Network constraints and multi-objective optimization for one-class classification. *Neural networks*, 9(3), pp.463-474.

66. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J. and Williamson, R.C., 2001. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7), pp.1443-1471.
67. Tang, J., Chen, Z., Fu, A.W.C. and Cheung, D.W., 2002, May. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 535-548). Springer, Berlin, Heidelberg.
68. Papadimitriou, S., Kitagawa, H., Gibbons, P.B. and Faloutsos, C., 2003, March. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)* (pp. 315-326). IEEE.
69. He, Z., Xu, X. and Deng, S., 2003. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10), pp.1641-1650.
70. Jin, W., Tung, A.K., Han, J. and Wang, W., 2006, April. Ranking outliers using symmetric neighborhood relationship. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 577-593). Springer, Berlin, Heidelberg.
71. Kriegel, H.P., Kröger, P., Schubert, E. and Zimek, A., 2009, November. LoOP: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 1649-1652).
72. Amer, M. and Goldstein, M., 2012, August. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)* (pp. 1-12).
73. Chiu, A.L.M. and Fu, A.W.C., 2003, July. Enhancements on local outlier detection. In *Seventh International Database Engineering and Applications Symposium, 2003. Proceedings.* (pp. 298-307). IEEE.
74. Jiang, S.Y., Li, Q.H., Li, K.L., Wang, H. and Meng, Z.L., 2003, November. GLOF: a new approach for mining local outlier. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)* (Vol. 1, pp. 157-162). IEEE.
75. Ren, D., Wang, B. and Perrizo, W., 2004, November. Rdf: A density-based outlier detection method using vertical data representation. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (pp. 503-506). IEEE.
76. Lozano, E. and Acufia, E., 2005, November. Parallel algorithms for distance-based and density-based outliers. In *Fifth IEEE International Conference on Data Mining (ICDM'05)* (pp. 4-pp). IEEE.
77. Fan, H., Zaïane, O.R., Foss, A. and Wu, J., 2009. Resolution-based outlier factor: detecting the top-n most outlying data points in engineering data. *Knowledge and Information Systems*, 19(1), pp.31-51.
78. Momtaz, R., Mohssen, N. and Gowayyed, M.A., 2013, June. DWOFF: A robust density-based outlier detection approach. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 517-525). Springer, Berlin, Heidelberg.
79. Cao, K., Shi, L., Wang, G., Han, D. and Bai, M., 2014, June. Density-based local outlier detection on uncertain data. In *International Conference on Web-Age Information Management* (pp. 67-71). Springer, Cham.

80. Goldstein, M. Anomaly Detection in Large Datasets. Ph.D. thesis, University of Kaiserslautern, Kaiserslautern, Germany, 2016.
81. Tang, B. and He, H., 2017. A local density-based approach for outlier detection. *Neurocomputing*, 241, pp.171-180.
82. Vázquez, F.I., Zseby, T. and Zimek, A., 2018, November. Outlier detection based on low density models. In *2018 IEEE international conference on data mining workshops (ICDMW)* (pp. 970-979). IEEE.
83. Ning, J., Chen, L. and Chen, J., 2018, December. Relative density-based outlier detection algorithm. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence* (pp. 227-231).
84. Su, S., Xiao, L., Ruan, L., Gu, F., Li, S., Wang, Z. and Xu, R., 2018. An efficient density-based local outlier detection approach for scattered data. *IEEE Access*, 7, pp.1006-1020.
85. Zhao, Y., Nasrullah, Z., Hryniewicki, M.K. and Li, Z., 2019, May. LSCP: Locally selective combination in parallel outlier ensembles. In *Proceedings of the 2019 SIAM International Conference on Data Mining* (pp. 585-593). Society for Industrial and Applied Mathematics.
86. Xu, Z., Kakde, D. and Chaudhuri, A., 2019, December. Automatic Hyperparameter Tuning Method for Local Outlier Factor, with Applications to Anomaly Detection. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 4201-4207). IEEE.
87. Yang, P., Wang, D., Wei, Z., Du, X. and Li, T., 2019. An outlier detection approach based on improved self-organizing feature map clustering algorithm. *IEEE Access*, 7, pp.115914-115925.
88. Pokrajac, D., Lazarevic, A. and Latecki, L.J., 2007, March. Incremental local outlier detection for data streams. In *2007 IEEE symposium on computational intelligence and data mining* (pp. 504-515). IEEE.
89. Pokrajac, D., Reljin, N., Pejic, N. and Lazarevic, A., 2008, September. Incremental connectivity-based outlier factor algorithm. In *Visions of Computer Science-BCS International Academic Conference* (pp. 211-223).
90. Ren, J., Wu, Q., Zhang, J. and Hu, C., 2009, August. Efficient outlier detection algorithm for heterogeneous data streams. In *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery* (Vol. 5, pp. 259-264). IEEE.
91. Karimian, S.H., Kelarestaghi, M. and Hashemi, S., 2012, May. I-inclof: improved incremental local outlier detection for data streams. In *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)* (pp. 023-028). IEEE.
92. Wang, Z., Zhao, Z., Weng, S. and Zhang, C., 2015. Incremental multiple instance outlier detection. *Neural Computing and Applications*, 26(4), pp.957-968.
93. Salehi, M., Leckie, C., Bezdek, J.C. and Vaithianathan, T., 2015, April. Local outlier detection for data streams in sensor networks: Revisiting the utility problem invited paper. In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)* (pp. 1-6). IEEE.

94. Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsihlias, K. and Manolopoulos, Y., 2016. Efficient and flexible algorithms for monitoring distance-based outliers over data streams. *Information systems*, 55, pp.37-53.
95. Zhang, L., Lin, J. and Karim, R., 2016. Sliding window-based fault detection from high-dimensional data streams. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(2), pp.289-303.
96. Salehi, M., Leckie, C., Bezdek, J.C., Vaithianathan, T. and Zhang, X., 2016. Fast memory efficient local outlier detection in data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(12), pp.3246-3260.
97. Hamlet, C., Straub, J., Russell, M. and Kerlin, S., 2017. An incremental and approximate local outlier probability algorithm for intrusion detection and its evaluation. *Journal of Cyber Security Technology*, 1(2), pp.75-87.
98. Siffer, A., Fouque, P.A., Termier, A. and Largouet, C., 2017, August. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1067-1075).
99. Mu, X., Ting, K.M. and Zhou, Z.H., 2017. Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE Transactions on Knowledge and Data Engineering*, 29(8), pp.1605-1618.
100. Ishimtsev, V., Bernstein, A., Burnaev, E. and Nazarov, I., 2017, May. Conformal  $k$ -NN Anomaly Detector for Univariate Data Streams. In *Conformal and Probabilistic Prediction and Applications* (pp. 213-227). PMLR.
101. Chen, Q., Luley, R., Wu, Q., Bishop, M., Linderman, R.W. and Qiu, Q., 2017. AnRAD: A neuromorphic anomaly detection framework for massive concurrent data streams. *IEEE transactions on neural networks and learning systems*, 29(5), pp.1622-1636.
102. Yao, H., Fu, X., Yang, Y. and Postolache, O., 2018. An incremental local outlier detection method in the data stream. *Applied Sciences*, 8(8), p.1248.
103. Munir, M., Siddiqui, S.A., Dengel, A. and Ahmed, S., 2018. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7, pp.1991-2005.
104. Na, G.S., Kim, D. and Yu, H., 2018, July. DILOF: Effective and memory efficient local outlier detection in data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1993-2002).
105. Manzoor, E., Lamba, H. and Akoglu, L., 2018, July. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1963-1972).
106. Yang, X., Zhou, W., Shu, N. and Zhang, H., 2019, February. A Fast and Efficient Local Outlier Detection in Data Streams. In *Proceedings of the 2019 International Conference on Image, Video and Signal Processing* (pp. 111-116).
107. Qin, X., Cao, L., Rundensteiner, E.A. and Madden, S., 2019, March. Scalable Kernel Density Estimation-based Local Outlier Detection over Large Data Streams. In *EDBT* (pp. 421-432).



108. Kalliantzis, I., Papadopoulos, A., Gounaris, A. and Tsihclas, K., 2019. Efficient Distributed Outlier Detection in Data Streams. [Research Report] Aristotle University of Thessaloniki; 54124 Thessaloniki; Greece.
109. Cai, S., Li, Q., Li, S., Yuan, G. and Sun, R., 2019. WMFP-Outlier: An efficient maximal frequent-pattern-based outlier detection approach for weighted data streams. *Information Technology and Control*, 48(4), pp.505-521.
110. Reunanen, N., Rätty, T., Jokinen, J.J., Hoyt, T. and Culler, D., 2019. Unsupervised online detection and prediction of outliers in streams of sensor data. *International Journal of Data Science and Analytics*, pp.1-30.
111. Din, S.U. and Shao, J., 2020. Exploiting evolving micro-clusters for data stream classification with emerging class detection. *Information Sciences*, 507, pp.404-420.
112. Alsini, R., Alghushairy, O., Ma, X. and Soule, T., 2020, July. A Grid Partition-based Local Outlier Factor for Data Stream Processing. In *Proceedings of the 4th International Conference on Applied Cognitive Computing, Las Vegas, NV, USA* (pp. 27-30).
113. Portnoy, L., 2000. *Intrusion detection with unlabeled data using clustering* (Doctoral dissertation, Columbia University).
114. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G. and Vázquez, E., 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2), pp.18-28.
115. Yeung, D.Y. and Ding, Y., 2003. Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, 36(1), pp.229-243.
116. Phua, C., Lee, V., Smith, K. and Gayler, R., 2010. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*.
117. Thiprungsri, S. and Vasarhelyi, M.A., 2011. Cluster Analysis for Anomaly Detection in Accounting Data: An Audit Approach. *International Journal of Digital Accounting Research*, 11.
118. Bolton, R.J. and Hand, D.J., 2001. Unsupervised profiling methods for fraud detection. *Credit scoring and credit control VII*, pp.235-255.
119. Bansal, R., Gaur, N. and Singh, S.N., 2016, January. Outlier detection: applications and techniques in data mining. In *2016 6th International Conference-Cloud System and Big Data Engineering (Confluence)* (pp. 373-377). IEEE.
120. Lin, J., Keogh, E., Fu, A. and Van Herle, H., 2005, June. Approximations to magic: Finding unusual medical time series. In *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)* (pp. 329-334). IEEE.
121. Balcázar, J.L., Bonchi, F., Gionis, A. and Sebag, M., 2010. *Machine learning and knowledge discovery in databases* (Vol. 6321).
122. Fawzy, A., Mokhtar, H.M. and Hegazy, O., 2013. Outliers detection and classification in wireless sensor networks. *Egyptian Informatics Journal*, 14(2), pp.157-164.

