A Grid Partition-based Local Outlier Factor for 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
Raed A. Alsini

Major Professor: Xiaogang Ma, Ph.D.
Committee Members: Terence Soule, Ph.D.; Frederick Sheldon, Ph.D.; Ahmed Ibrahim, Ph.D.
Department Administrator: Terence Soule, Ph.D.

May 2021

Authorization to Submit Dissertation

This dissertation of Raed A. Alsini, submitted for the degree of Doctor of Philosophy with a Major in Computer Science and titled "A Grid Partition-based Local Outlier Factor for 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:	_____ Xiaogang Ma, Ph.D.	Date: _____
Committee Members:	_____ Terence Soule, Ph.D.	Date: _____
	_____ Frederick Sheldon, Ph.D.	Date: _____
	_____ Ahmed Ibrahim, Ph.D.	Date: _____
Department Administrator:	_____ Terence Soule, Ph.D.	Date: _____

Abstract

Outlier detection is getting significant attention in the research field of big data. Detecting the outlier is important in various applications such as communication, finance, fraud detection, and network intrusion detection. Because of their unique characteristics, such as large volume and high velocity, data streams pose a challenge to traditional outlier detection methods. Local Outlier Factor (LOF) is one of the most appropriate techniques for determining outliers in the density-based method. However, it faces some challenges when dealing with the data stream. One issue is that LOF requires the entire dataset as well as the distance value to be stored in the computer memory. Another issue arises when a change occurs in the dataset, which necessitates a significant recalculation from the beginning. To address these issues, this dissertation proposes a new method for detecting local outliers in data streams called the Grid Partition-based Local Outlier Factor (GP-LOF). We improve the GP-LOF algorithm even further by adding another technique known as the Local Outlier Factor by Reachability Distance (LOFR). The improved algorithm is thus called the Grid-Partition-based Local Outlier Factor by Reachability Distance (GP-LOFR). We tested both GP-LOF and GP-LOFR with several benchmark datasets. They outperformed the Density Summarization Incremental Local Outlier Factor (DILOF) algorithm, which is the most representative algorithm in existing studies of data stream processing. We also worked with real-world datasets of concrete mixture. In that work, a new algorithm called the Isolation Forest based on a sliding window for the Local Outlier Factor (IFS-LOF) was developed. The IFS-LOF outperformed both LOF and LOF-Sliding Window (LOF-SW) in accuracy of the results. In summary, the three new algorithms GP-LOF, GP-LOFR, and IFS-LOF are the major contributions of this PhD research. All proposed algorithms work without any previous knowledge of data distributions and are capable to execute with limited computer memory. This PhD research makes a solid contribution to the field of local outlier detection in big data streams. In the near future, we will extend the developed algorithms by applying Evolution Computation (EC) methods to further improve the accuracy and reduce the execution time. Moreover, we will apply these algorithms to more real-world datasets.

Acknowledgements

All praise be to God, who has granted me the knowledge and ability to complete this dissertation. This mission could not have been accomplished without God's blessings.

I want to thank my major professor Prof. Marshall (Xiaogang) Ma. It has been a great experience to be a doctoral student under your guidance. I want to express my appreciation for all you have done to assist with my writing skills development, advance my knowledge, encourage me, and help me accomplish my goals. This dissertation will not be completed without your valuable assistance and ongoing encouragement. Also, I am grateful to Professor Terrance (Terry) Soule for his assistance and guidance. You have played an enormous part in the successful completion of this dissertation.

Moreover, I would like to thank two other committee members for their support and encouraging feedback on my dissertation: Prof. Fredrick Sheldon and Prof. Ahmed Ibrahim.

My heartfelt gratitude goes to the Department of Computer Science at the University of Idaho. It has been a wonderful place with such a pleasant working atmosphere. Also, this achievement would not have been possible without the constant help and support of all IDEA lab members at the University of Idaho.

I am grateful for the support from my sponsors, King Abdulaziz University and Saudi Arabian Cultural Mission (SACM), and the Kingdom of Saudi Arabia government to allow me pursue a doctoral degree. Also, I would like to thank the University of Idaho for offering me this opportunity to study at this university.

Dedication

I dedicate this Ph.D. dissertation to my parents, my wife, my daughter, my brothers, my sister, and the people who've supported me along with this scholarly study.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

"Data Streaming."In: Schintler L., McNeely C. (eds) Encyclopedia of Big Data. Springer, Cham.DOI: https://doi.org/10.1007/978-3-319-32001-4_324-1.

Alsini, R., Alghushairy, O. Almakrab, A., Soule, T. and Ma, X., 2021. Local Outlier Detection Techniques in Real-World Streaming Data Processing: A Literature Review (Under Review)

1.1 Outlier Detection

Outlier detection is a process by which outliers are distinguished from the rest of a dataset. Outliers arise during a procedure or as a result of a measuring error [1]. Outliers may occur because of human mistakes, technological or machine failures, noise, device change, dishonest behavior, etc. [2]. It is necessary to evaluate the data, which may include valuable information regarding different areas, including the identification of network intrusion detection, industries, healthcare, transportation, and many others. In machine learning and data mining, outlier detection methods have been commonly used to gather information to assist in decision-making in various domains by understanding the data's behavior to detect the outlier for cleaning the data [3,4].

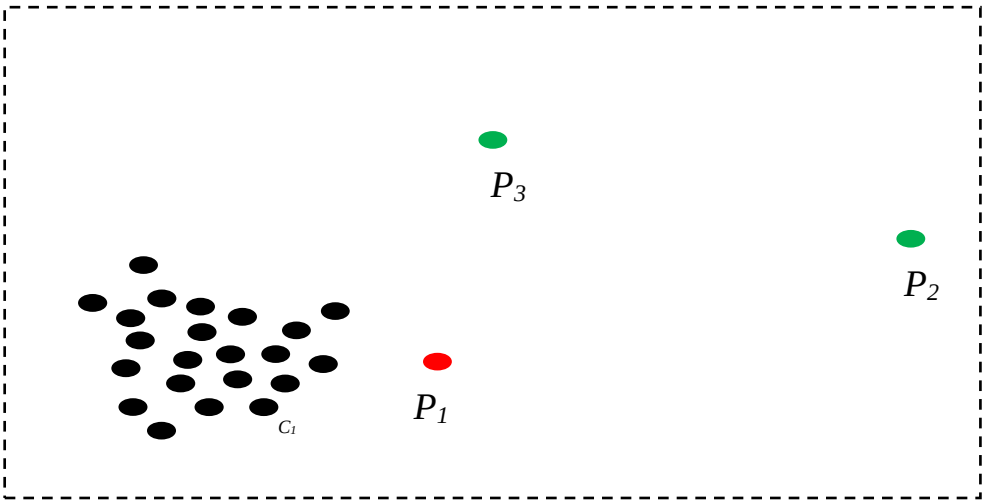


Figure 1.1 The categories of outlier in the two-dimensional space, where p2 and p3 are global outliers (green) and p1 are a local outlier (red).

Outliers occur in two forms: as a global outlier or as a local outlier [5]. If a global outlier, a data point that is far from the rest of the dataset is considered an outlier. If a local outlier, the outlier is based on the

distance between the points according to the k-Nearest-Neighbors (kNN) algorithm [6]. Figure 1.1 represents the difference between the global outlier and local outlier in a dataset. The local outlier factor (LOF) measures the amount of the density of data points and their nearby neighbors to assess local scores.

1.2 The Data Stream

Data has become an essential component of not only research but also of our daily lives. In the digital world, people are able to use various types of technology to collect and transmit big data, which has the features of overwhelming volume, velocity, variety, value, and veracity. More importantly, big data represents a vast amount of information and knowledge. The Internet of Things (IoT) is interconnected with big data. IoT applications use a data stream as a primary way for data transmission and to make a data stream a unique type of big data. A data stream is a sequence of data blocks that are transmitted. The real-time feature of the data stream requires corresponding technologies for efficient data processing. Streaming the data is built upon resources that are commonly used for communication, web activity, E-commerce, and social media. How the data is processed determines how information can be extracted from the data stream. Analyzing the data stream through queries ensures and improves the efficiency of data by the aspect of data science. Many techniques can be used in data stream processing, among which data mining is the most common approach used for detecting data latency, pattern frequency, and anomalous values, as well as for classification and clustering, and outlier detection. The computer science community has created many open-source libraries for data streams and has built various best practices to facilitate the applications of the data stream in different disciplines.

1.3 The Data Science Aspect of a Data Stream

In recent years, data has become the “crude oil” that drives technological and economic development. There is an extremely high demand, and almost everyone uses data. We need to refine crude oil before using it; it is the same with data. We can benefit from data only when data processing, mining, analysis, and extraction provide useful results. Using the data stream in data science involves understanding the data life cycle. Usually, it begins with collecting the data from its sources. Today, data stream collection

can be seen on search engines, in social media, on IoT devices, and in marketing. For instance, Google Trends generates a massive amount of data by searching certain topics on the web. Afterwards, it can provide results based on what a user is looking for within a specific range of time. The benefit of processing the data stream is getting the right information immediately. However, processing needs methods and models. Two standard models are batch processing and stream processing.

Batch processing can handle a large amount of data by first collecting the data over time and then doing processing it. For example, the operating system on a computer can optimize the sequencing of jobs to make efficient use of the system. Micro-batch is a modified model of batch processing. It groups data and tasks into small batches. Completing the processing of a batch in this model is based on how the next batch is received. Stream processing is a model used for processing the data without waiting for the next data to arrive. The benefit of stream processing is that the system can receive the data more quickly. For example, an online banking application runs stream processing when a customer buys a product. The bank transaction is then verified and executed without fail. Stream processing can handle a huge amount of data without suffering any issues related to data latency. A sensor network that generates massive data can be organized easily under this method.

Data mining, as a part of data science, is used to discover knowledge in data. Data stream mining usually involves methods, such as machine learning, to extract and predict new information. A few widely used methods are clustering, classification, and stream mining on the sensor network. Clustering is a process of gathering similar data into a group. Clustering deals with unsupervised learning. This means the system does not need to have a label in order to discover a hidden pattern in data. K-means is the most common method used for clustering. Clustering can be used for fraud detection; for example, it is able to find anomalous records on a credit card, after which the cardholder can be informed. Classification is a process of identifying the category of a piece of new data. Based on a set of training data, the system can set up several categories and then determine into which category a piece of new data belongs. Classification is one of the supervised learning methods in which the system learns and determines how

to make the right decision. For example, buying and selling holds on the stock market can be done using this method in order to make the right decisions based on the given data.

1.4 The Data Stream Management System (DSMS)

Regardless of what precedes the data stream and how it is stored, data management is required in the data life cycle. Managing the data stream can be done using queries as a primary method, such as the structured query language (SQL). SQL is a common language used for managing the database. The data stream management system (DSMS) uses an extended version of SQL, known as the continuous query language (CQL). The reason for the use of CQL is to ensure any continuous data over time can be used on the system. The operations of CQL can be categorized into three groups: relation-to-relation, stream-to-relation, and relation-to stream [7]. Relation-to-relation is usually done with a SQL query. For instance, the relation between two queries can be expressed by using either equal, above, greater, or less symbols. Stream-to-relation is accomplished using the sliding window method. The sliding window method is based on having a window that has historical points when the data is streamed. Specifically, when there are two window sizes, the second window will not begin until the difference between the windows is removed. Relation-to-stream usually involves the tree method to deal with the continuous query. Detailed operations include insertion, deletion, and relation.

1.5 Stream Reasoning

Stream reasoning is about processing the data stream to get a conclusion or decision on continuous information. Stream reasoning handles the continuous information by defining factors on the velocity, volume, and variety of big data. For example, a production company might use several sensors to estimate and predict the types and amounts of raw materials needed for each day. Another example is the detection of fake news on social media. Each social media platform has various users across the world. Streaming reasoning can be used to analyze the features in the language patterns in message spreading. The semantic web community has proposed several tools that can be used in stream reasoning. The semantic web community introduced (RDF) for the modeling and encoding of data schemas and ontologies on the

fundamental level. The linked open data is an example of how the database can be linked in the semantic web. (SPARQL) is a query language developed by W3C. SPARQL queries use the triplet pattern of RDF to represent patterns in the data and graphs. Recently, the RDF Stream Processing (RSP) working group recently proposed an extension of both RDF and the SPARQL query to support stream reasoning. For instance, the continuous SPARQL (C-SPARQL) is an example of the SPARQL language for expanding the use of continuous queries.

1.6 The Practical Approach

In real-world practice, the application of a data stream is tied to big data. The current approach to data stream usage can be grouped into these categories: scaling data infrastructure, the mining heterogeneous information network, graph mining, discovery, and recommender system [8]. Scaling data infrastructure is about analyzing the data from social media, such as Twitter, that carry various types of data, such as video, image, text, or even a hashtag trend. The data is generated based on how the users communicate on a certain topic, which leads to various analytics for understanding human behavior and emotions based on the communication between users. Snapchat is now another popular social media application that generates and analyzes live data streams based on the location and the event that occurred.

The mining heterogeneous information network is about discovering the connections between multiple components, such as people, organizations, activities, communication, and system infrastructure. The information network here also includes the relations that can be seen on social networks, sensor networks, graphs, and the web. Graphs are being used to represent nodes and their relations, and graph mining is an efficient method for discovering knowledge in big data. For example, Twitter can represent graph information by visualizing each data type and its relations. Many other kinds of graph information can be obtained from the web. For example, Google has constructed knowledge graphs for various objects and relations. The recommender system is another approach for analyzing a data stream in big data. Through collaborative filtering (CF), the queries in a DSMS system can be improved by adding a new statement, such as rating. It can extend the functionality of DSMS for finding optimization, query sharing,

fragmentation, and distribution. Another strategy is using the content-based model; several platforms, like Amazon, eBay, YouTube, and Netflix, already use this in their systems.

1.7 Significance and Contribution

Data is essential during the big data era because it allows many domains to profit by extracting knowledge and information. This information is used to make the right decisions involving various factors, such as prediction, pattern, profiles, and outliers. Outlier detection is a crucial step in data mining and machine learning. If outliers are not uncovered, the knowledge will be unreliable, and many problems, such as incorrect decisions or predictions, will occur. How to measure outliers is a significant problem in the analysis of big data. Data streams represent big data by their unique characteristics, i.e., large volume and sequential structure. The traditional local outlier detection algorithm has a gap in terms of processing the data stream. The local outlier factor (LOF) is a well-known method used in anomaly detection to detect the outlier. The main challenge of the LOF is that it needs the whole dataset to be stored in memory. Another issue is how the dataset is handled; if anything changes, it must be recalculated from the beginning.

This PhD research proposes three novel local outlier detection algorithms and demonstrates their effectiveness in a number of experiments. The first approach, called a grid partition-based local outlier factor (GP-LOF), has been applied in a data stream and addresses the limitation of the LOF. The second approach is to improve the GP-LOF algorithm by introducing a new method of calculation for the local outlier factor and is known as the local outlier factor by reachability distance (LOFR); it proposes a new technique called grid partition-based local outlier factors by the reachability distance (GP-LOFR). The GP-LOFR algorithm showed some improvement in the precision of outlier detection in several real-world datasets. Both GP-LOF and GP-LOFR have initiated some new ideas for the future extension of the LOF algorithm in big data stream processing. Another technique, called the isolation forest-based on the sliding window for local outlier factor (IFS-LOF), has achieved impressive results with real-world datasets for a concrete mixture. When dealing with large amounts of data in memory, the new IFS-LOF

algorithm can overcome the challenges of both the isolation forest and the LOF algorithm. This dissertation main contribution is the introduction of these GP-LOF, GP-LOFR, and IFS-LOF algorithms.

1.8 Research Questions

Six research questions were addressed in the dissertation. These questions aim to improve the LOF efficiency in processing the data stream and solve the LOF algorithm issue. The six questions are as follows:

1. How does the GP-LOF algorithm apply the LOF in processing the data stream?
2. How does the GP-LOF algorithm solve the memory consumption issue?
3. How does the GP-LOF algorithm deal with incoming data points?
4. Does the GP-LOF algorithm perform better than the DILOF algorithm for the accuracy of outlier detection?
5. Does the GP-LOF algorithm perform better than the DILOF algorithm in execution time?
6. How can the new approach for outlier detection be evaluated regarding the concrete mixture problem?