123. Jain, A.K. and Dubes, R.C., 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.
124. Boulila, W., Farah, I.R., Ettabaa, K.S., Solaiman, B. and Ghézala, H.B., 2011. A data mining based approach to predict spatiotemporal changes in satellite images. *International Journal of Applied Earth Observation and Geoinformation*, 13(3), pp.386-395.
125. Han, J., Pei, J. and Kamber, M., 2011. *Data mining: concepts and techniques*. Elsevier.
126. Cugola, G. and Margara, A., 2012. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3), pp.1-62.
127. Namiot, D., 2015. On big data stream processing. *International Journal of Open Information Technologies*, 3(8).
128. McKnight, W., 2013. *Information management: strategies for gaining a competitive advantage with data*. Newnes.
129. Widom, J., 2003, January. Query processing, approximation, and resource management in a data stream management system. In *Proc. First Biennial Conf. on Innovative Data Systems Research (CIDR)*.
130. Ramaswamy, S., Rastogi, R. and Shim, K., 2000, May. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (pp. 427-438).
131. Thakran, Y. and Toshniwal, D., 2012, November. Unsupervised outlier detection in streaming data using weighted clustering. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 947-952). IEEE.
132. Cao, F., Estert, M., Qian, W. and Zhou, A., 2006, April. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining* (pp. 328-339). Society for industrial and applied mathematics.
133. Solaimani, M., Iftekhhar, M., Khan, L. and Thuraisingham, B., 2014, October. Statistical technique for online anomaly detection using spark over heterogeneous data from multi-source vmware performance data. In *2014 IEEE International Conference on Big Data (Big Data)* (pp. 1086-1094). IEEE.
134. Zhang, Y., Meratnia, N. and Havinga, P., 2010. Outlier detection techniques for wireless sensor networks: A survey. *IEEE communications surveys & tutorials*, 12(2), pp.159-170.
135. Kale, A. and Ingle, M.D., 2015. SVM based feature extraction for novel class detection from streaming data. *International Journal of Computer Applications*, 110(9).
136. Masud, M.M., Chen, Q., Khan, L., Aggarwal, C.C., Gao, J., Han, J., Srivastava, A. and Oza, N.C., 2012. Classification and adaptive novel class detection of feature-evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 25(7), pp.1484-1497.
137. Sun, L., Zhou, K., Zhang, X. and Yang, S., 2018. Outlier data treatment methods toward smart grid applications. *IEEE Access*, 6, pp.39849-39859.
138. Lin, F., Le, W. and Bo, J., 2010. Research on maximal frequent pattern outlier factor for online high dimensional time-series outlier detection. *Journal of convergence information technology*, 5(10), pp.66-71.

139. Knox, E.M. and Ng, R.T., 1998, August. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the international conference on very large data bases* (pp. 392-403). Citeseer.
140. Angiulli, F. and Fassetti, F., 2007, November. Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (pp. 811-820).
141. Yang, D., Rundensteiner, E.A. and Ward, M.O., 2009, March. Neighbor-based pattern detection for windows over streaming data. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 529-540).
142. Niennattrakul, V., Keogh, E. and Ratanamahatana, C.A., 2010, December. Data editing techniques to allow the application of distance-based outlier detection to streams. In *2010 IEEE International Conference on Data Mining* (pp. 947-952). IEEE.
143. Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsihlias, K. and Manolopoulos, Y., 2011, April. Continuous monitoring of distance-based outliers over data streams. In *2011 IEEE 27th International Conference on Data Engineering* (pp. 135-146). IEEE.
144. Sun, P. and Chawla, S., 2004, November. On local spatial outliers. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (pp. 209-216). IEEE.
145. Yu, J.X., Qian, W., Lu, H. and Zhou, A., 2006. Finding centric local outliers in categorical/numerical spaces. *Knowledge and Information Systems*, 9(3), pp.309-338.
146. Dua, D. and Graff, C., 2019. UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.
147. Shebuti Rayana (2016). ODDS Library [<http://odds.cs.stonybrook.edu>]. Stony Brook, NY: Stony Brook University, Department of Computer Science.
148. Afzal, M. and Ashraf, S.A., 2016. Genetic algorithm for outlier detection. *Int. J. Comput. Sci. Inf. Technol.(IJCSIT)*, 7(2), pp.833-835.
149. Raja, P.V. and Bhaskaran, V.M., 2012. An effective genetic algorithm for outlier detection. *International Journal of Computer Applications*, 38(6), pp.30-33.
150. Desale, K. and Ade, R., 2015. Preprocessing of Streaming Data using Genetic Algorithm. *International Journal of Computer Applications*, 975, p.8887.
151. Vivekanandan, P. and Nedunchezian, R., 2011. Mining data streams with concept drifts using genetic algorithm. *Artificial Intelligence Review*, 36(3), pp.163-178.
152. Iwashita, A.S. and Papa, J.P., 2018. An overview on concept drift learning. *Ieee Access*, 7, pp.1532-1547.
153. Sharma, N. and Makhija, P., 2018. A Review on Optimizing Clustering Technique for Data Stream using Genetic Algorithm. *International Journal of Computer Sciences and Engineering*.
154. Man, K.F., Tang, K.S. and Kwong, S., 1996. Genetic algorithms: concepts and applications [in engineering design]. *IEEE transactions on Industrial Electronics*, 43(5), pp.519-534.

155. Eiben, A.E. and Smith, J.E., 2015. *Introduction to evolutionary computing*. Springer-Verlag Berlin Heidelberg.
156. Carr, J., 2014. An introduction to genetic algorithms. *Senior Project*, 1(40), p.7.
157. Mitchell, M., 1998. *An introduction to genetic algorithms*. MIT press.
158. Srinivas, M. and Patnaik, L.M., 1994. Genetic algorithms: A survey. *computer*, 27(6), pp.17-26.
159. Sivaraj, R. and Ravichandran, T., 2011. A review of selection methods in genetic algorithm. *International journal of engineering science and technology*, 3(5), pp.3792-3797.
160. Ochoa, G., Harvey, I. and Buxton, H., 2000, July. Optimal mutation rates and selection pressure in genetic algorithms. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation* (pp. 315-322).
161. Baker, J.E., 1987, July. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms* (Vol. 206, pp. 14-21).
162. Baker, J.E., 1985, July. Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications* (pp. 101-111).
163. Ponsich, A., Azzaro-Pantel, C., Domenech, S. and Pibouleau, L., 2008. Constraint handling strategies in genetic algorithms application to optimal batch plant design. *Chemical Engineering and Processing: Process Intensification*, 47(3), pp.420-434.
164. Magalhaes-Mendes, J., 2013. A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS transactions on computers*, 12(4), pp.164-173.
165. Abdoun, O., Abouchabaka, J. and Tajani, C., 2012. Analyzing the performance of mutation operators to solve the travelling salesman problem. *arXiv preprint arXiv:1203.3099*.
166. Soni, N. and Kumar, T., 2014. Study of various mutation operators in genetic algorithms. *International Journal of Computer Science and Information Technologies*, 5(3), pp.4519-4521.
167. Cervantes, J. and Stephens, C.R., 2006, July. " Optimal" mutation rates for genetic search. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 1313-1320).
168. Simulation of Genetic based Incremental Local Outlier Factor. Available online: <https://www.youtube.com/watch?v=YY-IHhhe2Ew&t=15s>
169. Deckert, M., 2013. Incremental rule-based learners for handling concept drift: an overview. *Foundations of Computing and Decision Sciences*, 38(1), pp.35-65.
170. Žliobaitė, I., Pechenizkiy, M. and Gama, J., 2016. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pp.91-114.
171. Hanley, J.A. and McNeil, B.J., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1), pp.29-36.
172. Bradley, A.P., 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7), pp.1145-1159.

173. Aggarwal, C.C. and Sathe, S., 2015. Theoretical foundations and algorithms for outlier ensembles. *Acm sigkdd explorations newsletter*, 17(1), pp.24-47.
174. Yamanishi, K., Takeuchi, J.I., Williams, G. and Milne, P., 2004. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3), pp.275-300.
175. <https://github.com/olmallet81/GALGO-2.0>
176. <https://github.com/xgmachina/GILOF>
177. Syarif, I., 2014. *Comprehensive review of classification algorithms for high dimensional datasets* (Doctoral dissertation, University of Southampton).
178. Yan, Y., Cao, L., Kulhman, C. and Rundensteiner, E., 2017, August. Distributed local outlier detection in big data. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1225-1234).
179. Yan, Y., Cao, L. and Rundensteiner, E.A., 2017, August. Scalable top-n local outlier detection. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1235-1244).
180. Alsini, R. and Ma, X., 2019. Data Streaming. *Encyclopedia of Big Data*; Schintler, L., McNeely, C., Eds.
181. Alghushairy, O., Alsini, R., Ma, X. and Soule, T., 2020, March. A Genetic-Based Incremental Local Outlier Factor Algorithm for Efficient Data Stream Processing. In *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis* (pp. 38-49).
182. Azeez, N.A., Bada, T.M., Misra, S., Adewumi, A., Van der Vyver, C. and Ahuja, R., 2020. Intrusion detection and prevention systems: an updated review. *Data Management, Analytics and Innovation*, pp.685-696.
183. Sahani, R., Rout, C., Badajena, J.C., Jena, A.K. and Das, H., 2018. Classification of intrusion detection using data mining techniques. In *Progress in computing, analytics and networking* (pp. 753-764). Springer, Singapore.
184. Siddiqui, M.K. and Naahid, S., 2013. Analysis of KDD CUP 99 dataset using clustering based data mining. *International Journal of Database Theory and Application*, 6(5), pp.23-34.
185. Tan, S.C., Ting, K.M. and Liu, T.F., 2011, June. Fast anomaly detection for streaming data. In *Twenty-Second International Joint Conference on Artificial Intelligence*.
186. Raghunath, B.R. and Mahadeo, S.N., 2008, July. Network intrusion detection system (NIDS). In *2008 First International Conference on Emerging Trends in Engineering and Technology* (pp. 1272-1277). IEEE.
187. Auskalis, J., Paulauskas, N. and Baskys, A., 2018. Application of local outlier factor algorithm to detect anomalies in computer network. *Elektronika ir Elektrotechnika*, 24(3), pp.96-99.