1.9 The Organization of the Dissertation

Researchers interested in local outlier detection will significantly benefit from this dissertation because it includes two new approaches for processing the LOF in data streams. Additionally, it includes another approach for solving the LOF by combining it with the Isolation Forest (IF) method. There are six remaining chapters. Chapter 2 presents the literature review of the local outlier detecting techniques and discusses the research challenge for the LOF algorithm in processing the data stream. Chapter 3 discusses the background of the outlier detection approach and the dataset used in the experiment. The Grid Partition-based Local Outlier Factor (GP-LOF) algorithm is presented in Chapter 4. The Grid Partition-based Local Outlier Factor by Reachability Distance (GP-LOFR) algorithm is proposed in Chapter 5 as an improved version of the GP-LOF algorithm. In the sixth chapter, a new method for improving outlier

detection in industrial applications is presented. In particular, it can be used to evaluate concrete mixture. Chapter 7 discusses the conclusions and future studies for the data stream processing.

Chapter 2: Local Outlier Detection Techniques in Real-World Streaming Data Processing: A Literature Review

Alsini, R., Alghushairy, O. Almakrab, A., Soule, T. and Ma, X., 2021. Local Outlier Detection Techniques in Real-World Streaming Data Processing: A Literature Review (Under Review)

2.1 Introduction

Outlier detection, often referred to as anomaly detection, detects a rare event, irregular patterns, or objects that differ considerably from the normal dataset. Outlier detection is also called novelty detection, fraud detection, and rare event detection. Several studies have defined the outlier as a record that is incompatible with the dataset [9-15]. Outliers arise from various causes, and understanding these causes helps determine what steps to take when outliers are detected [16]. Therefore, outlier detection may provide important information and have consequences for various fields, such as finance, industry, health, transport, and network intrusion detection, to name a few. Although numerous algorithms have been developed to recognize outliers, they have been more widely used in static environments, and their implementation is difficult in dynamic environments such as streaming data. Large amounts of data are generated in our daily lives, and many are recorded as data streams [17]. The data stream is known as real-time data and can be described as a sequence of inputs to be processed. Extracting information from the data stream is a significant challenge that needs to be addressed. A common approach is data mining, which overcomes the limitation of processing the data stream and stores it in the computer memory [18]. A data mining process includes two key parts: data preprocessing and data mining. Data preprocessing aims to ensure the data are consistent during the data-cleaning phase from the outlier data. If outlier data is not detected in the data-cleaning phase, the output will be neither reliable nor accurate in various practical applications or when detecting the outlier dataset. The data mining approach includes understanding and exploiting the data's behavior by applying several machine learning methods to extract information [5]. Recently, many research studies have been published on outlier detection in data mining. A number of outlier detection reviews have been conducted, and while that number is rising, a new approach of outlier detection algorithms needs to be introduced and discussed. Therefore, we aim to

provide a recent overview of the local outlier algorithms and other methods for applying the local outlier factor (LOF) algorithm. This research is distinct from other research because we focus on local outlier algorithms from state-of-the-art research on stream environments. We also discuss the issues and challenges facing the LOF algorithm and different local outlier algorithms when identifying the outlier in the data stream. In addition, our literature review includes a new technique for determining a LOF score in a data stream. By summarizing the local outlier detection in both static and stream environments, our literature review will significantly benefit researchers and academics in the field of local outlier detection.

The rest of this chapter consists of seven sections: Section 2.2 addresses the literature review methodology, selection, and analysis. Section 2.3 presents the algorithms for the local outlier detection in both the static and stream environment. Analysis and discussion are provided in Section 2.4. Advantages and disadvantages of existing methods are provided in Section 2.5, as well as for the research challenge in Section 2.6. Section 2.7 outlines the conclusions regarding the local outlier detection process.

2.2 Literature Review Methodology, Selection, and Analysis

Our aim with the literature review is primarily to address the recent progress of the local outlier algorithm in both the traditional approach to outlier detection and in the stream environment. Our literature review covers the period from May 2000 to November 2020. We comprehensively scanned electronic databases and papers published in English. The sources for the search included Web of Science, IEEE Explorer, Science Direct, ACM, MDPI, Springer, Taylor and Francis, Wiley, and Google Scholar; these were selected due to their full coverage of quality papers. The keywords used for the search included "local outlier detection, local outlier detection in the stream environment, local outlier factor in the data stream, outlier detection methods, and data stream mining." The requirements for inclusion in this review were articles that dealt with the following: (1) unsupervised approaches for local outlier detection algorithms; (2) new local outliers, depending on a density method, in the static environment; (3) any new LOF algorithm to identify local outliers in the stream environment; and (4) techniques to identify local

outliers in the stream environment. We excluded articles that did not fit the above criteria and any studies that did not provide the complete text, as shown in (Figure 2.1).

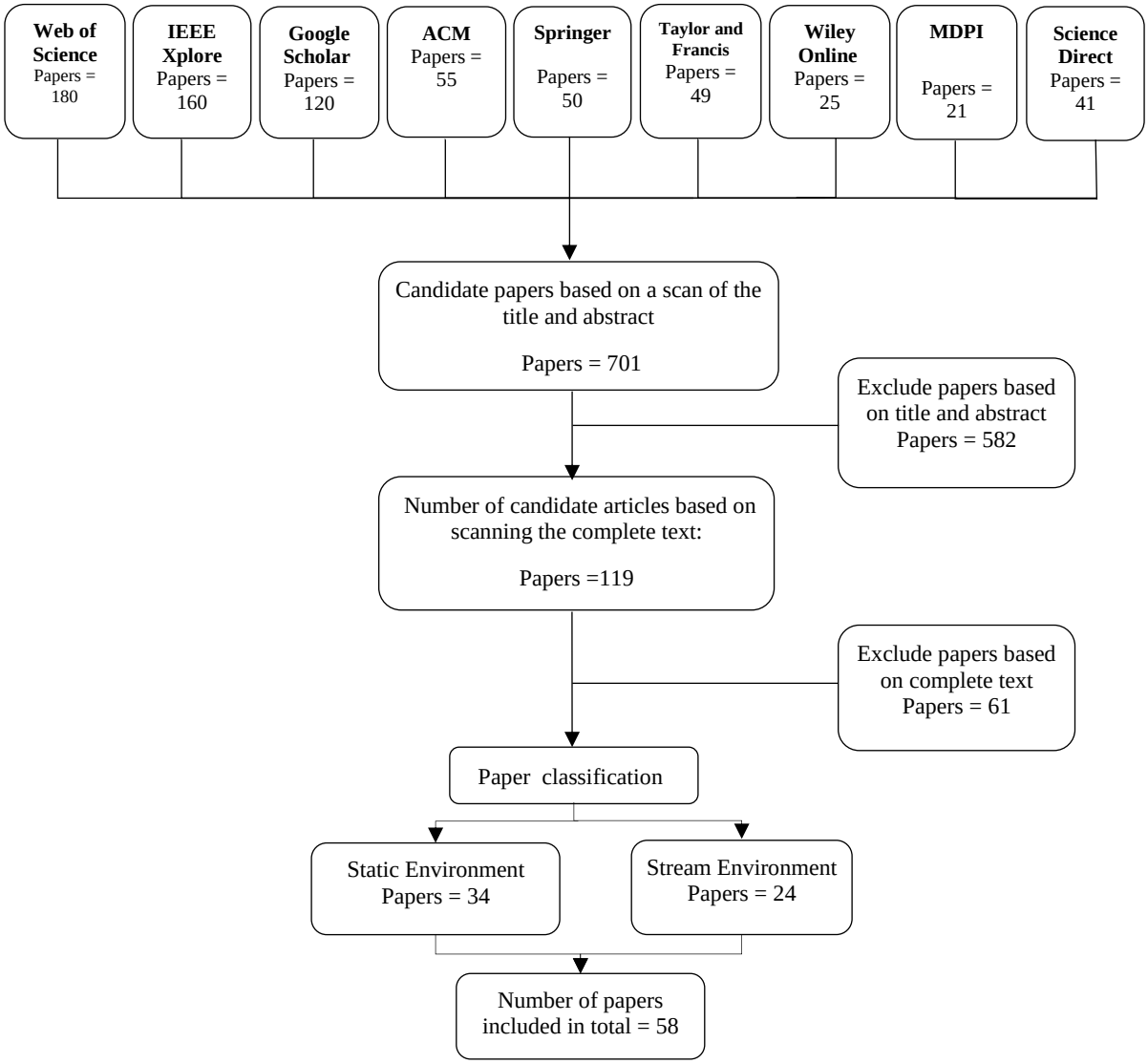


Figure 2.1 The literature search methodology in our work.

After removing duplicates, a total of 701 research papers were reviewed during the initial search based on the title and abstract of the papers. The research papers' contents were then classified into either the static or stream environment. Following this, the complete texts of the articles were used to process the final selection. This resulted in 58 papers in total, as illustrated in Figure 2.1. The common popular local

outliers are discussed in the static environment section; the other local outlier detection papers are briefly mentioned. The recent publications on the LOF algorithm in data streams are discussed in depth in the stream environment section; the remaining research papers on local outliers in the stream environment are briefly considered. Figure 2.2 presents the most local outlier detection methods perform in the static and stream environment.

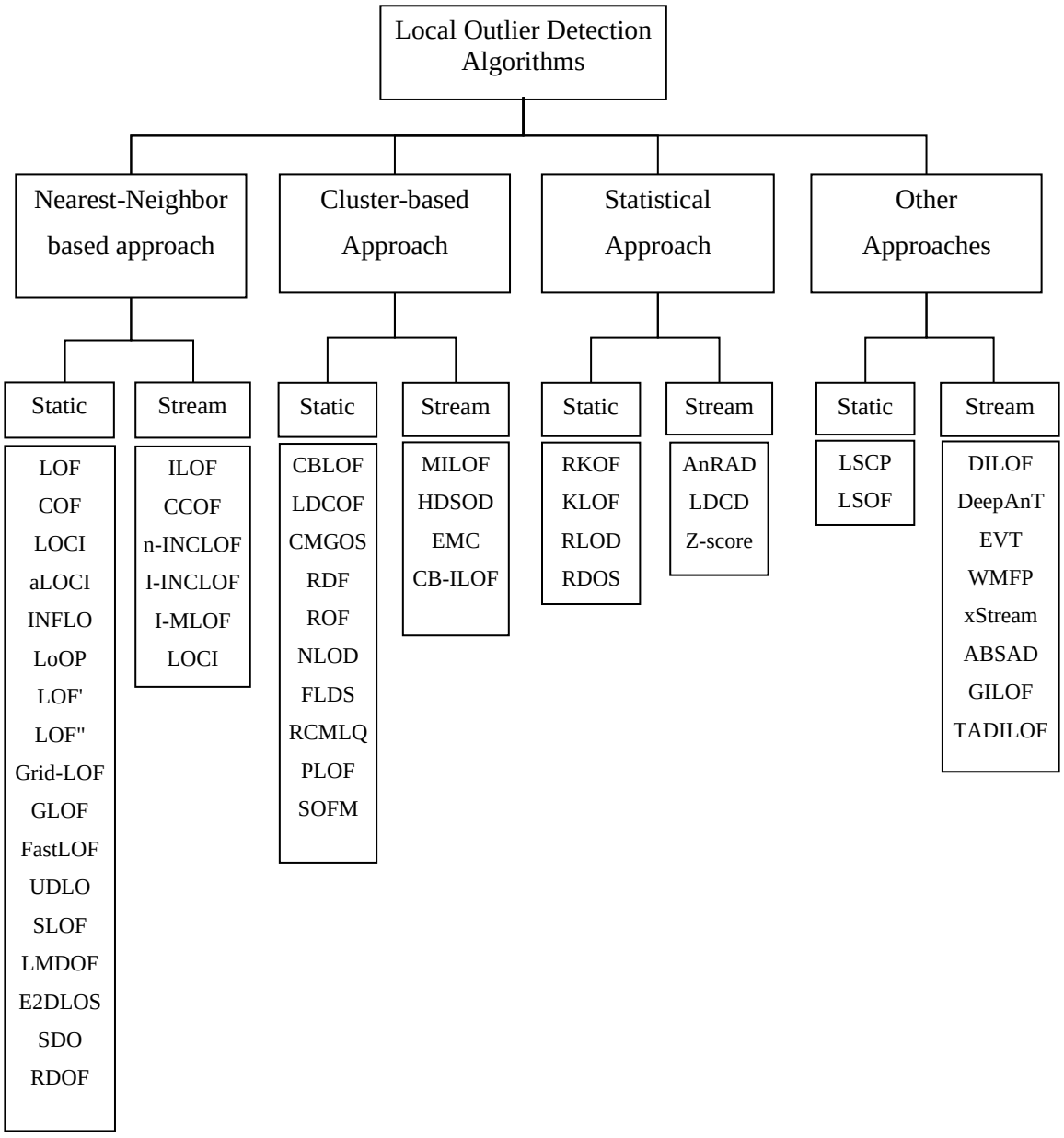


Figure 2.2 An overview of local outlier detection methods performed at the static and stream environment.

2.3 Algorithms for Local Outlier Detection

2.3.1 Outlier Detection in a Static Environment

This section provides a deeper insight into the most well-known algorithms used in local outlier detection and provides some knowledge of the LOF method, which will describe the 34 research papers from the literature search performed in the static environment. Tables 2.1 and 2.2 lists the most well-known techniques in the local outlier detection process, as described in [19]. The two strategies for local outlier detection are as follows. Data mining and machine learning are very significant components in outlier detection in static environments. Several algorithms have been developed for local outlier detection using unsupervised models. The research on local outlier detection has significantly grown over time and is now approached from various perspectives.

Table 2.1 Overview of popular local outlier detection algorithms. NN, nearest neighbors approach; LOF, local outlier factor; COF, connectivity-based outlier factor; LOCI, local correlation integral; aLOCI, approximate local correlation integral; INFO, influenced Outlierness.

Ref., year	Authors	Method	Taxonomy	Characteristic	Comments	Real-World Dataset
[20], 2000	Breunig et al.	LOF	NN-based approach	Manage the spherical data	The first approach to define and clarify the concept of a local outlier	Hockey, Soccer
[21], 2002	Tang et al.	COF	NN-based approach	Handle the path to link neighbors	The linear distribution is solved by applying the chain distance for locating the local outlier	
[22], 2003	Papadimitriou et al.	LOCI	NN-based approach	Apply the half-Gaussian distribution for the distance	Identical to LoOP method, except for the instance quantity being used rather than the distance; it is not important for the parameters	NBA, NY Women
[22], 2003	Papadimitriou et al.	aLOCI	NN-based approach	The quad trees approach is used to speed up the count process	Fast density estimate based on the quantity of data points and depth; overcomes the LOCI method in time complexity	NBA, NY Women

[23], 2006	Jin et al.	INFLO	NN-based approach	Apply the reverse NN for data points	Tackles the problem of data points inside the cluster boundaries; suitable for clusters of different densities for data points	
[24], 2009	Kriegel et al.	LoOP	NN-based approach	Apply the half-Gaussian distribution for the distance	Considers probabilities and statistical approach for the outlier score	Wisconsin breast cancer, pen-based recognition of handwritten digits, metabolic

Table 2.2 Overview of the popular local outlier detection algorithms in clustering-based methods. CBLOF, cluster-based local outlier factor; LDCOF, local density cluster-based outlier factor; CMGOS, clustering-based multivariate gaussian outlier score.

Ref, years	**Authors**	**Method**	**Taxonomy**	**Characteristic**	**Comments**	**Real-World Dataset**
[25], 2003	He et al.	CBLOF	Cluster	A heuristic approach for large and small clusters	Many parameters are involved in the process; local variety of clusters is noted; efficiently finds the local outlier	Annealing lymphography Wisconsin breast cancer
[26], 2012	Amer et al.	LDCOF	Cluster	Assess the cluster density using the spherical distribution	Many parameters are involved in the process; efficiently finds the local outlier	Breast cancer, Pen-local
[27], 2016	Goldstein et al.	CMGOS	Cluster	Apply Mahalanobis distance for estimating the outlier	Divide the size of data points using the k mean	

2.3.1.1. Popular Local Outlier Detection Algorithms

- Local Outlier Factor (LOF)

The LOF is a well-established algorithm used for detecting a local outlier in a density-based model. The principle of the LOF is to process data points through a comparison with their nearby neighbors.

Every data point is calculated to decide if the data points are normal or outliers by degree, such as an outlier factor. The interpretation of the LOF has been explained in [20, 28, 29] and functions as follows.

- Definition 1: *The k distance of data point pt*.

The space between two data points *pt*, *o*, can be determined using the Euclidean distance in n-dimensional space as shown in Equation (1):

$$d(pt, o) = \sqrt{\sum_{i=1}^{n} (pt_i - O_i)^2} \quad (1)$$

Since k is a positive integer and point pt is given in dataset D, the k distance (pt) is defined according to the distance between the point pt and the furthest distance o (o ∈ D) as in the following cases:

- ❖ With at least k data point o'∈D \ {pt} it manages that d(pt ,o') ≤ d (pt ,o).
- ❖ With at most k-1 data point o'∈D \{pt} it manages that that d (pt ,o') < d(pt ,o).

- Definition 2: *k-nearest neighbors of pt*.

Any data point pt is described here by the k-nearest nearest neighbors (kNN), which can be described as any data point q whose distance does not exceed the k distance (pt), as defined in:

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

- Definition 3: *Reachability distance (Reach-dist (pt)) with respect to o.*

If *k* is a positive integer, the reachability distance of point *pt* with any point *o* is described by:

$$Reach\text{-}dist_k(pt, o) = max\{ k\text{-}dist(o), dist(pt, o)\} \quad (3)$$

According to definition three and Figure 2.3, the distance is determined on the basis of the k distance (o) in two directions. If the distance is far from the k distance (o), it processes as a reachable distance. Otherwise, if the distance is shorter than the k distance (o), it performs as a k distance (o).

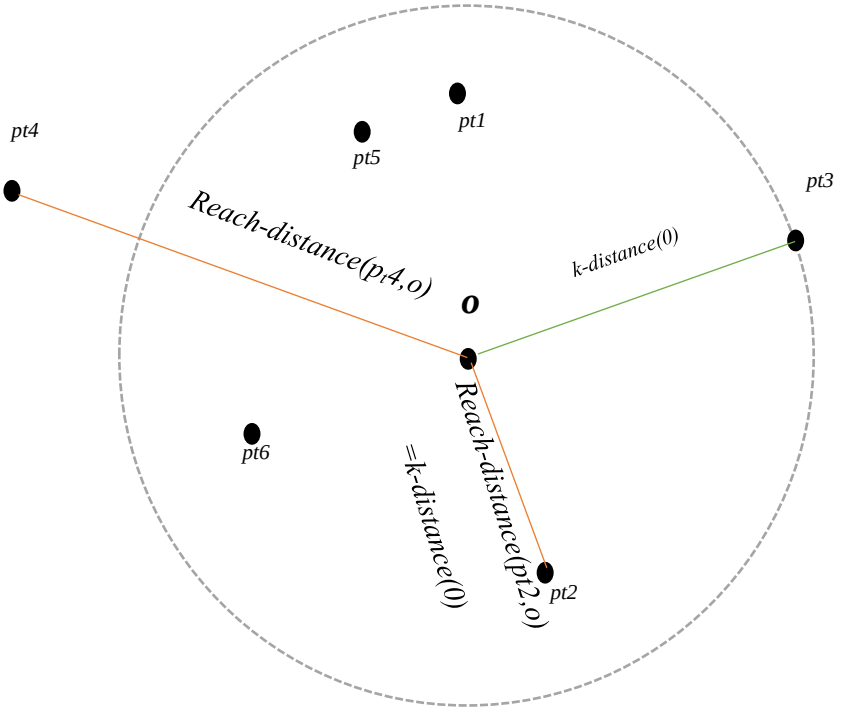


Figure 2.3 Multiple data points for measuring the reachability distance pt to o, if k is equal to 6.

- *Definition 4: Local reachability density of lrd (pt).*

Density-based clustering algorithms define density using two parameters: 1) MinPts for the minimum number of data points and 2) volume. The authors of [2] used reach-distMinPts(pt,o) for o∈ NMinPts(pt) as a volume measure. The local reachable density (lrd) is the reverse of the reachable density of pt. The following Equation describes the local reachability density as:

- $$Lrd_{Min-points}(pt) = 1/\left(\frac{\sum_{o\in N_{Minpoint}(pt)} Reach\text{-}dist_{Minpoint}(pt,o)}{|Minpoints(Pt)|}\right) \qquad (4)$$

- *Definition 5: LOF of pt.*

To calculate the LOF of (pt), each definition must be followed to obtain the LOF score, as shown in Equation (5).

$$LOF_{Min-point}(P_t) = \frac{\sum_{p\in P_{Minpoint}(pt)} \frac{Lrd_{Minpoint}(o)}{Lrd_{Minpoint}(pt)}}{|P_{Minpoint}(pt)|} \qquad (5)$$

The LOF generates the score according to the proportion of the local reachability density and the Minpts of pt. To assess whether the pt is normal or an outlier, the threshold score θ is used. The LOF algorithm is valuable because it effectively finds local outliers. The LOF's downside is the sensitivity of the execution time and the minimum points value.

- Connectivity-based Outlier Factor (COF)

The COF method involves a principle similar to that of the LOF technique. It works differently in estimating the kNN required to compute the distance between the neighbors and data points. The LOF method computes the kNN using the Euclidean distance, as the data points are assumed to be distributed in spherical form. The COF method is calculated based on a linear correlation approach known as chain distance [21]. In particular, it works by adding the closest data points to the neighbor's set until it reaches a size of the kNN that is a minimum of all the remaining data points. The COF method follows the same definitions as the LOF to find the outlier score. The COF approach's strength is based on its ability to discriminate between low density and isolation [14]. Execution time is the downside of this strategy; the COF takes longer than the LOF.

- Local Correlation Integral (LOCI)

Choosing the k value is a primary decision for predicting the previous methods' efficiency. However, one of the challenges is related to the estimation of the k value based on the dataset. The local correlation integral (LOCI) solves the problem by choosing the k value using a maximization approach [22]. The key principle is to pick the k with the highest value from each data point. The LOCI uses the r neighborhood with the use of a radius r to obtain the top score. The radius r expands over time. The LOCI uses a local outlier probability (LoOP)-like half-Gaussian distribution but with a particular method; the neighbors' number of points is used instead of the distances. It also makes two different estimates of the local density: first, instead of the local density ratio, it compares the sizes of two different neighborhoods; and, second, an α parameter manages the ratios of the various neighborhoods. The LOCI's advantage is in obtaining the maximum value. However, for large datasets, the LOCI

method is very slow, since the r radius must be expanded from one stage to the next, which is a drawback for detecting an outlier.

- Approximate Local Correlation Integral (aLOCI)

The approximate local correlation integral (aLOCI) was introduced to increase the LOCI speed when processing two neighbors. The aLOCI contains quadtrees and some constraints for parameter α [22]. For the counting estimation in the aLOCI method, performance is considered good if there is a data point in the middle of a cell in a quadtree. Otherwise, if the data point is near the border, performance is considered weak. Therefore, several quadtrees are built with the expectation of producing a good approximate tree for every data point. Additionally, the depth of the tree (L) should be defined to produce a good approximate tree. The aLOCI's power is its potential to speed up the efficiency of a quadtree. The number of tree processes is a weakness of the aLOCI method.

- Influenced Outlierness (INFLO)

The influenced outlierness (INFLO) plays a role when the data points are in a cluster with different densities and are related to each other [23]. The INFLO is more efficient than the LOF for handling a cluster's data points by adding a Reverse Nearest-Neighbor (RNN). Additionally, with the kNN for storing the points and their neighbors, the INFLO method computes the outlier score by combining both neighbors' sets to obtain the local density. The INFLO method uses the same computation method as the LOF for the local reachability distance. The benefits of using the INFLO algorithm are that the outliers scores can be measured in greater depth if the dataset contains clusters of various densities that are close together. In other adjacent density distributions, it can also find outliers. The drawback of this approach is that it has a long processing time.

- Local Outlier Probability (LoOP)

The local outlier probability (LoOP) follows another technique of the density-based method to estimate the local density. It uses a static approach to determine outlier probability, whereas the LOF

method uses the outlier score to estimate local density [24]. Therefore, LoOP uses a Gaussian distribution to compute the distance to evaluate the neighbors. Usually, because distances are often on the positive scale, the LoOP uses a half-Gaussian range to obtain the local density. The proportion of each point is compared to its neighbors to get the outlier score. A normalization and a Gaussian-error feature are used to transform it into outlier probability. The benefit of the LoOP concerning outlier probability is that it provides a new outlier identification strategy. The LoOP's limitations are that it is time consuming, and it can result in incorrect steps due to the probability measurement size.

- Cluster-Based Local Outlier Factor (CBLOF)

The CBLOF's core principle is based on using clustering to assess the clustered areas in the dataset and then measuring each cluster's intensity [25]. The CBLOF works as follows: first, the k- means is used to cluster the data points; then based on the result, the CBLOF classifies the corresponding clusters as either large or small clusters; lastly, an anomaly score is achieved, which depends on the distance from the center of each data point's cluster that is based on being multiplied by the data points belonging to its cluster. The CBLOF measures the cluster densities using a cluster approach. The downside to this approach is the sensitivity of the k value.

- Local Density Cluster-Based Outlier Factor (LDCOF)

As previously mentioned for CBLOF, it is controversial to estimate local density using only the number of cluster members and to neglect the density of the cluster. LDCOF focuses on this issue by estimating the cluster densities, given that the cluster members are spherically distributed [26]. The LDCOF uses a similar method of determining the k mean by dividing the data points into small or large clusters. Then, the average distance of all members of the cluster is calculated for each cluster to the centroid. Lastly, the LDCOF values are determined by dividing the size of the instance by the average distance from their cluster center. Cluster density can be determined by the LDCOF method and for the spherical distribution. The disadvantage is the sensitivity of the k value.

- Clustering-Based Multivariate Gaussian Outlier Score (CMGOS)

In CMGOS, a multivariate Gaussian model is calculated for local density when Mahalanobis distance is used as a basis for estimating the outlier [27]. The first step is applying the k-means to divide the clusters into small or large clusters. Next, the covariance matrix is robustly determined for each cluster. Finally, the GMGOS score is determined in the chi-squared distribution, separated by a specific confidence interval between an instance's Mahalanobis distance and the closest cluster's center. The essence of the interval in the Mahalanobis distance implies that outlier scores are increasing [30]. A large score for outliers is observed. Robustness is essential for the estimation of the covariance matrix because outliers are known to significantly affect the change. In general, three specific methods of computation are suggested to address the issue: reduction, regularization, and the minimum covariance determinant (MCD). The CMGOS method's strength is using both the k mean and x mean on the multivariate Gaussian scale to measure the outlier score. This algorithm's shortcomings are that more k values are needed, and it is not appropriate for large datasets.

2.3.1.2. Different Local Outlier Detection Techniques in the Static Environment

Chiu at el. [31] proposed three enhancements for the LOF algorithm: LOF', LOF'', and grid-LOF. The advantage of the LOF' is that it can deal with a large dataset by eliminating both rd and lrd for the outlier score. The drawback of this method is that the Minpts can be incorrectly adjusted based on the measurement outcome. More than LOF and LOF', the LOF'' approach identifies the Minpts for neighbors. The grid-LOF introduces the concept of dividing the points into several spaces known as grids. The strength of grid-LOF is that it prunes the non-outliers; the drawback of the grid-LOF method is that it is slower due to the challenge of selecting its parameters. Jiang et al. [32] introduced a new method based on the nearest-neighbors called the generalized local outlier factor (GLOF). The advantage of the GLOF is that it processes without a threshold for the outlier in the dataset. The GLOF algorithm relies on the k value for performing the measurement. Goldstein [33] introduced the FastLOF outlier detection algorithm for unsupervised anomaly detection in datasets. Based on the study results,

the FastLOF algorithm was shown to be an expectation-maximization algorithm that computes anomalies in an incremental manner about 80% faster than the standard or traditional LOF methods. Cao et al. [34] introduced a new approach for detecting the outlier quickly through the specification of uncertain data points. The proposed method, Density-based Local Outlier detection on Uncertain data (UDLO), focuses on computing the density of the data points rather than using all the k neighbors to search for the outlier, as defined in the LOF approach. The algorithm's power is provided by concentrating on the Euclidean distance rather than measuring all the k neighbors. According to Guan et al. [35], the similarity-based local outlier factor (SLOF) algorithm is an example of density-based outlier detection. The SLOF algorithm is more accurate and versatile in handling big data and datasets than the LOF algorithm. However, the SLOF algorithm is ineffective when the dimensions of the datasets are low, and the data points are dense. Liu et al. [36] proposed an outlier detection method based on local minima density (LMDOF). The advantage of the LMDOF method is that it can detect outliers more accurately than the traditional LOF algorithms. Su et al. [37] implemented an efficient method for detecting the local outlier in dispersed data, called E2DLOS. Instead of using the LOF as a method of estimation, they added a modern definition of the local outlier, called the local deviation coefficient (LDC), which is used to identify completely all the closest neighbors of the data points. This algorithm has the benefit of enhancing the measurements for computational efficiency. Vazquez et al. [38] suggested a novel algorithm for detecting outliers that focuses on low density, known as sparse data observers (SDO). SDO can reduce the time and formulation of the calculation. Ning et al. [39] indicated a reasonable outlier approach for calculating the density of the object; it is called the relative density-based outlier factor (RDOF) detection method. This method measures the density of data point neighbors. This strategy's value is that it can cope with low-density pattern issues.

Ren et al. [40] presented the relative density factor (RDF) algorithm, which uses a vertical data model for finding outliers (P-trees). The advantage of the RDF when the size of the data is expanded is that it demonstrates better scalability; however, it needs more time for computation. Fan et al. [41] introduced a new approach for defining the cluster called the resolution-based outlier factor (ROF), which has close

r-neighbors that can find the outlier based on metrics, such as a score. The advantage of the ROF is achieved by using a growing window for the outlier score. The ROF approach has drawbacks because it cannot rank or distinguish the outliers, cannot accommodate various density clusters, and needs more storage space. Du et al. [42] proposed a novel local outlier detection (NLOD) algorithm. The advantage of using the NLOD algorithm is its high-performance values, which are indicated by greater Area under the ROC Curve (AUC) values compared to LOF algorithms. However, the disadvantage of the algorithm is that its execution time is slower. The authors of [43] proposed fast outlier detection based on a local density score (FLDS), which is able to identify points in a dataset for effective outlier detection. The advantage of the FLDS is that it is applicable in local outlier detection. Su et al. [44] proposed rough clustering based on multi-level queries (RCMLQ) to address both separation and dispersion between objects and their neighbors. The advantage of the RCMLQ algorithm is that it reduces the amount of data required by the LOF and LDC to be quantified to detect outliers in scattered datasets. However, the problem with the RCMLQ is that it can fail to truly reflect the associated abnormalities in data irrespective of the shortcomings. Babaei et al. [45] developed a pruning-based outlier detection algorithm (PLOF) that is computationally cheaper, more reliable, and more effective than the existing LOF models, which are computationally expensive. The drawback of the PLOF algorithm is the need to cluster data in advance before pruning the outliers. Yang et al. [46] analyzed anomaly detection using the self-organization feature map (SOFM) clustering algorithm. The advantage of using the SOFM algorithm is that its functionalities can be improved using other algorithms that vary the number of neurons, such as the canopy algorithm. It provides better performance compared with LOF and NELOF; however, the disadvantage of the SOFM is that it requires further neuron improvement to function better.

Gao et al. [47] analyzed the robust kernel-based local outlier detection (RKOF) algorithm. The advantage of the RKOF algorithm is that it addresses most of LOF's challenges. However, the RKOF model's disadvantage is that it may fail to detect other outliers even though it is fast and efficient. Miao et al. [48] proposed the kernel density-based local outlier factor (KLOF) algorithm, which they describe as anomalous cell detection with a kernel density-based local outlier factor. After analyzing the KLOF

algorithm, they concluded that it addresses major challenges affecting density-based outlier detection algorithms by capturing the exact relative degree of isolation. It has values greater than one, which makes it easier to detect whether an incoming object is an outlier to avoid false alarms. Du et al. [49] discussed robust local outlier detection (RLOD). The advantage is that the algorithm does not make any assumptions concerning the probability distribution of the real data under analysis. The disadvantages of the algorithm are that it does not support distributed computing and it cannot be deployed when dealing with abnormal load detection. Tang et al. [50] suggested a method of finding outliers using the local Kernel Density Estimation (KDE) algorithm. The relative density-based outlier score (RDOS) was implemented to evaluate the local outer score for data points. The RDOS method uses KDE to measure the local outliers by applying an extension to the nearest neighbors from the data point to measure the local density. The advantage of the local KDE is that it can provide density estimation at the data point location, whereas the RDOS can identify the local outliers in the local KDE. Wang and Deng [51] examined the variable local outlier factor (VLOF). The main advantage of the VLOF algorithm is that it more easily reflects the local variable information. Furthermore, it is easier to compare its performance efficiency than the traditional LOF models. However, the model's disadvantage is that it takes time to detect a fault in systems through big data analysis. Zhao et al. [52] provided a new local algorithm called locally selective combination in parallel outlier ensembles (LSCP). The power of this algorithm is that the local outliers can be quantified. Wang and Zhu [53] proposed a local structure outlier factor (LSOF) algorithm. The advantage of the proposed LSOF algorithm is that it has an incredible ability to estimate, measure, and define the novel nearest-neighbors tree (NNT), allowing it to estimate the existing neighborhood instantaneously. The LSOF algorithm has shorter time complexity. However, the main drawback of the LOSF algorithm is that it requires more memory to achieve maximum speed.