188. Ding, T., Zhang, M. and He, D., 2017, July. A Network Intrusion Detection Algorithm Based on Outlier Mining. In *International Conference in Communications, Signal Processing, and Systems* (pp. 1229-1236). Springer, Singapore.
189. Agrawal, S. and Agrawal, J., 2015. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60, pp.708-713.
190. Goren, H.G., Tunali, S. and Jans, R., 2010. A review of applications of genetic algorithms in lot sizing. *Journal of Intelligent Manufacturing*, 21(4), pp.575-590.
191. Alghushairy, O., Alsini, R., Ma, X. and Soule, T., 2020, July. Improving the Efficiency of Genetic based Incremental Local Outlier Factor Algorithm for Network Intrusion Detection. In *Proceedings of the 4th International Conference on Applied Cognitive Computing, Las Vegas, NV, USA* (pp. 27-30).

## Appendix A – Experiment’s performance results with different hyperparameters

The following tables represents the performance results of the algorithms based on certain windows size. For GILOF and GILOF\_NS, each table shows the experiment’s performance results with different hyperparameters.

GILOF Performance in Normalized UCI Vowel Dataset													
<i>PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.</i>													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
<b>W 100</b>	77.3	72.3	70.8	73.4	70.1	70.6	71.8	71.6	74	73.5		72.54	
<b>W 120</b>	80.4	83.4	83.4	81.7	82.5	80	79.2	79.6	81.8	80.6		81.26	
<b>W 140</b>	87.8	86.7	85.3	84	85.1	85.3	87.2	85.5	85.7	87.6		86.02	
<b>W 160</b>	87.4	86.1	88.1	87.3	87.2	88.2	88	88.8	88.6	87.6		87.73	
<b>W 180</b>	89.7	89.7	88.6	89.7	90.1	89.8	89.4	90.5	89.6	89.1		89.62	
<b>W 200</b>	90.8	90.7	91.2	90.5	90.7	91.5	89.8	90	90.2	91.1		90.65	
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.074	0.076	0.071	0.074	0.074	0.073	0.073	0.073	0.073	0.071		0.0729	0.729
<b>W 120</b>	0.082	0.085	0.082	0.081	0.084	0.082	0.082	0.083	0.082	0.084		0.0829	0.829
<b>W 140</b>	0.093	0.092	0.096	0.092	0.093	0.089	0.089	0.09	0.09	0.092		0.0917	0.917
<b>W 160</b>	0.107	0.103	0.106	0.105	0.106	0.103	0.103	0.104	0.103	0.104		0.1045	1.045
<b>W 180</b>	0.118	0.117	0.118	0.117	0.115	0.114	0.114	0.117	0.115	0.114		0.116	1.16
<b>W 200</b>	0.129	0.124	0.129	0.127	0.128	0.124	0.124	0.124	0.124	0.123		0.1258	1.258

GILOF Performance in Unnormalized UCI Vowel Dataset													
<i>PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.</i>													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
<b>W 100</b>	72.8	75.1	74.9	74.1	76.5	73.1	74.7	72.1	75.4	73.4		74.21	
<b>W 120</b>	83.6	84.2	82.3	84.4	84.2	82.1	83.2	80.8	82.8	82		82.96	
<b>W 140</b>	87.4	86	87.3	85.9	86.1	87.8	85.6	85.6	87.3	83.3		86.23	
<b>W 160</b>	88.3	87.5	85.9	87.4	86.8	86.6	86.9	86.2	86.7	88.3		87.06	
<b>W 180</b>	90	89.4	89.8	88.9	88.8	88.6	90.1	89.3	89.7	90.1		89.47	
<b>W 200</b>	89.9	89.2	89.1	89.5	90.6	90.1	89.7	89.2	89.2	89.3		89.58	
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.074	0.073	0.073	0.069	0.071	0.073	0.074	0.068	0.073	0.073		0.0721	0.721
<b>W 120</b>	0.085	0.082	0.079	0.08	0.083	0.082	0.082	0.082	0.084	0.081		0.082	0.82
<b>W 140</b>	0.092	0.092	0.092	0.092	0.091	0.092	0.092	0.092	0.9	0.093		0.1728	1.728
<b>W 160</b>	0.104	0.106	0.104	0.103	0.104	0.102	0.103	0.104	0.105	0.104		0.1039	1.039
<b>W 180</b>	0.117	0.115	0.117	0.115	0.117	0.116	0.115	0.115	0.115	0.115		0.1157	1.157
<b>W 200</b>	0.129	0.124	0.125	0.135	0.126	0.124	0.126	0.124	0.124	0.126		0.1263	1.263

GILOF_NS Performance in Normalized UCI Vowel Dataset													
PS=2. NG=4. One Point Crossover=0.7. BDM Mutation=0.07. Selection= RWS.													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	75.2	74.9	72.7	76.5	73.9	72	72.6	75.7	74.4	76.5		74.44	
W 120	83.3	81.3	81.5	82.7	84.2	83.5	82.2	82.1	82.9	82.4		82.61	
W 140	87.4	86.7	84.8	88.6	84.7	87.8	88	86.4	86.9	86.3		86.76	
W 160	87.6	88	87.5	88.1	87.9	88.3	88.1	87.6	88.1	87.8		87.9	
W 180	91.2	90.7	90.8	91.1	89.8	91.3	90.3	90	90	90.9		90.61	
W 200	92.1	92.2	92.5	92.5	92.5	92.1	93	92.9	91.3	92.4		92.35	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.082	0.081	0.079	0.079	0.078	0.079	0.079	0.078	0.08	0.081		0.0796	0.796
W 120	0.095	0.09	0.09	0.09	0.09	0.089	0.092	0.089	0.09	0.089		0.0904	0.904
W 140	0.101	0.1	0.099	0.099	0.099	0.099	0.101	0.099	0.098	0.099		0.0994	0.994
W 160	0.115	0.111	0.11	0.112	0.114	0.112	0.112	0.112	0.114	0.114		0.1126	1.126
W 180	0.124	0.124	0.126	0.124	0.124	0.124	0.124	0.124	0.124	0.126		0.1244	1.244
W 200	0.14	0.135	0.137	0.139	0.137	0.137	0.139	0.137	0.145	0.135		0.1381	1.381

GILOF_NS Performance in Unnormalized UCI Vowel Dataset													
PS=2. NG=4. One Point Crossover=0.7. BDM Mutation=0.07. Selection= TNT.													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	76.9	75.3	77	75.7	73.9	76.2	73.7	73.1	75.4	75.7		75.29	
W 120	82.9	81.6	82.1	82.6	83.5	82.7	82.2	84.1	80.6	81.1		82.34	
W 140	87.2	88.6	86.4	86.9	86.5	87.2	87.3	86	86	87.6		86.97	
W 160	88.1	87.4	87.3	87.5	88.3	87.8	87.9	87.9	87.4	87.5		87.71	
W 180	90.4	90	90.3	90.6	90.1	90.2	91.1	90.3	91	90.2		90.42	
W 200	91.9	92	91.8	92.2	91.9	91.8	91.7	91.8	91.8	92.1		91.9	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.084	0.081	0.081	0.079	0.081	0.081	0.081	0.081	0.081	0.081		0.0811	0.811
W 120	0.092	0.09	0.09	0.092	0.09	0.092	0.089	0.09	0.09	0.092		0.0907	0.907
W 140	0.101	0.101	0.101	0.103	0.101	0.101	0.098	0.101	0.101	0.101		0.1009	1.009
W 160	0.115	0.115	0.114	0.115	0.114	0.114	0.114	0.115	0.114	0.112		0.1142	1.142
W 180	0.128	0.127	0.126	0.124	0.123	0.126	0.124	0.126	0.128	0.124		0.1256	1.256
W 200	0.139	0.14	0.139	0.139	0.137	0.135	0.139	0.137	0.137	0.137		0.1379	1.379

DILOF Performance in Normalized UCI Vowel Dataset													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	69.1	70	73.7	72.8	65.9	67.7	68.8	68	70	69.1		69.51	
W 120	77.2	79.4	83.5	81.6	78.8	76.9	80.7	75.2	76.9	77		78.72	
W 140	83.4	85.5	84.8	85.7	83	84.4	83.3	84.8	84.6	84.5		84.4	
W 160	85.5	86.8	85.1	88.7	87.2	85.8	85.6	84.8	85.1	84.2		85.88	
W 180	88.5	87.3	88.3	90.1	88.4	88.4	89	89.3	89	89.9		88.82	
W 200	89.6	89.9	91.2	90.3	88.9	89.9	89.2	89.1	89.4	87.9		89.54	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.074	0.075	0.073	0.073	0.073	0.067	0.066	0.071	0.07	0.071		0.0713	0.713
W 120	0.082	0.084	0.085	0.084	0.083	0.08	0.081	0.083	0.081	0.08		0.0823	0.823
W 140	0.095	0.09	0.093	0.093	0.091	0.09	0.092	0.091	0.092	0.092		0.0919	0.919
W 160	0.1	0.104	0.104	0.104	0.101	0.102	0.099	0.101	0.103	0.103		0.1021	1.021
W 180	0.11	0.118	0.115	0.115	0.117	0.114	0.115	0.114	0.116	0.115		0.1149	1.149
W 200	0.12	0.13	0.13	0.13	0.129	0.126	0.128	0.124	0.126	0.128		0.1271	1.271



DILOF Performance in Unnormalized UCI Vowel Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC		
<b>W 100</b>	67.5	68.1	67.6	69.2	72.9	66.8	68.6	69.8	70.1	71.5	69.21		
<b>W 120</b>	73.1	71.8	77.8	71.8	80.2	77	77.3	73.7	74.4	78.1	75.52		
<b>W 140</b>	82.5	83.4	84.1	81.4	86.2	85.7	85	82.9	84.7	83.6	83.95		
<b>W 160</b>	85.1	85.8	84.2	84.8	87.5	85.1	85.2	86	85.7	85.7	85.51		
<b>W 180</b>	88.8	87.9	88.3	88.6	88.4	88.9	88.7	88.7	88.2	88.1	88.46		
<b>W 200</b>	88.6	88.9	88.2	88.7	90.2	89.1	88.6	88.3	88.6	88.2	88.74		
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.07	0.075	0.075	0.0751	0.073	0.074	0.07	0.07	0.07	0.068	0.07201	0.7201	
<b>W 120</b>	0.08	0.085	0.082	0.087	0.085	0.082	0.079	0.079	0.081	0.082	0.0822	0.822	
<b>W 140</b>	0.09	0.095	0.092	0.094	0.093	0.093	0.093	0.092	0.09	0.09	0.0922	0.922	
<b>W 160</b>	0.1	0.104	0.103	0.112	0.104	0.104	0.104	0.104	0.103	0.103	0.1041	1.041	
<b>W 180</b>	0.11	0.118	0.118	0.12	0.114	0.115	0.117	0.114	0.115	0.115	0.1156	1.156	
<b>W 200</b>	0.12	0.131	0.128	0.133	0.127	0.128	0.126	0.131	0.125	0.126	0.1275	1.275	

DILOF_NS Performance in Normalized UCI Vowel Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC		
<b>W 100</b>	71.3	75.1	68.8	72.5	71.4	67.1	68.9	69.3	72.4	69.8	70.66		
<b>W 120</b>	81.6	83	81.9	83.5	80.7	82.4	81.9	81.1	81.7	81.7	81.95		
<b>W 140</b>	84.5	87.2	86.6	87.1	83.9	85.5	86.5	83.9	85.8	84.7	85.57		
<b>W 160</b>	86.1	88.2	85.5	88.6	86.9	87.5	86.6	86.5	87.5	86.3	86.97		
<b>W 180</b>	88.9	90	88.9	89.1	89.9	88.3	88.9	87.6	88.3	89	88.89		
<b>W 200</b>	91.3	92	90.4	91.5	90.4	91.3	90.4	91.1	91.2	92	91.16		
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.074	0.075	0.073	0.073	0.073	0.067	0.066	0.071	0.07	0.071	0.0713	0.713	
<b>W 120</b>	0.082	0.084	0.085	0.084	0.083	0.08	0.081	0.083	0.081	0.08	0.0823	0.823	
<b>W 140</b>	0.095	0.09	0.093	0.093	0.091	0.09	0.092	0.091	0.092	0.092	0.0919	0.919	
<b>W 160</b>	0.1	0.104	0.104	0.104	0.101	0.102	0.099	0.101	0.103	0.103	0.1021	1.021	
<b>W 180</b>	0.11	0.118	0.115	0.115	0.117	0.114	0.115	0.114	0.116	0.115	0.1149	1.149	
<b>W 200</b>	0.12	0.13	0.13	0.13	0.129	0.126	0.128	0.124	0.126	0.128	0.1271	1.271	

DILOF_NS Performance in Unnormalized UCI Vowel Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC		
<b>W 100</b>	74.3	73.9	73.7	70.8	73.9	72.8	69.9	71.7	69.6	70.3	72.09		
<b>W 120</b>	82.9	82.8	80.6	80.2	80.3	81.2	81.4	80.3	80.6	81.3	81.16		
<b>W 140</b>	85.1	85.6	87.5	86.5	84.8	83.4	85.6	83.4	84.8	84	85.07		
<b>W 160</b>	86.5	87.9	84.7	87	86.5	86.4	87.1	85.7	86.8	86.1	86.47		
<b>W 180</b>	89.7	88.9	90	89.1	89	89.3	89.7	89.3	88.9	88.8	89.27		
<b>W 200</b>	91.1	90.9	92.6	91.2	90.5	90.9	90.9	90.6	91.3	91.1	91.11		
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.082	0.08	0.081	0.081	0.082	0.082	0.074	0.074	0.076	0.074	0.0786	0.786	
<b>W 120</b>	0.093	0.092	0.091	0.092	0.091	0.09	0.089	0.087	0.089	0.089	0.0903	0.903	
<b>W 140</b>	0.103	0.103	0.102	0.102	0.102	0.099	0.098	0.101	0.099	0.098	0.1007	1.007	
<b>W 160</b>	0.114	0.113	0.114	0.114	0.114	0.11	0.112	0.11	0.109	0.112	0.1122	1.122	
<b>W 180</b>	0.128	0.128	0.126	0.128	0.128	0.124	0.124	0.124	0.124	0.126	0.126	1.26	
<b>W 200</b>	0.142	0.142	0.141	0.143	0.143	0.14	0.139	0.14	0.137	0.139	0.1406	1.406	

GILOF Performance in Normalized UCI Pendgit Dataset													
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= TNT.													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	95.9	95.1	95.2	95.4	95.6	95.6	95.9	95.5	95.3	95.1		95.46	
W 120	97.6	97.1	97.6	97.5	97.5	97.4	97.7	97.4	97.8	97.6		97.52	
W 140	97.9	98.2	98	98	98.1	98	98.1	98.2	98.1	97.9		98.05	
W 160	98.7	98.8	98.6	98.7	98.5	98.7	98.7	98.6	98.7	98.7		98.67	
W 180	99	98.9	98.8	99	98.9	98.8	98.9	98.8	98.9	98.8		98.88	
W 200	98.3	98.5	98.4	98.5	98.4	98.1	98.5	98.5	98.4	98.6		98.42	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.198	0.194	0.195	0.195	0.196	0.196	0.194	0.192	0.195	0.193		0.1948	1.95
W 120	0.225	0.222	0.223	0.223	0.223	0.223	0.224	0.223	0.223	0.224		0.2233	2.23
W 140	0.249	0.247	0.248	0.247	0.248	0.249	0.247	0.248	0.247	0.248		0.2478	2.47
W 160	0.282	0.281	0.282	0.282	0.282	0.281	0.282	0.283	0.283	0.285		0.2823	2.82
W 180	0.316	0.315	0.317	0.316	0.317	0.317	0.314	0.315	0.315	0.315		0.3157	3.15
W 200	0.349	0.348	0.35	0.348	0.347	0.346	0.35	0.346	0.346	0.347		0.3477	3.47

GILOF Performance in Unnormalized UCI Pendgit Dataset													
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	51.4	52.1	50.1	49.6	51.8	51.8	50.1	51.2	50.1	50.4		50.86	
W 120	49.7	49.2	49.8	47.6	49.1	50	49.7	51.2	50.3	50.9		49.75	
W 140	51.2	51.7	50.4	50.9	49.8	50.5	50.4	51.1	50.3	50		50.63	
W 160	49.9	50.9	51	51.3	51	51.9	50.7	51.9	50.3	49.2		50.81	
W 180	51.1	50.9	51.8	49.5	50.4	51	49.9	52.4	49.1	50.3		50.64	
W 200	52.9	54.1	53	52.4	51.5	52.7	53	53.2	52.7	53.1		52.86	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.199	0.193	0.193	0.194	0.201	0.195	0.194	0.193	0.193	0.194		0.1949	1.949
W 120	0.224	0.221	0.221	0.222	0.228	0.222	0.221	0.222	0.221	0.222		0.2224	2.224
W 140	0.242	0.242	0.243	0.249	0.251	0.246	0.246	0.247	0.245	0.246		0.2457	2.45
W 160	0.282	0.283	0.282	0.281	0.282	0.282	0.282	0.282	0.281	0.281		0.2818	2.81
W 180	0.315	0.317	0.32	0.314	0.315	0.314	0.312	0.312	0.313	0.313		0.3145	3.14
W 200	0.348	0.348	0.35	0.349	0.347	0.343	0.345	0.343	0.344	0.344		0.3461	3.46

Memory consumption for KDD Cup 99 http Dataset					Memory consumption for KDD Cup 99 smtp Dataset			
W Size	GILOF	GILOF_NS	DILOF	DILOF_NS	GILOF	GILOF_NS	DILOF	DILOF_NS
W 100	47.5 MB	47.5 MB	47.7 MB	47.7 MB	11.2 MB	11.2 MB	11.3MB	11.3MB
W 200	50 MB	50 MB	49.5 MB	49.5 MB	12.6 MB	12.6 MB	12.5 MB	12.5MB
W 300	51.7 MB	51.7 MB	51.2 MB	51.2 MB	15.1 MB	15.1 MB	14.8 MB	14.8 MB
W 400	52.6 MB	52.6 MB	52.8 MB	52.8 MB	16.6 MB	16.6 MB	16.4 MB	16.4 MB
W 1000	69.2 MB	69.2 MB	69.2 MB	69.2 MB	35.6 MB	35.6 MB	35.6 MB	35.6 MB

GILOF_NS Performance in Normalized UCI Pendigit Dataset													
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	95.8	95.2	95.3	95.7	95.6	95.6	96.2	95.7	95.4	95.1		95.56	
W 120	97.5	97.6	97.6	97.1	97.8	97.5	97.1	97.8	97.2	97.5		97.47	
W 140	98.3	98.1	98.4	98	98.3	97.9	98	98	98.2	98.3		98.15	
W 160	98.5	98.9	98.7	98.7	98.6	98.7	98.7	98.6	98.8	98.7		98.69	
W 180	98.9	98.9	99	99	98.8	98.8	98.8	98.9	98.7	98.9		98.87	
W 200	98.3	98.4	98.3	98.6	98.5	98.4	98.4	98.2	98.4	98.6		98.41	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.199	0.195	0.195	0.195	0.196	0.193	0.193	0.193	0.197	0.195		0.1951	1.95
W 120	0.22	0.221	0.225	0.223	0.217	0.223	0.224	0.223	0.225	0.22		0.2221	2.23
W 140	0.247	0.248	0.252	0.243	0.248	0.246	0.248	0.246	0.248	0.248		0.2474	2.47
W 160	0.284	0.281	0.282	0.281	0.279	0.281	0.281	0.282	0.282	0.281		0.2814	2.81
W 180	0.315	0.313	0.321	0.312	0.313	0.313	0.313	0.312	0.315	0.312		0.3139	3.13
W 200	0.345	0.348	0.346	0.345	0.343	0.345	0.345	0.343	0.343	0.343		0.3446	3.44