2.3.2 Outlier Detection in the Stream Environment

So far, the bulk of research on local outlier detection has focused on static environments rather than stream environments, so the stream environment has been studied less than the static environment. As a consequence of recent interest in creating conventional local outlier detection algorithms that operate in

the stream setting, the development of new algorithms in terms of the LOF algorithm has increased. This section discusses several LOF methods that led to the estimation of the LOF score and reviews other local outlier detection algorithms in the stream environment, describing the 24 research papers performed in the stream environment. Table 2.3 provides the essential information for LOF methods in the data stream that met the criteria for our review.

2.3.2.1 Popular LOF Algorithms in Data Stream Processing

- Incremental Local Outlier Factor (ILOF)

To overcome the vulnerability of the LOF in processing the data stream, Pokrajac et al. [54] introduced the incremental local outlier factor (ILOF). The key principle of the ILOF procedure is to provide an efficient method for detecting the outlier in a stream environment. ILOF is based on a density-based outlier detection method that works by updating and preserving the *k-distances, lrd,* and the LOF values when a new data point (np) is added, or an old data point is deleted. To measure the outlier score, ILOF uses the same elements as the LOF algorithm; those elements are the *k-distance*, *the reachability distance (rd), and the lrd*. The ILOF insertion process involves two stages: first, *rd, lrd,* and the LOF score are measured according to the np value; and second, the *k-distance, rd, lrd,* and LOF score for existing data points are updated. Despite the strength of the ILOF algorithm in the data stream, the main issue is the memory consumption for storing the old data points. As a consequence, significant amounts of memory and time are required for every *np*.

- Memory-efficient Incremental Local Outlier Factor (MILOF)

Memory-efficient incremental local outlier factor (MILOF) is an unsupervised outlier approach used to determine the local outlier in the data stream [55]. The MILOF is capable of reducing the time complexity as well as the memory constraint, and it is suitable for a variety of applications. It also resolves the memory limitation in both the LOF and the ILOF stream environments. The MILOF method processes the data points in three phases: summarization, merging, and revised insertion. As the number of points exceeds the memory limit, the summarization phase is undertaken. The first half of the data

points are summarized, whereas the later points remain unchanged to give the new streaming data points higher resolution. The k-means algorithm is used to calculate the LOF score for clustering the first half of the data points. The merging phase is conducted after the second half of the data points from the summarization stage are obtained. Based on incoming data points, a new cluster is formed. A new cluster is subsequently paired with an established cluster to create 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 uses a similar ILOF algorithm concept. First, it computes the LOF for the new incoming data points. Next, if necessary, it updates the k-distance, rd, lrd, and the LOF values for the established data points

- Density Summarization Incremental Local Outlier Factor (DILOF)

In data stream processing, DILOF was built to overcome the weaknesses of the ILOF by having two stages: detection and summarization [56]. The detection process is used for updating previous data points as new incoming data points arrive by applying both the ILOF method and a skipping scheme. The ILOF method works for detecting the outlier, whereas the skipping scheme is used to detect any sequence in a number of points. The summarization stage is used for an optimized result based on the nonparametric density summarization (NDS). NDS involves using a decision variable that relies on the gradient descent method to summarize the data points. In general, the DILOF performs well, compared with MILOF and ILOF, in processing the data stream in terms of accuracy and time complexity. However, the problem with the DILOF is that the gradient descent may be trapped in the local minima.

Table 2.3 Summary of LOF algorithms in data stream processing. ILOF, incremental local outlier factor; MILOF, memory-efficient incremental local outlier factor; DILOF, density summarization incremental local outlier factor.

Ref, Year	Authors	Method	Types of window techniques	Characteristic	Method	Comments	Real-World Dataset
[54], 2007	Pokrajac et al.	ILOF	Landmark window	Whenever a new data point is inserted, it stays up to date		Introduces the concept of local outlier factor in data stream	
[55], 2016	Salehi et al.	MILOF	Sliding window	Data summarization by k-means	Cluster	Overcomes the limitation faced by the ILOF algorithm for keeping the density of the data and for the time complexity	Vowel, Pendigit, Letter, Motion Trajector
[56], 2018	Na et al.	DILOF	Sliding window	Data summarized by the gradient descent methods	Optimization	Issues of keeping the data point density in MILOF are addressed Gradient descent methods may be trapped in local minima	Vowel, Pendigit KDD SMTP, KDD HTTP

2.3.2.2 Different Local Outlier Detection Techniques in the Stream Environment

Gao et al. [57] constructed a model for automatically changing the n-IncLOF threshold to boost outlier identification. The LOF adjustment of the n threshold relies on the standard deviation of the LOF in the sliding window. This model's strength dramatically improves the detection rate and the false alarm detection rate. The connectivity-based cumulative outlier factor (CCOF) was developed by Pokraiac et al. [58] and is as effective as the connectivity-based outlier factor (COF) algorithm. Karimian et al. [59] proposed improving the incremental LOF with a more suitable algorithm for dynamic data streams. The

I-IncLOF algorithm uses a flexible window that allows the update and detection of outliers. Wang et al. [60] proposed incremental multiple instance outlier detection (inc I-MLOF) as a suitable algorithm for the multiple instances (MI) background. Kalliantzis et al. [61] used a distributed and density-based outlier identification strategy. The method deals with the data stream's multidimensionally. The study illustrates how the estimated calculation technique for the local correlation integral (LOCI) can be used to enhance anomaly detection scalability and reliability when large amounts of data are involved. Liu et al. [62] proposed an algorithm referred to as the Lazy update method of UKOF (LUKOF). The outcome of this approach showed that the LUKOF algorithm can detect outliers for data streams more quickly. However, this proposed method does not enable the update on the new dataset. Yang et al. [63] introduced the detection algorithm to extract local outlier factor (ELOF). This algorithm was split into three phases: anomaly classification, anomaly detection, and data extraction. The ELOF model works more effectively in regard to the aspects of time and accuracy; however, the ELOF model is considered to perform poorly when it deals with different parameters in a large dataset.

Ren et al. [64] addressed the memory properties of data streams. They implemented a modern method, called heterogeneous data streams outlier detection (HDSOD), to address the problems of processing heterogeneous data streams. The strategy's strength is using a partition-cluster method for the data stream, partitioned and placed in a cluster reference. The outlier value is measured and displayed according to the number and extent of the cluster reference representation. The evolving micro-cluster (EMC) is a new process that is adopted in processing the data stream, as provided in [65]. To determine progression on the concept drift term, the method dynamically learns the changes in the micro-clusters. This approach's benefit is that it enables evolution to be separated from noise distribution using the concept drift. According to Gao et al. [66], the incremental local outlier factor (ILOF) is an effective outlier detection method that assigns outlier scores to different data points according to their degree of abnormality. The authors analyzed the ILOF algorithm and tried to address its deficiencies by proposing a cube-based ILOF (CB-ILOF) algorithm. Based on Gao et al. [66], the time complexity of the CB-

ILOF algorithm, which depends on the runtime memory, performs better than ILOF. However, the disadvantage of the algorithm is that it requires more memory space to perform better.

Chen et al. [67] developed a model called a neuromorphic anomaly detection (AnRAD). The AnRAD model is based on deduced probabilistic inferences. The model aims to improve memory and measurement accuracy while detecting abnormalities. This model's power is that it overcomes memory consumption problems by growing the incremental learning pace and enhancing the efficiency and consistency in detecting anomalies. Ishimtsev et al. [68] developed a technique using the model-free anomaly detection method for a time series model that provides a probabilistic outcome for abnormalities. The technique's advantage is that by using simple approaches like the lazy drifting conformal detector (LDCD), it achieves a result close to advanced anomaly detection methods. The drawback of the model is that more LDCD methods are necessary to maintain appropriate validity. Yang et al. [69] provided a quick method to identify the local outliers in data stream. The value of this approach is that it focuses on reducing the local outlier factor estimates by applying the Z-score pruning method to overcome the constraints, such as time consumption. However, the method relies on a model of prediction. Qin et al. [70] suggested a new outlier strategy to fix the limitations associated with current approaches given contemporary high-speed data streams. The algorithm uses KDE to find and enhance outliers in the streaming environment.

Munir et al. [71] unveiled a new form of detection for anomalies based on deep learning. The procedure, called DeepAnT, applies to data from a time series. The advantages of the method are that abnormalities can be observed in real-life situations, as in stream environments; however, the low quality of the datasets may weaken the influence of the process. Siffer et al. [72] suggested a strategy to identify outliers based on the extreme value theory (EVT) in streaming time series. They provided a robust method for detecting the univariate and unimodal models. This method's benefit is that there is no requirement for a manually set threshold or distribution assumptions. However, the gap in the multivariate case weakens this strategy. Cai et al. [73] identified a pattern-based, multi-phased approach for recognizing outliers, which is referred to as the Weighted Maximal Frequent Pattern-based outlier

(WMFP). The benefit of this approach is that it can distinguish outliers more easily, in particular in a weighted data stream. The acceleration of the detection phase of outliers liberates the recurrent patterns that are better than the frequency patterns. Manzoor et al. [74] proposed xStream, a density-based collaborative anomaly sensor for settings with extreme attributes, such as evolving data points and feature space. The algorithm is efficient and reliable for evolving streams. The strength of the xStream model is that the anomaly can be more easily observed, even with massive noise. In a high-dimensional and non-stationary data stream, Zhang et al. [75] used fault detection techniques. The method of separating subspace faults from high-dimensional datasets focused on angles. Alghushairy et al. [76] introduced a new approach for local outlier detection by using the genetic algorithm to optimize local minima. The goal of the GILOF algorithm is to solve the limitation of DILOF in optimizing local minima. As with the DILOF method, the GILOF function operates in two phases: the detection and summarization phases [56]. The detection process is identical to the DILOF process. The GILOF algorithm improves the DILOF method in the summarization phase. The genetic density summarization (GDS) applies a genetic algorithm (GA) to summarize the data points, which leads to the search for the best local minima [76]. According to Huan et al. [77], the time-aware density-based incremental local outlier detection (TADILOF) approach aims to resolve of the variance in data that change after a while, and which has not been addressed in other types of algorithms. The proposed algorithm was divided into two stages: summarization and detection. The TADILOF applies only a minimal level of memory compared to the DILOF, although it can be stated that these two models do not use a large volume of memory and are appropriate for a data stream setting. In terms of execution time, TADILOF achieves the same performance as the DILOF algorithm.

2.4 Analysis and Discussion

Outlier detection is a meaningful research subject for many domains. A variety of reviews, studies, surveys, and books have addressed outlier detection methods in several fields and applications. Patcha et al. [78] provided an exhaustive analysis of recent and current systems for anomaly detection in intrusion detection systems and the tools used to identify outliers. Agrawal et al. [79] explored different

anomaly detection approaches in data mining to further grasp current techniques that support researchers regarding intrusion detection. Ahmed et al. [80] addressed finding outliers in the financial domain. Markou et al. [81,82] presented an outlier detection method using a statistical approach and neural network. Hodge et al. [83] presented various outlier detection algorithms and techniques as unsupervised approaches. Goldstein et al. [30] looked at different unsupervised outlier detection methods, whereas Domingues et al. [84] addressed several outlier detection techniques applicable for detecting outliers. Wang et al. [14] addressed the recent progress and challenges faced in outlier detection methods. Chen et al. [2] discussed the most widely used outlier detection approaches in big data streams. Tellis et al. [86] presented various methods for detecting outliers in multiple data stream applications. Alghushairy et al. [19] described recent local outlier detection algorithms applicable to processing a data stream. Additionally, papers on outlier detection are available in [13,87-89].

2.4.1 Motivation and Limitation

These research papers presented a variety of techniques for detecting outliers and applied them in a variety of domains. Patcha et al. [78] present some of the existing solutions for anomaly detection in network intrusion detection systems. This research aimed to demonstrate how different anomaly detection approaches can be utilized to identify unusual traffic in networks resulting from attacks. The limitation of this research paper is that it focuses on the current anomaly detection techniques, as including false alarms and the inability of the system to scale to different environments. False alarms occur when the system detects genuine traffic as being unusual. The result of this is that legitimate traffic could be stopped, resulting in poor data transmissions. Therefore, improving the accuracy of the outlier detection techniques is needed. Ahmed et al. [79] mentioned that their article's rationale was to survey anomaly detection techniques from the financial perspective and discuss issues relating to datasets. According to the article, only a few publicly available real-world datasets assess anomaly detection systems in the financial sector. The main reason for this is financial data is sensitive, and most organizations are not willing to release it due to privacy reasons and competition. Agrawal et al. [80]

offered a fundamental review of data mining techniques for detecting anomalies. The limitations of this study are related to current anomaly detection methods. One of the issues was insufficient precision when processing a large dataset. Markou et al. [81] presented a statistical method that could be used to detect outliers. The limitation of this approach is that it cannot deal with large datasets; this could be due to an inability to adequately minimize parameters, insufficient independence and generalization, poor adaptability, or high computational complexity. Also, Markou et al. [82] reviewed neural network-based approaches for detecting the outlier. The motivation of the researchers was to present a comparative analysis of neural network-based systems because such a study was missing in the available literature at the time. Only a few studies had compared different models using the same dataset, which meant that it was difficult to establish the appropriateness of different algorithms on different types of data. The authors also wanted to provide foundational comparative work to support future analyses and assessments. Some of the neural network-based approaches that they explored had limited effectiveness and efficiency in local outlier detection. Hodge et al. [83] provided a broad survey of current techniques for detecting outliers; the idea was to acquaint the audience with a feel for the assortment and diversity of the methods available. The authors also aimed to highlight considerations when choosing an algorithm to suit the data and objectives at hand and to meet the requirements of scalability, incremental capabilities, and accuracy. Goldstein et al. [30] provided an in-depth evaluation of unsupervised anomaly detection algorithms using different datasets. According to the authors, existing studies did not evaluate the algorithms using different datasets, which limited decision-making on applying the algorithms for diverse uses. The study highlighted some of the weaknesses associated with current approaches to local outlier detection. Domingues et al. [84] utilized publicly available datasets and new industrial datasets to conduct experiments to determine the scalability, robustness, and memory consumption of different outlier detection algorithms. Therefore, rather than developing outlier detection algorithms, the objective was to evaluate and compare their performances. The findings of such a study could help organizations and other researchers when choosing the best algorithm. The approaches examined had limitations in terms of achieving optimal precision, robustness, and memory

usage. The ability to detect anomalies precisely is a key aspect of effective detection systems. In terms of time complexity, the algorithm should perform computations efficiently and identify outliers more quickly. The research papers [14,19,85] focused on the issues and challenges affecting approaches developed and deployed in detecting outliers in big data streams. For example, density-based methods' effectiveness depends on the careful configuration of several factors and the experience of time complexities.

Various techniques for detecting outliers are presented in these research papers. However, when a new method in the unsupervised approach, specifically the local outlier detection method, is proposed, these research papers limitations become an impediment when dealing with large amounts of data. As a result, we provide an overview of local outlier algorithms as well as alternative methods for implementing the local outlier factor (LOF) algorithm in the most common big data approach. Outlier detection is typically implemented in these approaches in three ways.

- *The Supervised Outlier Detection Approach*

This approach aims to predict the classified dataset anomalies. Data collection in the supervised approach requires preparing the data for training and testing the module to indicate the normal and abnormal classes. Therefore, some data may be defined as outliers, while other data may be taken as normal. The classification methodology is associated with the supervised outliers for predicting the anomalies [90]. A problem in classification is the data distribution inequality when training the dataset: outliers are much less frequent than normal points. This issue has already been discussed in the machine learning and data mining research papers [91-93]. For example, decision trees like the C4.5 [94] method suffer from an imbalance in data, which can be resolved using techniques, such as support vector machines (SVMs) or a machine learning method like an Artificial Neural Network (ANN)[95].