GILOF_NS Performance in Unnormalized UCI Pendigit Dataset													
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	50.1	49.6	49.9	51.6	49.7	49.9	50	51	50.1	50.2		50.21	
W 120	50.5	50.5	49.1	50.7	50.7	48.6	51	51	48.8	49.7		50.06	
W 140	50.3	50.5	51.3	51.4	50.1	50.9	50.6	51.6	50.4	50.6		50.77	
W 160	52.4	51.3	50.2	50.8	52.4	49.5	51.9	52	51.2	51		51.27	
W 180	52	51.4	52.4	50.9	50.6	52.8	52.4	51.9	51.7	50.8		51.69	
W 200	52.2	52	51.2	51.8	52.9	51	53.3	51.4	51.6	52.3		51.97	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.199	0.195	0.194	0.194	0.195	0.195	0.194	0.195	0.199	0.195		0.1955	1.95
W 120	0.226	0.223	0.222	0.222	0.226	0.223	0.222	0.223	0.227	0.222		0.2236	2.23
W 140	0.247	0.245	0.246	0.246	0.243	0.253	0.247	0.248	0.247	0.246		0.2468	2.46
W 160	0.282	0.283	0.281	0.276	0.279	0.286	0.281	0.285	0.282	0.283		0.2818	2.81
W 180	0.315	0.315	0.313	0.317	0.312	0.313	0.314	0.314	0.314	0.315		0.3142	3.14
W 200	0.345	0.356	0.346	0.345	0.342	0.345	0.344	0.345	0.345	0.345		0.3458	3.45

DILOF Performance in Normalized UCI Pendigit Dataset													
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	94.5	94.8	95.4	94.5	94.3	94.2	94.3	94.5	94.8	94.6		94.59	
W 120	95.9	96.2	97.4	96.2	95.3	96	96.3	95.7	95.6	95.9		96.05	
W 140	96.3	96.1	98	96.6	96.5	97	96.6	96.3	96.5	96.7		96.66	
W 160	97.6	97.6	98.7	97.7	97.5	97.6	97.5	97.1	97.4	97.4		97.61	
W 180	98.1	98.1	99	97.5	98.1	97.6	98.1	98.1	97.5	97.7		97.98	
W 200	97.1	97.1	98.7	97	97.8	97.3	96.9	97.5	97.5	97.7		97.46	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	0.19	0.192	0.193	0.192	0.189	0.193	0.187	0.185	0.194	0.184		0.1899	1.899
W 120	0.218	0.219	0.22	0.218	0.216	0.218	0.221	0.22	0.218	0.218		0.2186	2.186
W 140	0.249	0.256	0.251	0.25	0.252	0.248	0.244	0.248	0.248	0.246		0.2492	2.492
W 160	0.282	0.291	0.287	0.285	0.282	0.282	0.281	0.274	0.281	0.277		0.2822	2.822
W 180	0.315	0.321	0.317	0.315	0.316	0.315	0.311	0.31	0.312	0.313		0.3145	3.145
W 200	0.353	0.353	0.355	0.347	0.35	0.345	0.349	0.346	0.348	0.348		0.3494	3.494

Memory Consumption for UCI Pendigit Dataset					Memory Consumption for UCI Vowel Dataset					
W Size	GILOF	GILOF_NS	DILOF	DILOF_NS	ILOF	GILOF	GILOF_NS	DILOF	DILOF_NS	ILOF
W 100	4.3 MB	4.3 MB	4.3 MB	4.3 MB	202 MB	3.8 MB	3.8 MB	3.7 MB	3.7 MB	36.7 MB
W 120	4.7 MB	4.7 MB	4.6 MB	4.6 MB		4.1 MB	4.1 MB	4.2 MB	4.2 MB	
W 140	5.1 MB	5.1 MB	4.9 MB	4.9 MB		4.8 MB	4.8 MB	4.6 MB	4.6 MB	
W 160	5.4 MB	5.4 MB	5.4 MB	5.4 MB		5 MB	5 MB	4.9 MB	4.9 MB	
W 180	5.7 MB	5.7 MB	5.7 MB	5.7 MB		5.1 MB	5.1 MB	5.1 MB	5.1 MB	
W 200	5.9 MB	5.9 MB	6 MB	6 MB		5.2 MB	5.2 MB	5.2 MB	5.2 MB	
W 1000	26.4 MB	26.4 MB	26.5 MB	26.5 MB		23.8 MB	23.8 MB	23.8 MB	23.8 MB	

DILof Performance in Unnormalized UCI Pendigit Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC		
<b>W 100</b>	49.8	50	51.2	50.3	51.1	52.4	50.9	51.8	50.4	51.7	50.96		
<b>W 120</b>	52.8	50.2	50.7	51.8	53.7	49.5	50.7	49.9	50.2	50.5	51		
<b>W 140</b>	51.8	52	53.4	53.5	54.3	52.5	51.7	51.1	49.5	52.9	52.27		
<b>W 160</b>	54.2	54.1	54.7	53.7	54.6	48.9	50.9	50.5	50.6	51.1	52.33		
<b>W 180</b>	52.2	54.4	54.5	53.3	53.4	50.6	50	51.6	50.8	50.9	52.17		
<b>W 200</b>	54.7	54	53	53.1	53.9	50.4	52.9	51.9	52.2	52.4	52.85		
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.19	0.185	0.186	0.187	0.183	0.192	0.185	0.187	0.191	0.188	0.1874	1.874	
<b>W 120</b>	0.212	0.218	0.2	0.193	0.19	0.221	0.217	0.218	0.216	0.217	0.2102	2.102	
<b>W 140</b>	0.247	0.235	0.234	0.245	0.246	0.249	0.242	0.24	0.242	0.242	0.2422	2.422	
<b>W 160</b>	0.283	0.284	0.274	0.284	0.284	0.282	0.277	0.276	0.283	0.282	0.2809	2.809	
<b>W 180</b>	0.291	0.32	0.318	0.318	0.316	0.318	0.312	0.313	0.31	0.314	0.313	3.13	
<b>W 200</b>	0.38	0.342	0.345	0.344	0.342	0.348	0.345	0.345	0.346	0.348	0.3485	3.485	

DILof NS Performance in Normalized UCI Pendigit Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC		
<b>W 100</b>	95.1	93.9	95.2	94.6	94	94.4	94.6	94.8	94.8	93.9	94.53		
<b>W 120</b>	95.8	95.9	97	95.7	95.6	96	95.8	95.5	96.6	95.9	95.98		
<b>W 140</b>	96.8	96.3	97.9	97.2	96	96.6	96.9	96.7	96.7	96.9	96.8		
<b>W 160</b>	97.5	97.2	98.5	97.4	96.9	97.6	97.9	97.3	97.5	97.9	97.57		
<b>W 180</b>	97.5	98	98.2	97.7	97.1	97.4	98.2	98	97.9	97.4	97.74		
<b>W 200</b>	96.8	97.6	98.7	97.1	96.4	97.3	97.3	97.3	97.1	97.2	97.28		
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.19	0.192	0.19	0.192	0.194	0.185	0.19	0.193	0.188	0.19	0.1904	1.904	
<b>W 120</b>	0.221	0.222	0.221	0.221	0.22	0.219	0.222	0.223	0.218	0.218	0.2205	2.205	
<b>W 140</b>	0.251	0.25	0.253	0.252	0.247	0.249	0.249	0.248	0.246	0.246	0.2491	2.491	
<b>W 160</b>	0.284	0.286	0.284	0.285	0.284	0.282	0.28	0.279	0.271	0.277	0.2812	2.812	
<b>W 180</b>	0.317	0.322	0.315	0.317	0.316	0.309	0.313	0.31	0.31	0.306	0.3135	3.135	
<b>W 200</b>	0.353	0.367	0.354	0.352	0.352	0.349	0.346	0.346	0.346	0.348	0.3513	3.513	

DILof NS Performance in Unnormalized UCI Pendigit Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC		
<b>W 100</b>	51.2	49.7	51.4	50.6	50.6	50.6	50.6	51.8	50.3	49.8	50.66		
<b>W 120</b>	52.4	53.4	51.2	55.2	52.6	49	51.3	50.5	50.6	51.6	51.78		
<b>W 140</b>	51.7	49.8	52.2	52.2	51.3	49.3	50.9	50.5	52.6	50.6	51.11		
<b>W 160</b>	55.2	54.7	53.9	54.8	53.3	51.3	49.6	50.8	50.5	50.3	52.44		
<b>W 180</b>	53.2	54	54.3	54.9	55.9	49.5	52.8	51.7	50.9	51.5	52.87		
<b>W 200</b>	54	55.2	53.9	53.9	53.7	52.1	52.1	51.5	49.7	50.1	52.62		
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	0.187	0.186	0.185	0.19	0.192	0.189	0.184	0.182	0.187	0.186	0.1868	1.868	
<b>W 120</b>	0.217	0.213	0.21	0.219	0.217	0.218	0.22	0.219	0.217	0.218	0.2168	2.168	
<b>W 140</b>	0.243	0.241	0.24	0.247	0.247	0.243	0.24	0.249	0.237	0.245	0.2432	2.432	
<b>W 160</b>	0.275	0.273	0.275	0.277	0.274	0.28	0.278	0.276	0.277	0.278	0.2763	2.763	
<b>W 180</b>	0.313	0.313	0.317	0.316	0.323	0.313	0.316	0.312	0.312	0.313	0.3148	3.148	
<b>W 200</b>	0.344	0.346	0.347	0.346	0.344	0.342	0.346	0.345	0.346	0.347	0.3453	3.453	

GILOF Performance in Unnormalized KDD CUP99 SMTP Dataset												
PS=2. NG=4. Uniform Crossover=0.7. BDM Mutation=0.07. Selection= RWS.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	86.6	84.2	85.7	80.5	82.9	83	79	83.6	85.5	79.6	83.06	
W 200	89.2	84.5	85.1	84.5	83.8	87.3	82.9	85.6	87.8	84.5	85.52	
W 300	87.3	85.4	82.9	86.6	86	86.6	86.5	85.8	84.4	85.7	85.72	
W 400	86	87	84.6	86.5	85.4	87.8	88.4	84.8	89	88	86.75	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	4.6	4.58	4.58	4.46	4.58	4.52	4.62	4.58	4.59	4.58	4.569	45.69
W 200	8.43	8.22	8.17	9.06	8.5	8.17	8.26	8.25	8.14	8.19	8.339	83.39
W 300	12.81	12.55	12.5	12.86	12.52	12.56	12.46	12.49	12.47	12.61	12.583	125.83
W 400	17.99	17.48	17.74	17.64	18.41	17.44	17.48	17.53	17.51	17.62	17.684	176.84