- *The Semi-Supervised Outlier Detection Approach*

This approach aims to identify anomalies based on the information gained from the data that form part of the labeled data [96-98]. The semi-supervised learning approach is more efficient than supervised learning because it uses unlabeled data that do not require the knowledge of the distribution of the data points or the historical data points. Therefore, it works by using a model learned from the training dataset to recognize the test result's abnormalities. Examples include network intrusion detection[99]. The most common machine learning method is the one-class support vector machine, which supports vector data to detect anomalies [100].

- *Unsupervised Outlier Detection Approach*

Unsupervised outlier detection relates to the use of unlabeled data to identify an unusual occurrence or object distinct from usual behavior. It is more flexible than the previous approaches in finding the outlier based only on the dataset structure. The key goal is to find the outlier according to the dataset's score. The most common methods in unsupervised applications focus on the densities or distance of data points to estimate the normal and outlier data. There are several techniques for unsupervised outlier detection that were addressed and discussed in [30,101,102]. This review focuses on the detection of local outliers in the unsupervised method.

2.5 Advantages and Disadvantages of Existing Methods

2.5.1 Nearest-Neighbor-based Outlier Detection Methods

The nearest-neighbor-based outlier detection has been explored in multiple outlier detection approaches [6,79]. The nearest-neighbor approach relies on a distance measurement between the data point and its nearest neighbor. The nearest-neighbor method utilizes the degree of closeness between the data point and its nearest neighbor to determine its distance from it. Different distances are used based on the attribute type [103]. Euclidean distance is a common choice of an attribute that has continuous characteristics, such as a data stream. There are two types of nearest-neighbor outlier

detection strategies [104]. The first, the distance between a data point and its closest neighbor, is used to measure the k-distance for the outlier score. The second, the relative density approach, calculates the outlier score to evaluate the data point. The strength of this approach is that it can process without prior assumption regarding the underlying data distribution. It is also suitable for various forms of data since it can handle different kinds of data points. The weakness of this approach is that it takes a long time to compute.

2.5.2 Cluster-based Outlier Detection Methods

The cluster-based outlier detection method relies on the data mining method that divides data points into different clusters with similar data points. In several clustering approaches, the primary assumption is that typical data is also related to broad and compact clusters, whereas outliers are separated or clustered in another class [105,106]. Two approaches to cluster-based techniques are widely used for getting the outlier score [107]. The first is the distance from the cluster core. The standard data points are near the cluster centers, while the outlier is far from them. The second approach implies that the cluster of usual data points is dense and wide, while the cluster of outliers is scattered and small. The benefit of the latter approach is that it can be applied in unsupervised mode. It is easy to change with the incremental model. Therefore, it can be modified to other complicated data types, such as the data stream, which can be processed and can manage data. It also has a fast test step because the number of clusters for comparison is small. The drawbacks of this method are as follows:

- The performance of cluster-based methods relies heavily on the clustering algorithms in processing the data points;
- Most techniques for the identification of outliers are cluster by-products and thus not ideally used to identify outliers;
- Several algorithms for clustering force each instance to be allocated to a cluster. Abnormalities may result in a large cluster being allocated, and techniques that function under the presumption that anomalies are not included in either cluster could then regarded as normal data points;

- Some clustering algorithms demand that a cluster be allocated to each case. Therefore, outliers may be connected to a wide cluster with methods that are often isolated from the outliers as a standard case;
- Some clustered approaches only operate where outliers are not a significant component in clusters; and,
- Measuring the clustering algorithm is challenging to compute.

2.6 Research Challenge and Objective

Big data is a significant aspect of our everyday lives. A large amount of data represents a lot of information to be analyzed. The disciplines most involved in the extraction of knowledge from large datasets are data science and machine learning. With the increasing need to extract and interpret data streams, traditional outlier detection algorithms cannot efficiently handle the data. The local outlier factor (LOF) is a prominent algorithm used for finding local anomalies that distinguishes between the outlier case and the normal case. The method by which outliers are identified is a significant issue in big data processing via a data stream.

The LOF's key problem is that all the datasets and distance values in the machine need to be retained in storage memory [10]. Another issue concerns the processing of the data stream. The algorithm needs to reprocess the measurement from the beginning of any change in the dataset. The ILOF algorithm is the updated LOF variant that was introduced as the first step in the stream environments procedure. However, to calculate the LOF score for each data point at different times, all data points in the machine memory must always be kept in the ILOF algorithm. Thus, ILOF requires considerable amounts of time and memory. Therefore, a new approach for processing the LOF vulnerabilities in the data stream needs to be developed. Accordingly, each new method can achieve the key objective of calculating the LOF score by considering the following circumstances (as set out in [19]): (1) a portion of the dataset is stored in computer memory; (2) no previous knowledge regarding the distribution of data as outliers is detected; (3) the algorithm does not have any knowledge regarding future data points when it detects an

outlier using the current dataset; and (4) the algorithm should check for an incoming data point to determine if it is either normal or an outlier. As discussed in Chapter 1, this research aims to establish GP-LOF for data streams as a new local outlier algorithm. The principal objective of this research is performed under the following circumstances as described above. The six detailed research questions mentioned in Section 1.5 are guided by these four conditions, which in turn directed the development of the algorithm and the experiment.

2.7 Conclusion

This literature review aims to provide a recent approach to the local outlier detection methods in big data. Local outlier detection techniques discussed in this literature can be applied to detect outliers in two different contexts. Nonetheless, these approaches have benefits and drawbacks. For example, in the k-nearest-neighbor approach, the majority of these local outlier detection techniques use an unsupervised model and process with no prior assumptions about the underlying data distribution. However, this takes a long time to compute. The cluster-based approaches can be modified to handle more complex data types, like data streams. However, the processing time is highly dependent on the clustering method used, which may necessitate a longer processing time.

Due to the infinite and dynamic nature of data streams, the primary concern is determining how to apply the local outlier when dealing with data streams. Some techniques for detecting local outliers store all historical data in memory or in a secondary storage location. When dealing with new data points, some algorithms outperform the ILOF algorithm. This appears to be a challenge in the case of several of these novel approaches. One issue is how to manage memory consumption while summarizing previous data points and processing new data points. This could have an effect on these algorithms' performance. It will be interesting to conduct a research to address these issues to improve the efficiency of detecting local outliers. Outlier detection aims to find the outlier among the data points. This chapter summarizes recent advances in the detection of local outliers in both static and stream environments. It focuses on the issues surrounding local outlier detection in the stream environments and considers how the LOF algorithm can be improved to be better suited to detecting local outliers in data streams. We

hope that this literature review is helpful to readers who want an overview of local outlier algorithms and alternative methods for implementing the local outlier factor (LOF) algorithm in the most common big data approach.

Chapter 3: Benchmark Datasets Used in This Research

"A Grid Partition-based Local Outlier Factor for Data Stream Processing" Forthcoming in Proceedings of the 4th International Conference on Applied Cognitive Computing, 2020. Springer

Alsini, R., Almakrab, A., Ibrahim, A. and Ma, X., 2021. Improving the outlier detection method in concrete mix design by combining the isolation forest and local outlier factor. *Construction and Building Materials*, *270*, p.121396.

3.1 Introduction

The volume of data has grown significantly, both for research and in the context of daily life. People can use different types of technology in the digital world to capture and distribute big data, which has the characteristics of volume, velocity, variety, value, and veracity [108,109]. The volume factor refers to the vast amount of data that has been collected and analyzed. The velocity factor refers to the rate at which data is produced and transmitted between various systems and devices. The variety factor employs a broad range of data types that can be used to obtain the necessary information or output. It includes data forms such as structured, unstructured, and semi-structured data [110]. The value factor has advantages when it comes to extracting information from big data. Finally, the veracity factor considers the accuracy, trust, security, and reliability of the data. More specifically, big data represents a huge amount of undiscovered information and expertise. A data stream is a common form of big data that embodies the five Vs. Because of the essence of the data stream across all five significant data components, data stream processing employs a variety of methods to analyze the data points in the data stream environment. The real-time feature of the data stream requires corresponding technologies for efficient data processing. How the data is processed determines how information can be extracted from the data stream.

3.2 Data Stream Processing

Processing the data stream is a crucial subject in the field of information flow processing (IFP)[111]. The goal of data stream processing is to understand the data action and to acquire the right information to make better decisions. It is important to analyze the data stream in the workflow and then store it. Many places produce the data stream, such as a stock exchange, monitors, log operation, social media,

etc. Processing the data stream is thus helpful and significant. Two types of strategies may be used for processing the data stream as [112]:

I. Apply the query processing.

II. Apply the data stream mining.

Various articles presented a promising performance in both areas, which also highlights the relevance of data stream analysis in the Big Data era. Managing the data stream can be done using queries as a primary method, such as the Structured Query Language (SQL). SQL is a common language use for managing the database. The data stream management system (DSMS) uses an extended version of SQL known as the Continuous Query Language (CQL). The reason behind CQL is to ensure any continuous data over time can be used on the system[7]. The other approach is processed under data mining as part of data science to discover knowledge in data. Data stream mining usually involves data mining or machine learning to extract and predict new information. The most used methods are clustering, classification, and data stream mining on the outlier detection.

3.3 Outlier Detection in the Data Stream Mining Approach

Outlier detection is a significant research issue in machine learning and data mining for detecting a rare object in real applications, such as in the fields of finance, industry, health, and materials science. The core concept of outlier detection is to identify abnormal data that is different from the majority of the data. Outliers can be divided into two categories: global outliers and local outliers. In [113], the authors explain how to locate a global outlier by using a sliding window technique. For local outliers, the distance between points is usually computed according to the local neighborhood, known as the k-Nearest Neighbors (KNN) algorithm. Many approaches of outlier detection in the data stream have been categorized, such as distance-based outlier method, density-based outlier method, clustering-based outlier method, and ensemble-based outlier method [13,14,29,114].

The distance-based outlier method is evaluated depending on the distance between the data points. Authors in [6] introduced the distance-based method to calculate the distance between points and their

nearest neighbors. Those with a far distance from the neighbors are counted as an outlier [3]. k-Nearest Neighbors (kNN) is the most common method used to evaluate the outliers based on their local neighbors. In the data stream, sliding windows have several methods applied in the distance-based model. In [115] and [116], the authors utilized a sliding window technique to uncover the global outliers regarding the current window. In [117], the authors enhanced the algorithm in [3] by introducing a continuous algorithm that has two versions to reduce the time complexity and memory consumption. The first version handles multiple values of k neighbors. The second version involves reducing the number of distances by using the micro-cluster for computation.

The density-based outlier method is based on measuring the densities to their local neighbors. Those densities that are far from their closest neighbors are considered an outlier. The local density measures the density-based method. LOF is an example of the density-based model that uses (kNN) to detect the data points using the local reachability density. LOF has a high detection accuracy and has several proposed methods in improvement, some of which are discussed in [21] and [31]. The author in [118] used the LOF to distribute data points into several grids. Since LOF works in a static environment, an incremental local outlier factor, i.e., ILOF, is presented using the LOF in the data stream. All the previous extensions in the LOF required the entire data points to be calculated to get the LOF score, which is not necessary for the ILOF technique since it can handle new incoming data points.

The clustering-based outlier method is an unsupervised method that processes data points by dividing the data into groups based on their distribution [17]. The method aims to cluster a data point and then to detect the outlier [119-121]. Some algorithms used the small cluster by representing a small number of points as an outlier, while other methods used the threshold to find the outlier. Several methods used the cluster in the detection of the outlier, such as partition cluster model, density cluster model, and grid-based cluster model [79]. The authors in [122] proposed using the cluster in a high data dimension data stream. k-means is used in the data stream to split the data into several segments to process. Authors in [123] generated histograms for the clusters in the streamed data, which they used later for data mining and in detecting outliers. In [124], the authors proposed an algorithm that works on outlier detection on

a data stream that can determine if any point is an outlier within any time. The accuracy of detection depends on the availability of time. In [125], the author used the data stream arriving time to learn and determine the normal behavior of the current period.

The Ensemble-based outlier detection is a new strategy for detecting outliers by combining the one set of results with the results of another outlier method to present and create a robust outlier detection method. In [126], the authors proposed combining the Ensemble-based outlier detection with the density-based outlier detection to solve both IF and LOF limitations. The developed method focused on clipping data points to detect the outlier according to a set of candidate outliers. Several authors have introduced extended versions of the IF to deal with several types of outliers. In [127], the authors improved the IF by using a new algorithm called Function isolation Forest (FIF) to identify outliers. In [128], the authors used the sliding window concept for data stream in the IF method.

3.4 Benchmark Datasets

The label information's specifics are not processed in an unsupervised outlier approach due to the need to be measured and contrasted for practical purposes. When designing and developing a new outlier detection algorithm, it is common practice to validate the results with available public datasets against better known unsupervised outlier detection algorithms such as the LOF algorithm. In the UCI machine learning repositories, there are many classification data sets fully accessible in [129]. Furthermore, some outlier detection datasets are included in [130]. The following datasets are real-world datasets containing outlier data points. These benchmark datasets have been used to test existing algorithms by using the proposed modern algorithm such as GP-LOF, GP-LOFR, and IFS-LOF algorithms.

3.4.1 UCI Vowel Dataset

The Vowel dataset is considered to be a multivariate time series and a classification dataset that categorizes speakers. In one case, two Japanese vocals were spoken by nine speakers. One speaker makes up a time series between 7 to 29 lengths, with twelve characteristics of each point of a time series. In the training dataset, every structure is seen as a single point for outlier detection, while in the UCI

learning repository, the frame talk is seen as a single point. Classes six, seven and eight are commonly referred to as UCI Vowel Data set as Inliers. This data set consists of 12 dimensions, including 1,456 data points, which constitute 3.4% of the outlier.

3.4.2 UCI Pendigit Dataset

The pendigit dataset originates fromthe UCI machine learning repository, as given in[129,130]. This data set is multiclass of 16 dimensions and ten classes. It consists of 25 samples that are generated from 44 authors. Thirty samples of the author are used for the training process, while the remaining 14 are used for the testing process. The initial training set consists of 7,494 points, and the test set includes 3,498 points.

3.4.3 UCI Shuttle Dataset

The UCI shuttle dataset consists of multiple classifications that have nine dimensions provided from the UCI machine learning repository at [129,130]. This dataset combines the output from both the training and testing process. Classes two, three, five, six, and seven are grouped to form the outliers. This dataset contains 49097 data points which constitute 7% of the outlier.

3.4.4 KDD Cup99 SMTP Dataset

The KDD CUP 1999 dataset, which is a UCI machine learning repository, uses its SMTP service[129]. The original dataset (KDDCUP99) has 4,898,431 data points, 3,925,651 data points (80.1 percent) being considered as an assault. 976,157 data points, including 3,377 attacks (0.35%), are forged for a smaller dataset. This smaller dataset is used for the generation of the SMTP KDDCUP99 data set. The data collection KDDCUP99 SMTP consists of three dimensions and contains 0.03% outliers and 95,156 data points.

3.4.5 UCI Concrete Dataset

The concrete data collection was obtained from the University of California Irvine (UCI) machine-learning repository that was released in [129]. The data collection included the results of the compressive strength of 1030 concrete mixtures. Compressive strength is considered one of the extremely important

parameters used by the engineering community in structural concrete design, including structures, bridges, etc.

3.5 Conclusion

This chapter addressed the background of processing the data stream and outlier detection approaches. A new unsupervised local outlier algorithm is validated by benchmarking dataset that is a real-world dataset. After that, the results will be compared to the common local outlier algorithm in the data stream process. The datasets used in this dissertation are also discussed in depth.

Chapter 4: A Grid Partition-based Local Outlier Factor for Data Stream Processing

"A Grid Partition-based Local Outlier Factor for Data Stream Processing" Forthcoming in Proceedings of the 4th International Conference on Applied Cognitive Computing, 2020. Springer

4.1 Introduction

The demand for data is increasing in the big data era. Massive data is generated through various sources such as smart homes, sensors, mobile applications, communication, and finance, which all lead to an increase in data processing. One of the challenges in big data processing is how to measure outliers in streaming data. Data streams change every second, and the size is potentially infinite because it keeps increasing constantly. As a result, it is difficult to store the entire data sets in the memory and process them with older algorithms [1]. The density-based method is a well-known method used to find the outlier in the multi-density data. Local Outlier Factor (LOF) is currently the most utilized density-based method. It handles the data without assuming any underlying distribution. Moreover, it is capable of finding the data set in the data with heterogeneous densities [55,131,132]. However, LOF faces some limitations in data stream processing. First, it works only on static data that does not change over time, and only scans the data at one time to process the whole data set. Also, for any change in the data points that occurred by adding or deleting data points, LOF needs to be recalculated on the whole data set. Because of the limitations of LOF, it can't be used in data streams as the size of data streams are potentially infinite, and the data are changing over time.