GILOF_NS Performance in Normalized KDD CUP99 SMTP Dataset												
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	81.3	82.3	80.3	81.4	81	81.3	81.2	81.1	80.7	81.2	81.18	
W 200	83.9	85.4	86.5	82.5	86.5	85	86.2	86.2	82	87	85.12	
W 300	86.9	87.6	84.8	85.7	88	87.8	84.9	85.7	87.3	86.6	86.53	
W 400	86.6	84.8	86.3	86	84.6	83.9	84.9	84.6	87.5	85	85.42	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	4.66	4.71	4.71	4.7	4.7	4.71	4.7	4.71	4.66	4.7	4.696	46.96
W 200	8.16	8.16	8.16	8.17	8.16	8.18	8.16	8.15	8.17	8.15	8.162	81.62
W 300	12.59	12.57	12.53	12.53	12.56	12.58	12.55	12.51	12.54	12.55	12.551	125.51
W 400	17.61	17.5	17.45	17.47	17.5	17.55	17.54	17.42	17.57	17.49	17.51	175.1

GILOF_NS Performance in Unnormalized KDD CUP99 SMTP Dataset												
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= TNT.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	82.4	81.1	81.3	80.8	81.1	80.1	81.1	81.3	80.4	81.2	81.08	
W 200	84.3	88.1	87.8	86.3	84.1	84.4	87.1	84.1	88.2	88.2	86.26	
W 300	86.5	85.6	85.7	86.2	88.2	86.6	89.1	85.2	86	85.4	86.45	
W 400	87.4	85.8	84.9	85.1	86.6	83.6	87.5	86.1	86.9	88.2	86.21	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	4.22	4.19	4.27	4.18	4.2	4.2	4.29	4.27	2.36	4.27	4.045	40.45
W 200	8.05	8.05	8.28	8.02	7.97	8.32	7.96	7.88	7.84	8.31	8.068	80.68
W 300	12.28	12.29	12.46	12.37	12.32	12.57	12.49	12.44	12.19	12.29	12.37	123.7
W 400	17.23	17.34	17.58	17.33	17.22	17.25	17.32	17.41	17.07	17.76	17.351	173.51

DILOF Performance in Normalized KDD CUP99 SMTP Dataset												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	83.6	83.5	86.7	85.2	85.8	83.3	80.5	82.2	80	84.9	83.57	
W 200	89.5	85.6	84.3	85	85.5	89.2	86.1	85.8	86.6	87.2	86.48	
W 300	82.2	85.4	88.9	86.7	82.9	79	86	82.4	84.2	88	84.57	
W 400	83.9	82.7	82.9	80.4	81.9	82.4	85.7	81.1	82.5	82.4	82.59	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	4.31	4.31	4.3	4.34	4.28	4.34	4.28	4.29	4.29	4.31	4.305	43.05
W 200	8.37	8.35	9.66	8.63	8.4	9.08	8.26	8.32	8.66	8.22	8.595	85.95
W 300	13.02	13.01	14.27	13.27	13.12	12.81	13.17	12.66	12.63	12.66	13.062	130.62
W 400	17.84	18.12	19.03	18.48	18.79	18.68	17.84	17.96	17.78	18.08	18.26	182.6

DILOF Performance in Unnormalized KDD CUP99 SMTP Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
<b>W 100</b>	83.3	83.5	81.2	87.2	86.5	77.6	84.2	85.6	82.2	83.2		83.45	
<b>W 200</b>	86.2	87	87.6	82.2	85.7	88.1	87.3	85.6	85.2	85.9		86.08	
<b>W 300</b>	81.8	86.5	83.3	84.3	84.3	84.9	82.2	80.8	82.1	81.7		83.19	
<b>W 400</b>	81.2	87.9	83.1	79.6	84.3	84.3	81	79.3	81.8	85.9		82.84	
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	4.33	4.28	4.28	4.29	4.28	4.28	4.32	4.28	4.26	4.27		4.287	42.87
<b>W 200</b>	8.33	8.2	8.21	8.21	8.23	8.21	8.21	8.16	8.13	8.14		8.203	82.03
<b>W 300</b>	12.75	12.58	12.62	12.56	12.6	12.57	12.64	12.6	12.7	12.62		12.624	126.24
<b>W 400</b>	17.81	17.83	17.82	17.72	17.61	17.97	18.24	17.86	18.17	17.69		17.872	178.72

DILOF_NS Performance in Normalized KDD CUP99 SMTP Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
<b>W 100</b>	81.7	81.6	80.4	81.6	79.7	81.9	81.6	80.7	81.7	82.6		81.35	
<b>W 200</b>	88.4	89.8	87.7	88.2	88.1	82.8	87.7	86.4	87.9	83		87	
<b>W 300</b>	87	88.4	88.7	87.7	88.5	88.2	86.9	86.9	88.4	89.5		88.02	
<b>W 400</b>	86.2	85.3	88.5	86.7	89.2	85.1	88	87.5	85.3	89.6		87.14	
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	4.41	4.36	4.33	4.31	4.43	4.37	4.38	4.37	4.37	4.37		4.37	43.7
<b>W 200</b>	8.83	8.72	8.57	8.38	8.77	8.28	8.3	8.29	8.28	8.3		8.472	84.72
<b>W 300</b>	13.1	13.74	13.21	13.19	13.14	12.74	12.75	12.76	14.67	12.81		13.211	132.11
<b>W 400</b>	18.88	18.65	18.93	18.18	19.12	17.85	17.81	17.96	18.05	17.78		18.321	183.21

DILOF_NS Performance in Unnormalized KDD CUP99 SMTP Dataset													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
<b>W 100</b>	79.6	81.1	82	79.9	81.4	83.7	81.7	81.5	80	81.3		81.22	
<b>W 200</b>	82.9	88.6	83.3	88.7	88.1	86.5	83.3	82.2	85.5	86.9		85.6	
<b>W 300</b>	90	87.8	87.1	88.3	87.6	86.2	89.8	87.7	88.2	86.5		87.92	
<b>W 400</b>	86.8	84.3	83.1	85	83	88.4	83.7	88.5	86.7	85.6		85.51	
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	4.38	4.37	4.38	4.38	4.38	4.35	4.35	4.36	4.36	4.37		4.368	43.68
<b>W 200</b>	8.23	8.26	8.27	8.26	8.25	8.24	8.25	8.25	8.24	8.26		8.251	82.51
<b>W 300</b>	12.68	12.74	12.63	12.57	12.69	12.73	12.63	12.68	12.66	12.68		12.669	126.69
<b>W 400</b>	17.78	17.99	17.82	17.81	17.67	17.8	17.6	17.66	17.79	17.71		17.763	177.63

GILOF Performance in Normalized KDD CUP99 HTTP Dataset													
<i>PS=2, NG=4, Two Points Crossover=0.7, BDM Mutation=0.07, Selection= RWS.</i>													
<i>W Size</i>	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
<b>W 100</b>	90.6	90.6	88.7	90.6	89.4	86.7	89	90.6	90	90.6		89.68	
<b>W 200</b>	91.7	91.5	91.4	92	91.5	91.2	91.4	91.5	91	91.6		91.48	
<b>W 300</b>	91.3	89.8	90	90.7	90.8	90.2	90.6	90	90.3	91.1		90.48	
<b>W 400</b>	89.6	90.1	90.5	89.3	90.3	90.8	90.9	90.6	89.6	90.7		90.24	
<i>W Size</i>	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
<b>W 100</b>	26.53	26.47	26.41	26.61	26.347	26.46	27.48	26.46	26.68	26.43		26.5877	265.877
<b>W 200</b>	47.81	47.42	47.19	47.16	47.31	47.11	47.68	47.28	47.6	47.28		47.384	473.84
<b>W 300</b>	73.8	72.84	73.09	73.08	72.79	72.85	73.41	72.53	76.49	72.65		73.353	733.53
<b>W 400</b>	103.57	102.21	103.83	103.75	102.22	102.2	103.7	102.37	103.44	101.23		102.852	1028.52

GILOF_NS Performance in Normalized KDD CUP99 HTTP Dataset												
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	67.5	68.8	67.9	67.5	67.7	69.9	68.2	67.5	67.9	68.8	68.17	
W 200	78.8	78.9	78.6	78	78.5	77.8	77.7	78.3	78	79.3	78.39	
W 300	80.4	80.1	80.3	81	80.9	80.7	81.6	80.8	80.9	81.1	80.78	
W 400	79	79.6	79.2	79.4	79.9	79.8	79.4	79.6	79.5	79.3	79.47	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	27.86	27.62	27.47	27.71	27.79	27.63	27.6	27.63	27.54	27.65	27.65	276.5
W 200	48.73	48.64	48.51	48.8	48.47	51.34	48.69	48.6	49.17	48.94	48.989	489.89
W 300	74.85	74.59	74.65	74.91	75.39	77.5	75.22	74.79	75.87	75.42	75.319	753.19
W 400	105.12	104.99	104.78	104.88	105.15	108.74	108.06	105.35	106.63	105.61	105.931	1059.31

GILOF_NS Performance in Unnormalized KDD CUP99 HTTP Dataset												
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	34.8	34.5	34.8	35	34.5	34.8	36.7	34.8	35.1	33.9	34.89	
W 200	39.2	42.4	40.1	41.4	41.3	39.7	41.3	41.5	41.6	40.9	40.94	
W 300	44.3	42.7	44.6	47	46.1	46.1	44.7	49.6	46.4	45.9	45.74	
W 400	53.7	52	53.6	53.2	52.4	52.1	53.2	51.5	52.2	52	52.59	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	27.86	27.74	27.7	27.9	27.62	27.74	27.76	27.78	27.96	27.83	27.789	277.89
W 200	48.9	48.92	48.7	48.72	48.64	48.81	48.67	48.71	48.79	49.1	48.796	487.96
W 300	74.83	74.72	75.69	74.92	74.59	74.45	75.02	74.93	74.89	74.95	74.899	748.99
W 400	104.53	104.62	104.02	105.66	104.54	104.51	104.44	104.94	104.29	104.48	104.603	1046.03

DILOF Performance in Normalized KDD CUP99 HTTP Dataset												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	89.7	91	89.5	90.1	91.5	90.2	88.6	90.1	90.4	90	90.11	
W 200	92.6	93.3	92.9	93	93.2	92.3	92.2	92.7	92.3	92.7	92.72	
W 300	91.3	91.9	92	92.2	91.8	91.7	92.1	91.8	91.7	93.1	91.96	
W 400	90.3	90.6	90.4	91.2	91.4	89.7	91.3	90.3	90.7	91	90.69	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	24.8	25.37	25.19	25.62	25.54	25.51	25.34	25.57	25.56	25.38	25.388	253.88
W 200	48.5	49.94	49.32	49.39	49.26	50.92	49.45	49.09	49.35	49.4	49.462	494.62
W 300	75.7	75.78	75.02	75.4	75.52	77.97	77.3	75.4	75.26	75.11	75.846	758.46
W 400	109.6	106.93	109.63	104.84	108.86	107.66	106.13	106.55	106.57	106.23	107.3	1073

DILOF Performance in Unnormalized KDD CUP99 HTTP Dataset												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	76.4	76.3	77.9	78.8	77	79.3	77.3	79.1	77.5	77.9	77.75	
W 200	78.7	80.2	79.7	81.1	80	79.5	79.6	79.5	80.2	79.8	79.83	
W 300	77.6	77.2	77.5	79.1	77.8	77.7	77.5	78.8	78.9	78.1	78.02	
W 400	79.7	76.4	77.3	77.3	76.1	75.6	76.4	77	76.6	77.4	76.98	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	26.6	24.89	24.89	25.68	25.04	25.85	25.57	26.38	26.04	25.44	25.638	256.38
W 200	48.7	48.63	48.9	50.34	49.24	49.46	49.78	49.32	48.8	49.78	49.295	492.95
W 300	77.8	75.04	74.95	75.27	76.11	75.29	75.56	79.98	75.16	75.57	76.073	760.73
W 400	106.1	105.5	105.66	108.34	108.32	105.93	106.72	105.95	106.21	106.07	106.48	1064.8

DILOF_NS Performance in Unnormalized KDD CUP99 HTTP Dataset												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	35.3	34.4	34.4	35	34.4	34.3	35.1	34.8	36	34.5	34.82	
W 200	39.9	39.6	40	40	39.9	39.9	37.2	38.6	39.9	40.1	39.51	
W 300	45.6	47.2	49.3	45.4	46.9	46.9	46.3	47.23	43.9	46.9	46.563	
W 400	50.6	48.8	48.6	50.3	48.4	48.7	48.3	48.7	51.6	50.2	49.42	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	28.1	26.32	26.53	32.32	26.4	26.74	26.57	26.81	26.69	26.46	27.294	272.94
W 200	53.7	50.84	51.05	52.67	50.73	51.22	50.68	52.42	51.54	51.87	51.672	516.72
W 300	84.3	77.95	77.89	85.15	76.97	81.9	80.86	79.82	80.26	81.45	80.655	806.55
W 400	122.7	109.3	115.29	108.91	107.94	109.32	109.86	111.93	114.73	108.82	111.88	1118.8

GILOF_NS Performance in Unnormalized KDD CUP99 HTTP Dataset													
PS=5. NG=10. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RSP.													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	35.5	46.64	35	46.42	34.3	46.53	35.7	46.11	35.9	46.39	35.28	46.418	464.18
W 200	41.4	63.71	40.3	64.89	42.2	63.47	44.4	60.86	41.4	62.44	41.94	63.074	630.74
W 300	47.5	86.73	46.8	85.82	47	85.66	43.5	85.08	45.2	85.75	46	85.808	858.08
W 400	53.46	115.46	54.1	114.84	53.6	114.72	53.9	114.8	52.7	114.91	53.552	114.946	1149.46

GILOF_NS Performance in Normalized KDD CUP99 HTTP Dataset													
PS=5. NG=10. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	69	45.85	70	45.67	67.6	46.27	69	46.34	68.3	46.53	68.78	46.132	461.32
W 200	78.9	60.9	79.5	60.83	79.3	61.7	79.5	60.94	78.3	60.646	79.1	61.0032	610.032
W 300	81.2	85.23	80.8	85.35	81.6	85.33	81.7	85.1	81.4	85.37	81.34	85.276	852.76
W 400	79.5	114.91	78.6	115.17	79.8	115.2	79.1	115.45	79.5	115.23	79.3	115.192	1151.92

GILOF Performance in Unnormalized KDD CUP99 HTTP Dataset													
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.05. Selection= RNK.													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	76.6	26.69	76	26.72	75.4	26.61	76.5	26.72	75.6	26.62	76.02	26.672	266.72
W 200	76.4	47.57	75.5	47.45	76.6	47.65	75.9	47.38	75.3	47.44	75.94	47.498	474.98
W 300	76.1	76.02	75.6	73.16	76.8	72.96	76.9	73.47	76.5	73.31	76.38	73.784	737.84
W 400	76.4	104.39	74.9	103.88	75.6	103.29	74.7	103	76.6	103.56	75.64	103.624	1036.24

GILOF Performance in Normalized KDD CUP99 HTTP Dataset													
PS=2. NG=4. Two Points Crossover=0.8. Uniform Mutation=0.07. Selection= SUS.													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	88.5	26.69	89.7	26.42	89.5	26.5	89.1	26.94	89.3	26.6	89.22	26.63	266.3
W 200	91.9	47.21	91	48.59	92	47	91.4	48.67	91.9	47.04	91.64	47.702	477.02
W 300	90.9	72.92	90.1	73.59	91	72.62	90.8	72.49	91	72.71	90.76	72.866	728.66
W 400	90.3	102.26	90.3	102.34	90.1	101.98	90	102.51	90.4	102.24	90.22	102.266	1022.66

GILOF Performance in Normalized KDD CUP99 SMTP Dataset													
PS=4. NG=8. One Point Crossover=0.5. BDM Mutation=0.07. Selection= SUS.													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	85.6	6.27	85.1	6.32	83.2	6.31	87	6.32	86.2	6.3	85.42	6.304	63.04
W 200	82.6	9.31	85.8	9.11	84.3	9.15	85.1	9.2	84.1	9.19	84.38	9.192	91.92
W 300	83.9	14.89	86.8	13.38	85.8	13.4	87.1	13.39	85.4	13.45	85.8	13.702	137.02
W 400	82.7	19.93	86.2	18.47	84.2	18.43	83.6	19.43	84.3	18.53	84.2	18.958	189.58



GILOF Performance in Unnormalized KDD CUP99 SMTP Dataset													
<i>PS=5. NG=10. One Point Crossover=0.5. Uniform Mutation=0.07. Selection= TNT.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
<b>W 100</b>	82.2	7.65	85.8	7.75	84.7	7.46	85.9	7.65	83.3	7.67	84.38	7.636	76.36
<b>W 200</b>	82.6	10.18	84.4	10.47	86	10.05	85.5	10.22	88.3	10.13	85.36	10.21	102.1
<b>W 300</b>	86.5	14.22	87.1	14.48	86	14.12	86.1	14.1	85.3	14.56	86.2	14.296	142.96
<b>W 400</b>	84.8	19.18	84.5	19.55	88.8	19.2	84.7	19.43	88.6	19.48	86.28	19.368	193.68

GILOF_NS Performance in Normalized KDD CUP99 SMTP Dataset													
<i>PS=2. NG=6. Uniform Crossover=0.7. Uniform Mutation=0.07. Selection= RNK.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
<b>W 100</b>	85.1	4.81	82.4	4.79	82.9	4.8	83.2	4.8	83.5	4.8	83.42	4.8	48
<b>W 200</b>	86.4	8.41	84.6	8.27	87.1	8.22	85.1	8.24	84.4	8.22	85.52	8.272	82.72
<b>W 300</b>	85.5	12.86	85.5	12.53	87.3	12.59	86.2	12.55	87.6	12.52	86.42	12.61	126.1
<b>W 400</b>	87.3	17.7	84.5	17.49	84.4	17.55	85.4	17.5	84.4	17.42	85.2	17.532	175.32

GILOF Performance in Normalized KDD CUP99 SMTP Dataset													
<i>PS=2. NG=4. One Point Crossover=0.7. Uniform Mutation=0.07. Selection= TNT.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
<b>W 100</b>	83.2	4.58	83.8	4.61	82.5	4.68	82.1	4.68	86.5	4.67	83.62	4.644	46.44
<b>W 200</b>	84	8.21	85.8	8.1	88	8.12	84.4	8.21	83.3	8.13	85.1	8.154	81.54
<b>W 300</b>	86.5	12.53	86	12.4	87.3	12.43	86.2	12.51	86.6	12.41	86.52	12.456	124.56
<b>W 400</b>	86	17.52	86.4	17.36	82.2	17.44	85.8	17.42	89.3	17.31	85.94	17.41	174.1

GILOF Performance in Unnormalized UCI Pendgit Dataset													
<i>PS=5. NG=10. One Point Crossover=0.6. BDM Mutation=0.07. Selection= TNT.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
<b>W 100</b>	52.6	0.309	51.5	0.299	51.1	0.302	52.5	0.301	51.7	0.301	51.88	0.3024	3.024
<b>W 120</b>	50.7	0.321	49.7	0.315	51.8	0.316	49.9	0.312	50.5	0.312	50.52	0.3152	3.152
<b>W 140</b>	51.2	0.335	50.7	0.335	50.2	0.332	50.2	0.329	51.8	0.328	50.82	0.3318	3.318
<b>W 160</b>	50.4	0.363	50.5	0.371	49.5	0.37	50.8	0.357	49.7	0.36	50.18	0.3642	3.642
<b>W 180</b>	52	0.385	52.4	0.415	51.3	0.388	51.7	0.385	50.6	0.462	51.6	0.407	4.07
<b>W 200</b>	51.9	0.417	51.6	0.427	51.7	0.426	52.3	0.413	52	0.44	51.9	0.4246	4.246