To further improve outlier detection and performance accuracy in the data stream, we propose a new technique called Grid Partition-based Local Outlier Factor (GP-LOF). The proposed method has the following characteristics. First, GP-LOF works with limited memory. Therefore, a sliding window is used to summarize the points. Second, a grid method splits the data points for the processing phases. Third, it detects outliers using LOF. In this research paper, GP-LOF is evaluated through a series of experiments with real-world datasets. Based on the experimental result, the GP-LOF algorithm has shown better accuracy and execution time than the DILOF algorithm [56]. The rest of this research paper

is organized as follows: Section 2 describes the methodology of the GP-LOF algorithm. Section 3 explains the experimental results. The conclusion is presented in Section 4.

4.2 Methodology and Methods

4.2.1 Grid Partition-based Local Outlier Factor (GP-LOF) Algorithms

In this section, we will explain the Grid Partition-based Local outlier Factor (GP-LOF) algorithm. The main objective of the GP-LOF is to find the outlier by the following characteristics: there is no prior knowledge of the data distribution; a part of the dataset is stored in the memory; the LOF algorithm is applied to detect the outlier. GP-LOF includes three phases to identify the outlier: Preprocessing phase, Processing phase, and Detection phase, as shown in (Figure 4.1). Algorithm 1 explains how the GP-LOF method operates as follows. The GP-LOF algorithm begins by collecting the data points in Pre-Processing Window (*PPW*) with specific window size *ws*. Then, once the size in the *PPW* is complete (line 3-6), the first half of the data points in PPW is selected and then sent to the Processing Window (*PW*). The grid technique is applied in the PW to divide data points in the grid index i. After the data points are partitioned into the grid, GP-LOF calculates the LOF and gets the result depending on a predefined threshold θ. The GP-LOF ensures any points that exceed the θ are removed from the *PW* (lines 7-17). Each phase in the GP-LOF is described as follows:

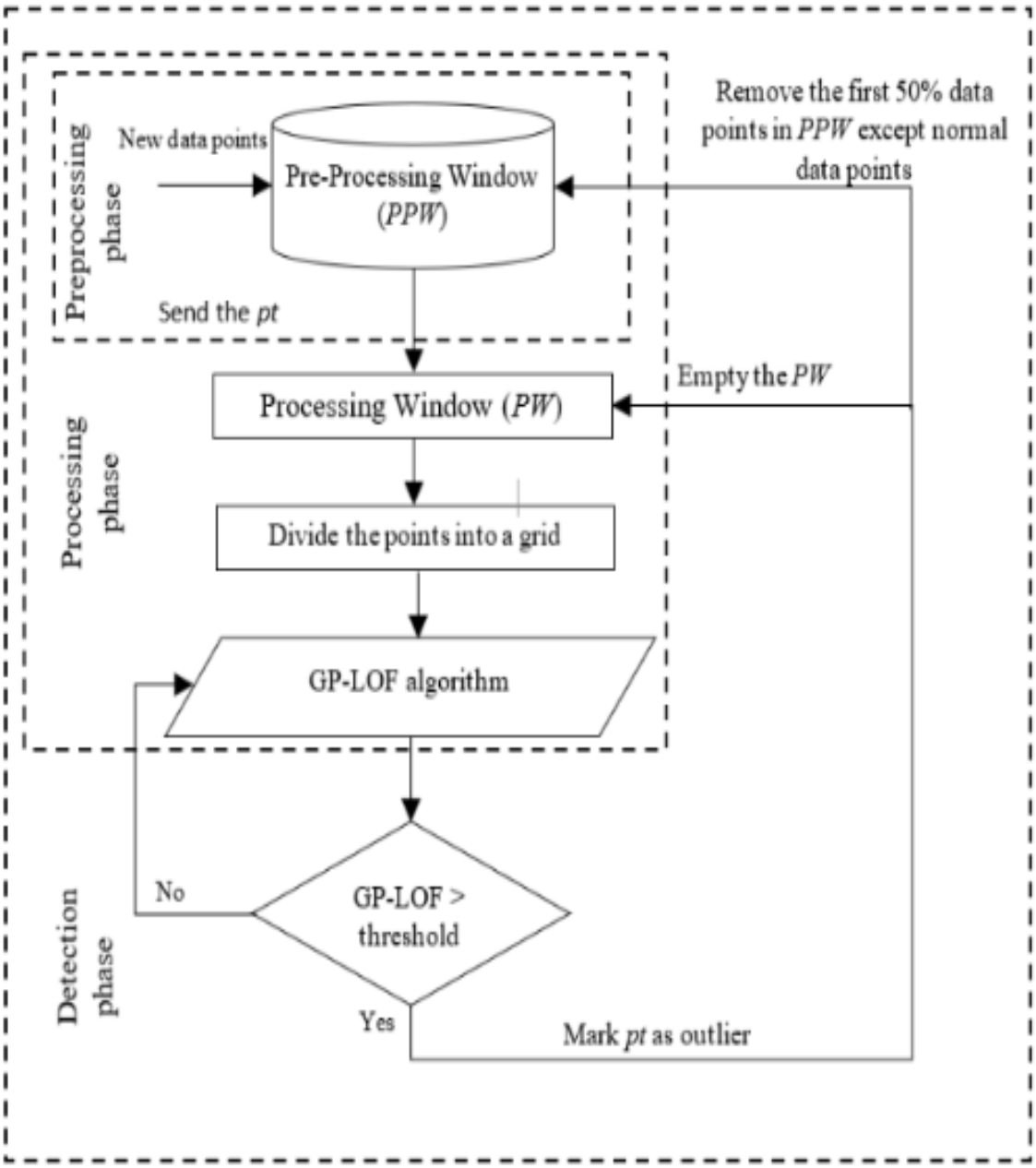


Figure 4.1 Framework of the GP-LOF algorithm in the data stream.

Algorithm 1: GP-LOF

Input: LOF threshold θ
Infinite data streams points $P=\{p_1, p_2, ..., p_t, ...\}$,
Preprocessing Window size P*WS*
processing Window PW
Number of grids *ng*
Grid G
Thresholds of LOF scores Θ,

1 **Init** *PWS* $\rightarrow$ {} // is representing the preprocessing window size
2 **Init** *PW* $\rightarrow$ {} // is representing processing window
3 **For each** *pt* $\in$ *P* **do**
4 **If** *PWS (pt)* < *PW (pt)*
5 **Add** *pt* to *PWS (pt)*
6 continue.
7 **else**
8 **Add** first 50% *PWS (pt) to the PW*
9 **For** *PW* **do**
10 *GP-LOF (pt)*$\leftarrow$ *G(ng,pt,*θ*)*
11 **For** every *GP-LOF (pt)* **do**
12 **If** the LOF_k (*pt*)> Θ then
12 *pt* is outlier
13 **End**
14 **End**
15 Empty the *PW*
16 Remove the first 50% data points from *PWS*
17 **End**
18 **End**

1 Pre-Processing Phase:

In this phase, a *PPW* is used to collect and store the data points that arrive in the stream. Then, once the *ws* of the *PPW* is filled with data points, the GP-LOF selects the first 50% of data points from *PPW* to be processed in the PW.

2 Processing Phase:

The processing phase begins once the data point in the *PPW* is moved to the *PW*. Next, *PW* divides the data points into a grid, which is a number of regions. Then, GP-LOF computes the LOF score for

each grid index i. After that, the LOF scores for each i are used to find the LOF score for all data points belonging to the grid. Then, the LOF results will forward to the detection phase. To compute the outlier score of data points, the *PW* ensures the following steps:

- Divide the dataset's dimensions equally into the grid.
- Allocate each data point p into a grid index i.
- In every grid index i, compute the LOF score for all data points using the LOF algorithm [20].

3 Detection Phase:

Once the LOF result is received in the detection phase, the GP-LOF method will scan all data points and select the data points that are greater than the threshold to be outliers. Then, it empties the *PW* and removes the first 50% of data points in the *PPW* except inlier data points as in Algorithm 1. Then, when any new data point is arriving in a stream, the preprocessing phase starts again to collect them.

4.3 Experiment Procedures

4.3.1 Datasets Used in Experiments

This section provides the experimental results of the GP-LOF algorithm. The GP-LOF results are compared with DILOF results in various datasets. The GP-LOF algorithm is implemented in Java by a machine that runs in intel® core (MT) i7- 4940MX CPU, 16GB RAM, 1 TB SSD hard disk, and Windows 10 (64-bit) operating system. The same machine implements the DILOF algorithm in C++, and the source code is available in [56]. For the DILOF setting and hyperparameter, the reader can refer to [56]. Both methods are measured under two metrics in the accuracy of detecting the outlier and execution time. In particular, the area under the ROC curve (AUC) is used in the first category as described in [130,131]. The efficiency of the AUC is evaluated by applying the True Positive Rate (TPR) and a False Positive Rate (FPR) with a scale of the threshold t = { 0.1, 1.0, 1.1, 1.15, 1.2, 1.3, 1.4, 1.6, 2.0, 3.0}. The evaluation of both TPR and FPR rate is tested according to each threshold value to obtain accuracy. Both GP-LOF and DILOF methods are tested with different window sizes ws through a real-world data set, as in (Table 4.1). All these real-world datasets can be obtained from the Machine

Learning database repository at UCI in [129]. The UCI Vowel dataset can be obtained from [133]. The UCI shuttle dataset is provided in [129]. Both the UCI Vowel dataset and the UCI shuttle have been modified to be suitable for data stream format like in [133]. In the KDD cup 99 SMTP, we did the same configuration as [134] to be an outlier. SMTP is a subset from KDD Cup 99 dataset developed to test intrusion detection in the network. In the SMTP service, it is possible to show some changes in distribution within the streaming series [135].

Table 4.1 Real-World Dataset

Datasets	**Number of data points**	**Dimension**	**Class**
UCI Vowel dataset	1456	12	11
UCI Pendigit dataset	3,498	16	10
UCI Shuttle dataset	58000	9	10
KDD Cup99 Smtp dataset	95,156	3	unknown

For the experimental setup, we set k to be 19 for the UCI vowel dataset. The UCI Pendigit dataset has k set to 18. The UCI shuttle has a k of 29. We set k to 8 in the KDD CUP99 SMTP. Both GP-LOF and DILOF have the same setup of the window size *ws* for their validation by having two categories of *ws*. In the DILOF, we setup the *ws* in the summarization phase *ws*={100,120,140,160,180,200} for both the UCI Pendigit dataset and UCI Shuttle Dataset. The remaining datasets, including the UCI vowel dataset and the KDD CUP99 SMTP dataset, both datasets are tested by *ws*={100, 200, 300, 400, 500}. For the UCI Pendigit dataset and UCI shuttle dataset, both methods are evaluated by *ws*={100,120,140,160,180,200}. For the GP-LOF method, we did the same configuration on the *ws* in the Preprocessing Windows PPW.

4.3.2 Experiment Discussion

4.3.2.1 Accuracy of the Outlier Detection:

The accuracy of outlier detection is tested under the AUC through a series of experiments with different window sizes. Figures 4.2 to 4.5 and Tables 4.2 and 4.3 represent the accuracy between the GP-LOF and DILOF for the UCI Vowel dataset, UCI Shuttle dataset, UCI Pendigit dataset, and KDD99 SMTP Dataset. In the UCI vowel dataset, DILOF has high accuracy in the size of the window *ws*=100 at 72.02% and in *ws*=200 at 89.038% compared to the GP-LOF algorithm, which takes 43.00% and 45.49%. However, when the size of the window is increasing, GP-LOF has better accuracy. For example, when the size of the window reaches *ws*= {300} or {500}, the GP-LOF outlier accuracy is better than DILOF. Both algorithms have a similar accuracy rate when the size of the window reaches *ws*= {400}. We noticed the GP-LOF algorithm performs better accuracy in all window sizes for the UCI shuttle dataset than the DILOF algorithm. The highest accuracy rate for the GP-LOF algorithm reaches the accuracy rate at 84.86% at *ws*={200}, while the DILOF algorithm reaches the highest accuracy rate as *ws*={160} at 71.03 %. For the UCI Pendigit dataset, the GP-LOF algorithm has a lower accuracy rate of 49.12% at *ws*={100}. When the window size increases, all accuracy rate of the GP-LOF algorithm has a higher accuracy rate than the DILOF algorithm. The highest accuracy rate achieves 61.41% compared to the DILOF algorithm at 52.56% at *ws*={200}. In the KDD99 SMTP dataset, there is a variance detection accuracy between GP-LOF and DILOF. For example, when the windows size *ws*> 300, there is a significant difference between GP-LOF and DILOF algorithms. The GP-LOF rate keeps increasing, while the DILOF rate has a lower accuracy when the size of the window keeps increasing.

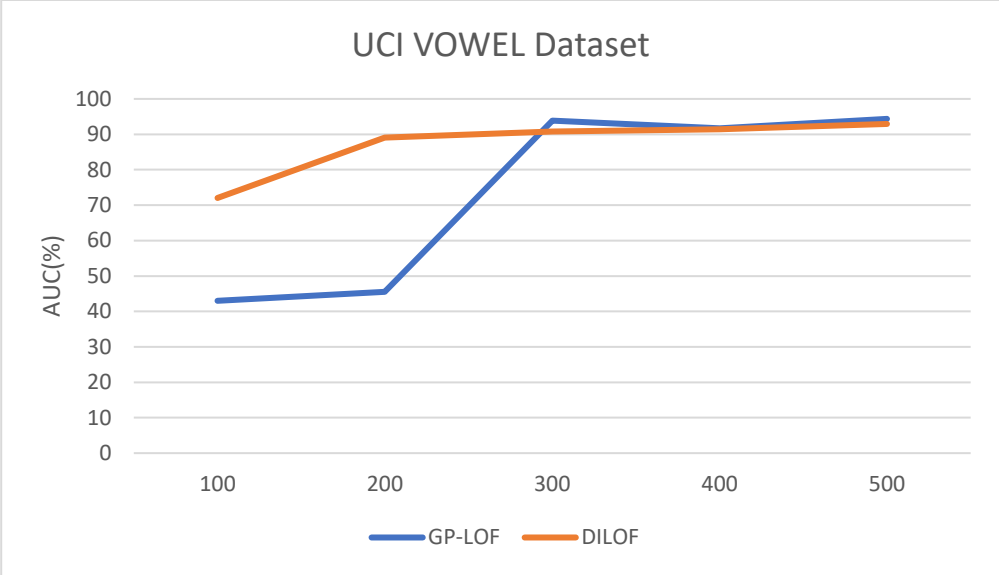


Figure 4.2 GP-LOF shows consistently higher accuracy in all the window sizes compared to the DILOF algorithm in the UCI Vowel Real-World dataset.

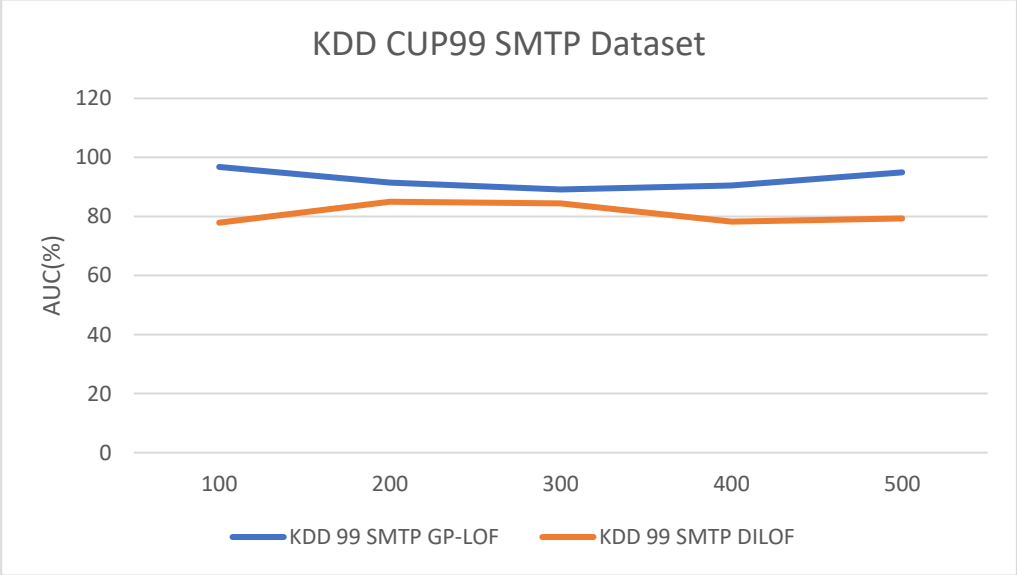


Figure 4.3 GP-LOF shows consistently higher accuracy in all the window sizes compared to the DILOF algorithm in the KDD Cup99 SMTP Real-World dataset.

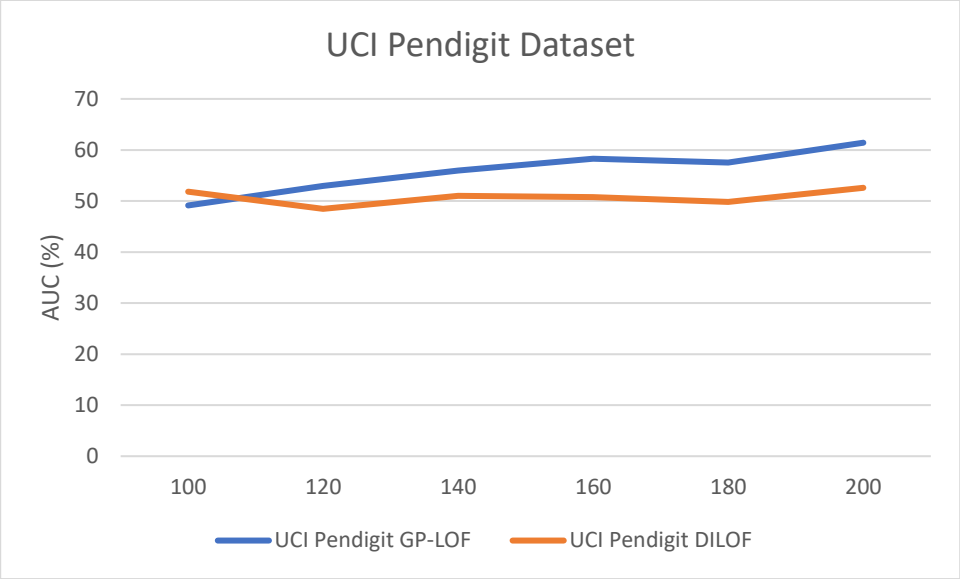


Figure 4.4 GP-LOF shows consistently higher accuracy in all the window sizes compared to the DILOF algorithm in the UCI Pendigit Real-World dataset.

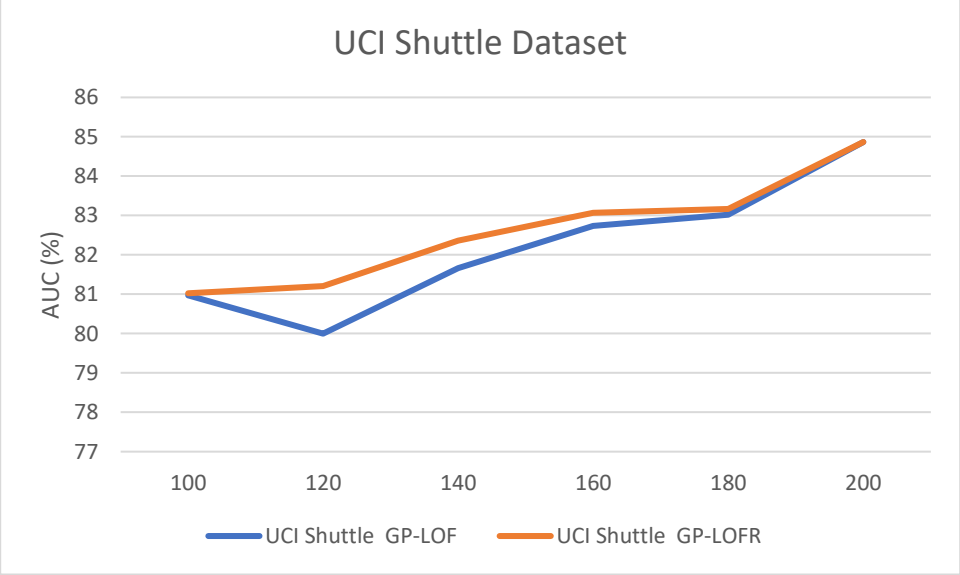


Figure 4.5 GP-LOF shows consistently higher accuracy in all the window sizes compared to the DILOF algorithm in the UCI Shuttle Real-World dataset.

Table 4.2 Accuracy result for the UCI vowel dataset and KDD Cup99 SMTP dataset between GP-LOF And DILOF algorithms

Window Sizes	UCI Vowel		KDD Cup99 SMTP	
	GP-LOF	DILOF	GP-LOF	DILOF
100	43.0016	72.037	96.79145	77.9116
200	45.49206	89.0384	91.46393	85.0027
300	93.8462	90.825	89.15365	84.484
400	91.68576	91.463	90.50625	78.2639
500	94.36069	92.9275	94.98262	79.3717

Table 4.3 Accuracy result for the UCI Pendigit dataset and The UCI Shuttle dataset between GP-LOF And DILOF algorithms

Window Sizes	UCI Pendigit		UCI Shuttle	
	GP-LOF	DILOF	GP-LOF	DILOF
100	49.129	51.834	80.9710	69.205
120	52.97	48.458	79.9981	70.6869
140	55.966	51.028	81.6585	69.0178
160	58.258	50.73	82.7370	71.0363
180	57.56	49.839	83.0175	70.964
200	61.418	52.58	84.8658	69.7915

4.3.2.2 Execution Time

Figures (4.6 to 4.9) and table 4.4 and table 4.5 represent the execution time for the UCI vowel, KDD Cup99 SMTP, UCI Pendigit, and UCI shuttle between GP-LOF and DILOF algorithms. We notice that the GP-LOF execution time is always much lower than the DILOF algorithm, even when the size is increasing. In the UCI Vowel Dataset, GP-LOF took almost 1.17 seconds, while the DILOF execution time took almost 4.32 seconds. This is because GP-LOF divides the points into several grids, which reduces the execution time. The same thing occurs for KDD99 SMTP, UCI Pendigit, and UCI Shuttle datasets. In KDD99 SMTP, GP-LOF execution time in the *ws*= 500 took almost 56.396 seconds, while

in the DILOF algorithm, it took almost 260.544. The DILOF executive time took 17.918 seconds for the UCI shuttle, which took more time than GP-LOF at 6.97 seconds. In the UCI Pendigit, GP-LOF performs higher in the execution time at 2.02 seconds than the DILOF algorithm at 1.35 seconds when the *ws*={100}. When the execution time for the *ws* is increased, we notice that the GP-LOF algorithm performs better execution time for the remaining *ws*={140,160,180,200}.

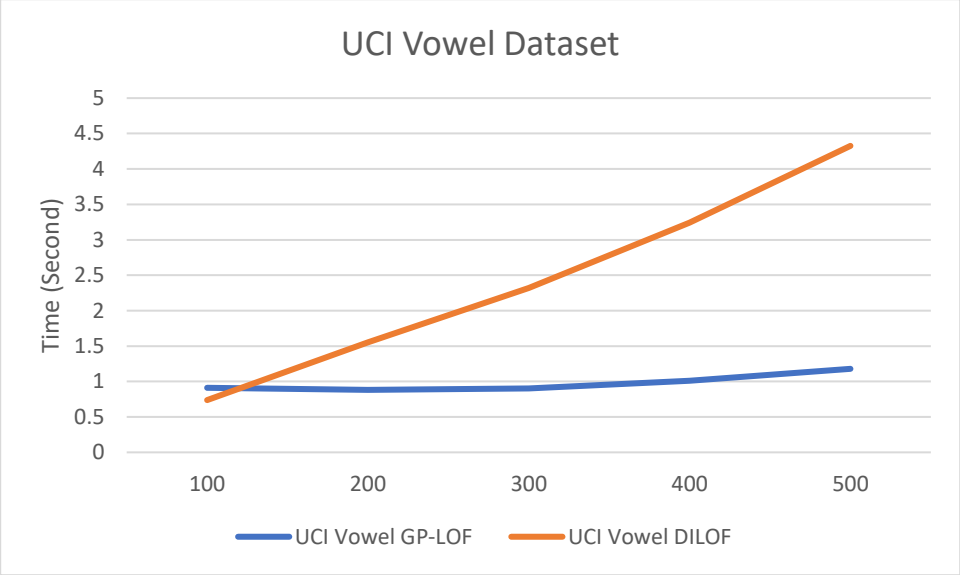


Figure 4.6 GP-LOF execution time in the UCI Vowel Real-world data set for all window sizes is better performance than the DILOF algorithm.

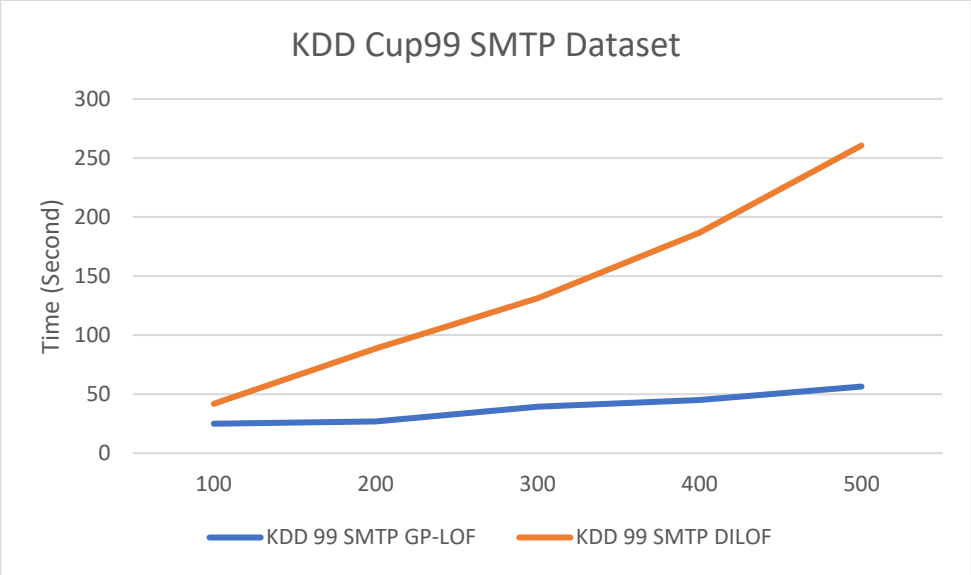


Figure 4.7 GP-LOF execution time in the KDD Cup99 SMTP Real-world data set for all window sizes is better performance than the DILOF algorithm.

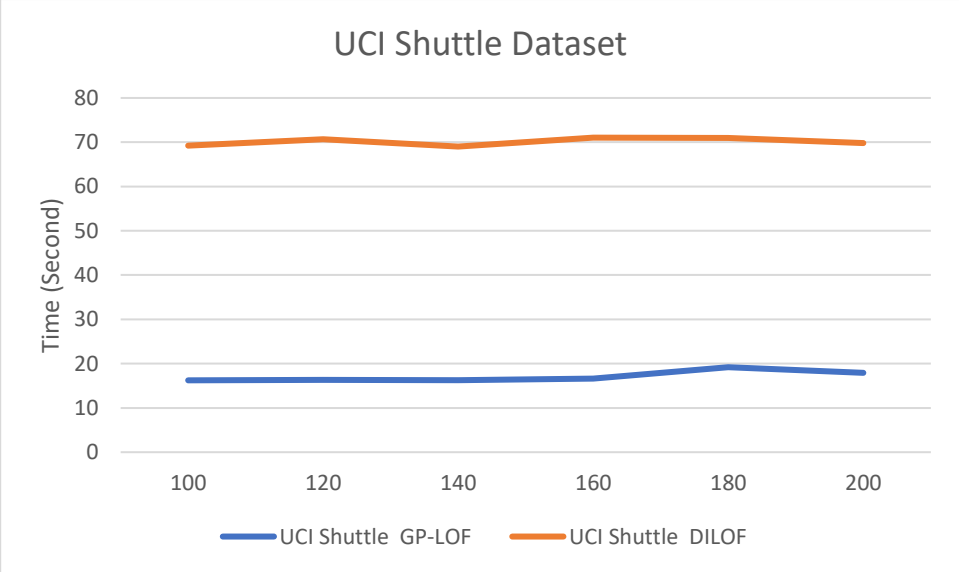


Figure 4.8 GP-LOF execution time in the UCI Shuttle Real-world data set for all window sizes is better performance than the DILOF algorithm.

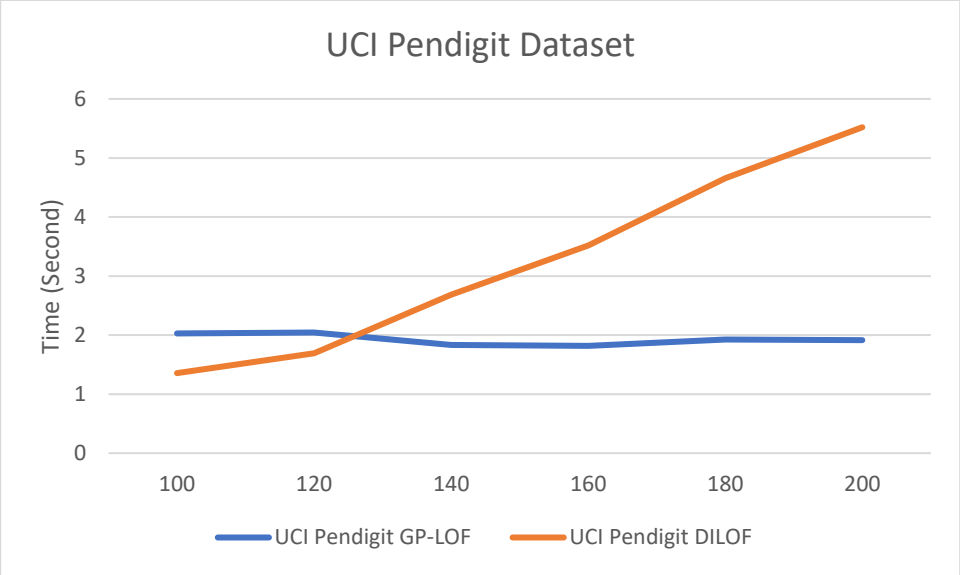


Figure 4.9 GP-LOF execution time in the UCI Pendigit Real-world data set for all window sizes is better performance than the DILOF algorithm.

Table 4.4 Execution time for the UCI Vowel dataset and KDD Cup99 SMTP dataset between GP-LOF And DILOF algorithms

Window sizes	UCI Vowel		KDD Cup99 SMTP	
	GP-LOF	DILOF	GP-LOF	DILOF
100	0.91	0.737971	24.91	41.7741
200	0.882	1.552	26.95	88.7286
300	0.903	2.32008	39.296	131.205
400	1.01	3.24402	44.879	186.835
500	1.179	4.32603	56.396	260.544

Table 4.5 Execution time for the UCI Shuttle dataset and UCI Pendigit dataset between GP-LOF And DILOF algorithms

Window sizes	UCI Shuttle		UCI Pendigit	
	GP-LOF	DILOF	GP-LOF	DILOF
100	16.225	69.205	2.029	1.3555
120	16.356	70.6869	2.043	1.6896
140	16.293	69.0178	1.831	2.681
160	16.65	71.0363	1.817	3.5207
180	19.194	70.964	1.922	4.6571
200	17.918	69.7915	1.916	5.5192

4.4 Conclusion

This research paper aims to solve the problem of detecting the outlier in the data stream. LOF is one of the algorithms that detect outliers in static data, but it has limitations when dealing with data streams. First, it consumes a lot of memory as the whole data set needs to be stored in the memory, which isn't applicable in the data stream as the data size is infinite. Next, it needs to process the whole data set since any change in the data requires that the LOF be recalculated from the beginning, which isn't applicable in the data stream as the data is changing. We propose a novel algorithm called Grid partition local outlier factor (GP-LOF), which overcomes the two limitations of the LOF in the data stream. Our experimental evaluations demonstrate that GP-LOF has better accuracy detection and execution time with several real-world datasets than the state-of-the-art DILOF algorithm.

Chapter 5: A Grid Partition-Based Local Outlier Factor by Reachability Distance for Data Stream Processing

"Grid Partition-Based Local Outlier Factor by Reachability Distance for Data Stream Processing" Forthcoming in the 7th International Conference on Computational Science and Computational Intelligence 2020, IEEE.