GILOF_NS Performance in Unnormalized UCI Pendgit Dataset													
<i>PS=2. NG=5. Two Points Crossover=0.8 Uniform Mutation=0.08. Selection= SUS.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
<b>W 100</b>	50.5	0.202	49.1	0.199	50	0.198	50.8	0.198	50.4	0.198	50.16	0.199	1.99
<b>W 120</b>	49.9	0.228	49.8	0.226	49.8	0.226	50	0.226	50.7	0.226	50.04	0.2264	2.264
<b>W 140</b>	50	0.249	50.2	0.249	51.6	0.25	51	0.249	50.2	0.249	50.6	0.2492	2.492
<b>W 160</b>	51	0.283	52.2	0.283	52.2	0.283	53	0.283	51.4	0.283	51.96	0.283	2.83
<b>W 180</b>	51.1	0.315	51.8	0.315	52.1	0.315	52.4	0.315	50.8	0.315	51.64	0.315	3.15
<b>W 200</b>	51.3	0.348	53.1	0.354	52.4	0.347	50.6	0.347	52.1	0.356	51.9	0.3504	3.504

GILOF Performance in Normalized UCI Pendgit Dataset													
<i>PS=7. NG=14. One Point Crossover=0.5. Uniform Mutation=0.07. Selection= RNK.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
W 100	95.5	0.431	95.4	0.424	95.3	0.425	95.1	0.424	95.9	0.424	95.44	0.4256	4.256
W 120	97.1	0.43	97.5	0.424	97.3	0.425	97.1	0.424	97.5	0.424	97.3	0.4254	4.254
W 140	98.4	0.432	98.1	0.428	98.4	0.427	98.2	0.429	98.3	0.427	98.28	0.4286	4.286
W 160	98.5	0.456	98.6	0.452	98.7	0.459	98.5	0.451	98.7	0.45	98.6	0.4536	4.536
W 180	98.9	0.472	98.9	0.427	98.8	0.474	99	0.472	98.7	0.472	98.86	0.4634	4.634
W 200	98.4	0.497	98.2	0.496	98.4	0.498	98.4	0.497	98.4	0.498	98.36	0.4972	4.972

GILOF Performance in Normalized UCI Pendgit Dataset													
<i>PS=2. NG=4. Uniform Crossover=0.6. Uniform Mutation=0.08. Selection= RSP.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	95.7	0.206	95.6	0.201	95.6	0.201	95.5	0.201	95.3	0.202	95.54	0.2022	2.022
W 120	97.8	0.232	96.9	0.23	97.4	0.23	97.2	0.229	97.3	0.23	97.32	0.2302	2.302
W 140	98.3	0.256	98.1	0.255	98.2	0.255	98.1	0.254	98.3	0.254	98.2	0.2548	2.548
W 160	98.7	0.291	98.5	0.29	98.8	0.291	98.6	0.29	98.7	0.29	98.66	0.2904	2.904
W 180	98.8	0.324	98.8	0.326	98.8	0.325	98.9	0.324	98.8	0.323	98.82	0.3244	3.244
W 200	98.5	0.357	98.3	0.358	98.1	0.356	98.2	0.357	98.3	0.355	98.28	0.3566	3.566

GILOF Performance in Unnormalized UCI Vowel Dataset													
<i>PS=2. NG=4. One Point Crossover=0.7. BDM Mutation=0.07. Selection= RSP.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	73.5	0.078	76.1	0.073	75.1	0.073	74.1	0.074	73.9	0.073	74.54	0.0742	0.742
W 120	79.3	0.085	84.5	0.082	83.6	0.084	84.4	0.084	82.4	0.082	82.84	0.0834	0.834
W 140	85.3	0.095	85.7	0.093	85.7	0.095	85.2	0.094	84.1	0.095	85.2	0.0944	0.944
W 160	86.9	0.107	86.4	0.106	87.9	0.104	88.5	0.103	88.6	0.104	87.66	0.1048	1.048
W 180	88.9	0.118	90	0.117	89.6	0.115	89.8	0.118	90.4	0.115	89.74	0.1166	1.166
W 200	88.9	0.128	89.4	0.129	89.2	0.129	90.5	0.128	89.9	0.128	89.58	0.1284	1.284

GILOF_NS Performance in Unnormalized UCI Vowel Dataset													
<i>PS=2. NG=6. Uniform Crossover=0.5. BDM Mutation=0.05. Selection= RWS.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	TIME	AVG - AUC	AVG - Time	Total - Time
W 100	73.8	0.859	74.5	0.082	74.2	0.816	75.8	0.082	76.4	0.082	74.94	0.3842	3.842
W 120	84.2	0.842	81.5	0.093	83.5	0.906	82.8	0.095	80.1	0.095	82.42	0.4062	4.062
W 140	86.9	0.101	86.5	0.104	85.4	0.104	87.7	0.103	87.8	0.104	86.86	0.1032	1.032
W 160	87.2	0.117	87.7	0.113	88.2	0.115	88.3	0.114	87.7	0.118	87.82	0.1154	1.154
W 180	90.1	0.129	89.8	0.129	90	0.131	90.7	0.128	91	0.129	90.32	0.1292	1.292
W 200	91.8	0.142	92.6	0.139	91.9	0.145	92.3	0.14	91.8	0.14	92.08	0.1412	1.412

GILOF Performance in Normalized UCI Vowel Dataset													
<i>PS=3. NG=6. One Point Crossover=0.7. BDM Mutation=0.07. Selection= TNT.</i>													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
W 100	74.8	0.087	71.4	0.082	75.5	0.079	74.3	0.082	76.5	0.082	74.5	0.0824	0.824
W 120	78.1	0.092	83.1	0.09	80.7	0.09	79.9	0.092	81.5	0.095	80.66	0.0918	0.918
W 140	84.8	0.101	87.6	0.098	86.6	0.098	87.2	0.099	84	0.098	86.04	0.0988	0.988
W 160	89.1	0.112	87.7	0.112	89	0.112	88.8	0.109	89.8	0.11	88.88	0.111	1.11
W 180	89.7	0.123	89.5	0.121	90.1	0.121	89.6	0.121	90.2	0.124	89.82	0.122	1.22
W 200	91.2	0.134	91	0.132	90.3	0.132	91	0.132	91.3	0.137	90.96	0.1334	1.334

GILOF Performance in Normalized UCI Vowel Dataset													
PS=100. NG=200. Two points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.													
W Size	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AUC	Time	AVG - AUC	AVG - Time	Total - Time
W 100	73	18.86	73.3	22.15	70.8	17.49	72.5	17.48	74.1	17.52	72.74	18.7	187
W 120	79.9	15.99	80	19.4	80.3	15.17	80	15.11	81.5	15.13	80.34	16.16	161.6
W 140	86.5	14.9	84.3	15.38	87.4	13.56	85.9	13.61	88.9	13.54	86.6	14.198	141.98
W 160	88.6	13.28	90.7	12.61	90.9	12.49	89.8	12.42	90.3	12.34	90.06	12.628	126.28
W 180	90.3	11.9	91.4	11.56	90.9	11.54	90.2	11.45	89.4	11.55	90.44	11.6	116
W 200	91.8	11.72	90.9	10.83	89.9	10.88	90.5	10.79	91.5	10.79	90.92	11.002	110.02

DILOF_NS Performance in Normalized KDD CUP99 HTTP Dataset												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	71	71.1	70.7	66.6	67.9	66.6	66	70.6	66.9	67.3	68.47	
W 200	79.7	79.8	79.1	80.3	80	79.7	79.7	80.4	80	79.4	79.81	
W 300	82.1	82.3	81.9	81.9	82.8	82.7	82.3	81.6	82.7	81.4	82.17	
W 400	79.7	80	79.5	80.3	80.6	80.5	80.3	79.9	80.5	80.1	80.14	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	26.4	26.97	27.32	27.12	27.05	27.01	26.92	26.93	28.11	27.58	27.141	271.41
W 200	50.8	51.48	51.45	51.45	51.47	51.36	51.65	51.6	51.63	51.38	51.427	514.27
W 300	79.2	77.86	78.17	84.38	77.94	77.79	77.96	78.17	78.08	78.34	78.789	787.89
W 400	108.3	108.37	108.05	108.52	108.71	108.86	109.22	108.82	107.75	107.53	108.413	1084.13

GILOF Performance in Unnormalized KDD CUP99 HTTP Dataset												
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	73.8	75.3	75.4	74.1	76.8	76.1	75.1	76	76.8	76.8	75.62	
W 200	76.8	75.5	76.3	76.3	76.4	75.2	75.8	76.3	75.6	76.6	76.08	
W 300	77.6	78.2	76.3	76.8	76.4	78.1	76.4	76.4	76.7	76	76.89	
W 400	77.7	76.5	76	75.2	76.7	78.4	77.2	77.6	76.5	77.3	76.91	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	26.59	26.56	26.76	26.41	26.64	26.47	26.7	27.52	26.72	26.74	26.711	267.11
W 200	47.57	47.71	48.94	47.79	47.54	48.61	48.13	47.42	47.78	47.76	47.925	479.25
W 300	73.26	73.3	73.06	73.44	75.26	73.92	75.98	73.37	73.28	73.5	73.837	738.37
W 400	103.63	101.73	102.41	102.5	103.93	101.75	103.57	103.1	102.35	101.73	102.67	1026.7

GILOF Performance in Normalized KDD CUP99 SMTP Dataset												
PS=2. NG=4. Two Points Crossover=0.7. BDM Mutation=0.07. Selection= RWS.												
W Size	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AUC	AVG of AUC	
W 100	83.5	85.4	85.4	86.4	85.4	85.4	85.5	88	86.6	85.4	85.7	
W 200	83.6	83.7	85.2	86.8	84.8	88.8	80	86.1	84	85.8	84.88	
W 300	88.6	86.6	85.8	85.9	86	84.9	86.5	85.9	88.6	86	86.48	
W 400	85	82.5	88	87.4	83	88.7	84.1	86.1	88.2	85.2	85.82	
W Size	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	AVG of Time	Total Time
W 100	4.57	4.66	4.66	4.6	4.65	4.66	4.67	4.66	4.66	4.66	4.645	46.45
W 200	8.58	8.1	8.1	8.06	8.08	8.06	8.14	8.07	8.11	8.06	8.136	81.36
W 300	12.53	12.45	12.4	12.45	12.44	12.44	12.43	12.39	12.44	12.45	12.442	124.42
W 400	17.38	17.3	17.33	17.4	17.36	17.43	17.28	17.39	17.4	17.25	17.352	173.52