5.1 Introduction

There has been significant interest in using data mining and machine learning to enhance outlier detection in big data sets. At present, outlier detection techniques have been used effectively in a number of fields [14]. For example, in network intrusion detection, it is used to identify distinct data points from any other data points [136,137]. These irregular data points reflect uncommon behavior and are considered outliers and potential threats. The density-based method has a remarkable ability to detect outliers in various densities. The Local Outlier Factor (LOF) is a popular technique that can process a dataset without any previous knowledge of the data distribution [20]. Furthermore, the LOF can process datasets with heterogeneous densities [78,136,137]. However, the LOF algorithm design has a weakness in the analysis of streaming data. The LOF executes in a static environment. Memory becomes a limiting factor because LOF must keep and process the whole dataset, which grows significantly with new data points [18]. The LOF is also constrained because any update of the data set requires recalculation on the whole data set. One of the main features of data streams is that the range of the data is infinite [17]. To overcome this challenge, the Incremental local outlier factor (ILOF) applies the LOF algorithm to data streams [77]. Nonetheless, it takes all data points to identify outliers. Several algorithms have been proposed to overcome the ILOF limitations, as in [56,76,138]. The GP-LOF algorithm overcomes LOF's issues by summarizing the data points using the grid method to split the data points and find the outliers using the LOF algorithm [139].In this chapter, we aim to improve the efficiency of the GP-LOF algorithm for the accuracy rate by using another measurement in the LOF; the new algorithm is called Grid Partition-based Local Outlier Factor by Reachability distance (GP-LOFR). The new measure of the LOF outlier score is based on the reachability distance (LOFR), as opposed to the LOF, which uses the local reachability distance.

The GP-LOFR algorithm has the following features: first, it works in limited memory by using a sliding window of data; second, the data points are split using the grid method in the processing phase; and lastly, the detection phase uses the new calculation method of LOFR for detecting outliers. The proposed algorithm was used in a series of experiments with multiple real-world datasets. The GP-LOFR algorithm appears more efficient and has slightly better accuracy than the GP-LOF algorithm. This research paper is arranged as follows: Section 2 describes the Local Outlier Factor by the Reachability distance (LOFR) and GP-LOFR methodology. The discussion results are presented in Section 3. Finally, the conclusion is presented in Section 4.

5.2 Methodology and Methods

5.2.1 The Local Outlier Factor by Reachability Distance (LOFR)

This section describes another calculation method of the LOF, called Local outlier Factor by reachability distance (LOFR). For the outlier score, LOFR works as the LOF algorithm. The only variation between the LOFR and LOF is the local reachability distance for processing the data points pt considering their neighbors [44]. The LOFR does not use the LRD. It uses the Reachability Distance (Reach-Dist) to obtain the LOF score, as shown in Equation 6:

- Definition 6: *LOFR of point pt*

$$LOFR_k(Pt) = \sum_{O \in n_k(pt)} \frac{Reach-Dist(pt)}{\left(\frac{Reach-Dist(o)}{k}\right)} \qquad (6)$$

The LOFR uses the LOF definitions except for the LRD. Instead, it produces a new score of the LOFR, which is illustrated in (Figure 5.1). The LOFR value is taken from the Reach-Dist of the data points (pt) and divided by the average of the Reach-Dist of its neighbors. For more details, the reader can refer to [136,138].

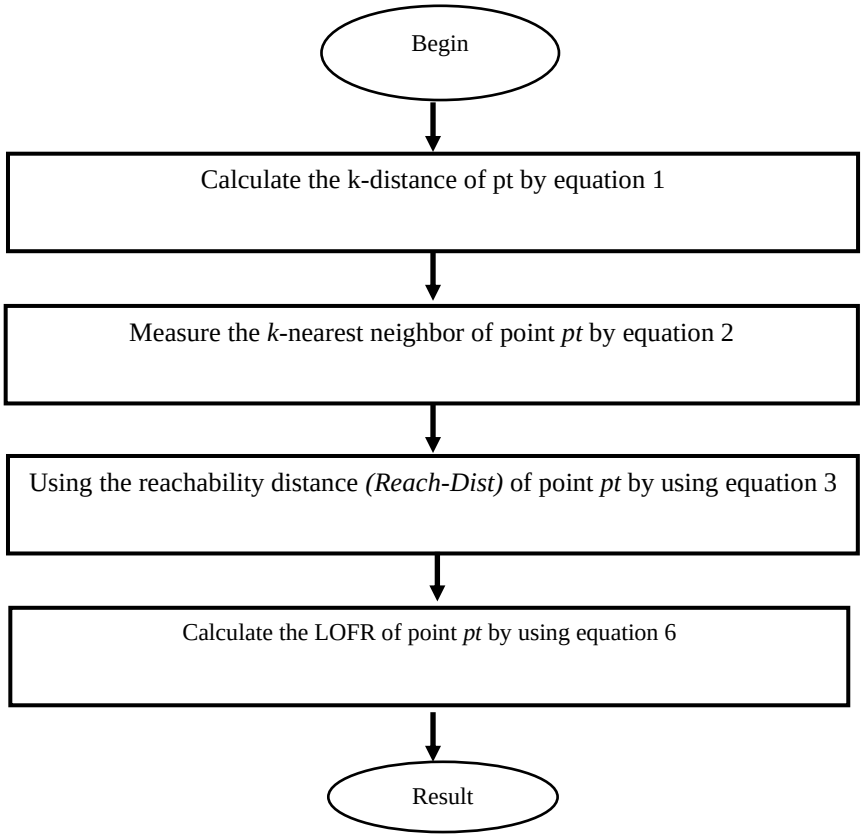


Figure 5.1 The LOFR Flow Diagram in obtaining the outlier score.

5.2.2 The Grid Partition-based Local Outlier Factor by Reachability Distance (GP-LOFR)

This section describes how the GP-LOFR algorithm searches for outliers. There are four main features of GP-LOFR. First, there is no prior knowledge of the data distribution when detecting outliers. Second, only a portion of the dataset is stored in memory. Third, the LOFR algorithm is applied within GP-LOFR to detect outliers. Fourth, GP-LOFR does not have any knowledge regarding future data points when it detects an outlier using the current data set. Like the GP-LOF algorithm, GP-LOFR has three phases: the pre-processing phase, the processing phase, and the detection phase. In the pre-processing phase, a sliding window is used to collect data points from the data stream. The processing phase use the new calculation method of LOFR for determining the outlier score in each grid. In the detection phase, a threshold θ is used to detect outliers.

Algorithm 1: GP-LOFR Algorithm

Input: The LOFR threshold θ
Unlimited data streams points $P= \{p_1, p_2, ..., pt\}$
Pre-Processing Window *PPW*
Processing Window *PW*
Number of grids *Ng*
Grid G

```
1  Init PPW → {} // is representing the Pre-Processing Window
2  Init PW → {} // is representing Processing Window
3   For each pt ∈ P do
4      If PPW (pt) < PW (pt)
5       Add pt to PPW (pt)
6       continue.
7      else
8       Add 50% PPW (pt) to the PW
9       For PW do
10        GP-LOFR (pt)← G(Ng,pt)
11           For every GP-LOFR (pt) do
12            If the LOFR_k (pt)> θ then
12               pt    is an outlier
13            End
14           End
15        Empty the PW
16        Remove the first 50% data points from PPW
          except for normal data points
17     End
18  End
```

Algorithm 1 describes the process of the GP-LOFR and entails the following steps: 1) the size of a window (*ws*) represents the amount of the data points from the Pre-Processing Window (*PPW*), and the PPW collects and stores the data points; 2) a Processing Window (*PW*) selects the first 50% of the data points to be processed from the *PPW* (lines 3-6); 3) the data points are split into an index of the grid, and the GP-LOFR calculates each grid to obtain the LOFR score, based on a threshold θ; 4) the GP-LOFR algorithm checks any point beyond the threshold is excluded from the *PW* (line 7-17); and, 5) when new data points are inserted, the same steps are repeated to calculate the LOFR score.

5.3 Experiment Discussion and Results

In this section, the experimental results of the proposed algorithm (GP-LOFR) are presented. The performance of GP-LOFR was compared to the performance of GP-LOF on four benchmark datasets. The evaluation process for GP-LOFR and GP-LOF were calculated by two metrics: the accuracy of

outlier detection and execution time. In particular, the Area Under the ROC Curve (AUC) is used for the first metric to obtain the accuracy rate [129,130]. All methods were implemented on Java and processed in a machine that operates in intel® core (MT) i7-4940MX CPU, 16GB RAM, 1 TB SSD hard disk, and Windows 10 (64-bit) operating system. Both GP-LOF and GP-LOFR approaches are evaluated with various windows sizes (ws) in a real-world data set, as seen in (Table 4.1). The experimental of both the GP-LOFR and GP-LOF algorithms in the KNN were determined for each of the datasets as follows: For UCI Vowel dataset, k = 19; for the remainder, the UCI Shuttle, K =29, KDD99 SMPT k=8, For the Pendigit dataset, it set the k=18. For the size of the windows (ws) used for all the dataset validation, the GP-LOFR and GP-LOF were determined by the Pre-Processing Windows PPW that has two categories of measuring the WS as described in chapter 4.

5.3.1 Experiment Results

5.3.1.1 The Accuracy of the Outlier Detection

The AUC was used to evaluate algorithms to obtain the outlier accuracy via multiple tests with a range of *ws*. Figure (5.2 to 5.5) presents the AUC results for the GP-LOFR and GP-LOF algorithms on the UCI Vowel Dataset, the KDD Cup99 SMTP Dataset, the UCI Shuttle Dataset, and the UCI Pendigit Dataset. In the UCI Vowel Dataset, the GP-LOFR had a better accuracy result in the *ws*= [100] (49.07%) compared to the GP-LOF at (43.0%). The highest accuracy result was obtained for *ws* = [500]; the GP-LOFR reached an accuracy rate of (95.37%) while the GP-LOF accuracy rate was (94.36%). GP-LOF only had higher accuracy for *ws*= [400], where it reached (91.68%) when compared to the GPLOFR at (89.0%). For the KDD Cup99 SMTP, the GP-LOF performance was better than the GP-LOFR for most window sizes. However, there was a difference based on the *ws*. For example, in the *ws* = [100], both GP-LOFR and GP-LOF accuracy results are close to each other. When the ws was increased to 300, the GP-LOFR accuracy performance was better than the GP-LOF, reaching (91.14%). The remaining window sizes were better for GP-LOFR algorithm. In the UCI Shuttle Dataset, GP-LOFR shows a better result than the GP-LOF for most of the *ws*. For example, the GP-LOFR has a better accuracy result at

81.02 % compared to the GP-LOF algorithm when it reaches *ws*=[100]. For the *ws*=[180], we notice both GP-LOFR and GP-LOF results are close to each other, with the advantage in the GP-LOFR when it reaches *ws*=[200]. For the UCI Pendigit, we notice the different output between the GP-LOFR and GP-LOF algorithm at the beginning of the *ws*. However, when the *ws* is getting larger, both GP-LOFR and GP-LOF are close to each other in the accuracy result.

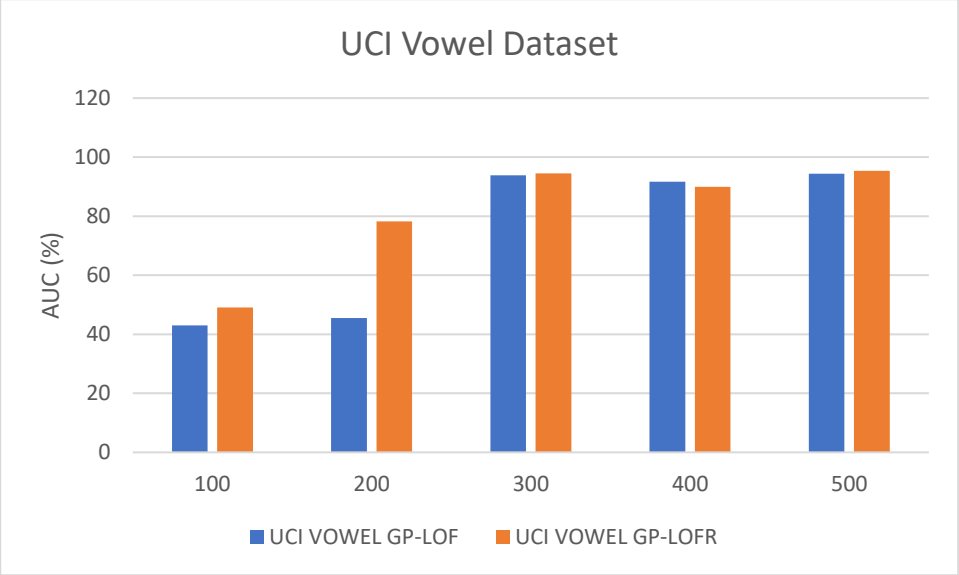


Figure 5.2 Comparison of accuracy result of outlier detection between the GP-LOFR and GP-LOF in UCI Vowel dataset.

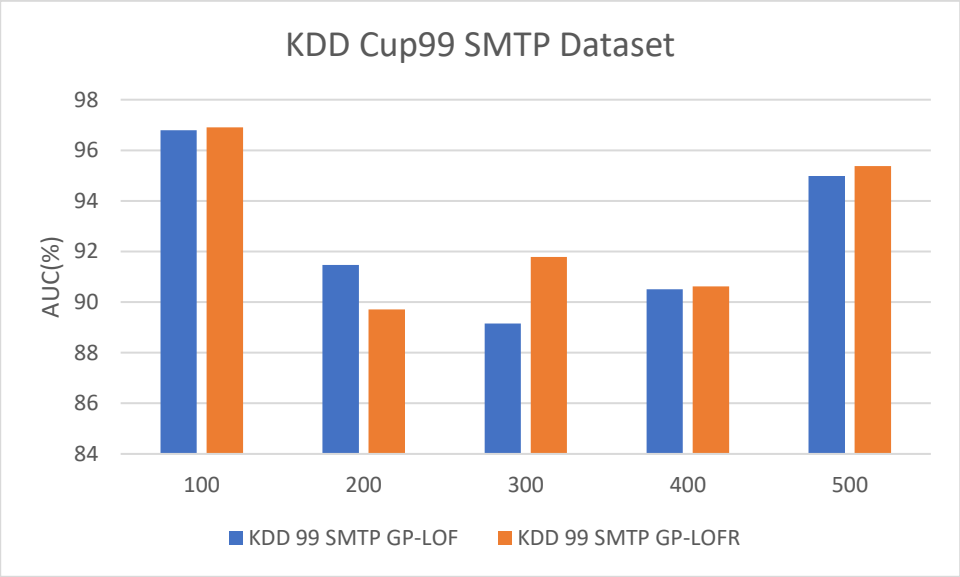


Figure 5.3 Comparison of accuracy result of outlier detection between the GP-LOFR and GP-LOF in KDD Cup99 SMTP dataset.

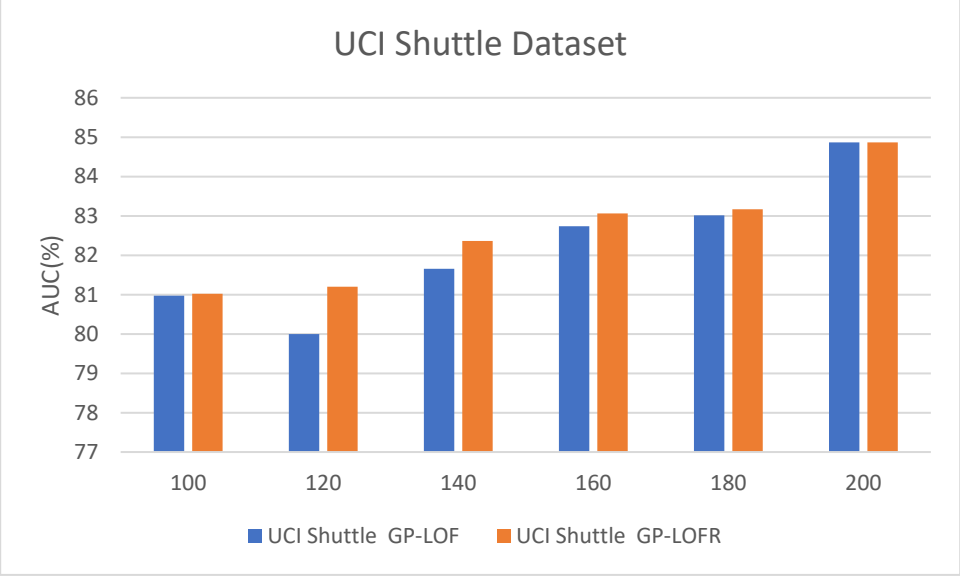


Figure 5.4 Comparison of accuracy result of outlier detection between the GP-LOFR and GP-LOF in the UCI Shuttle dataset.

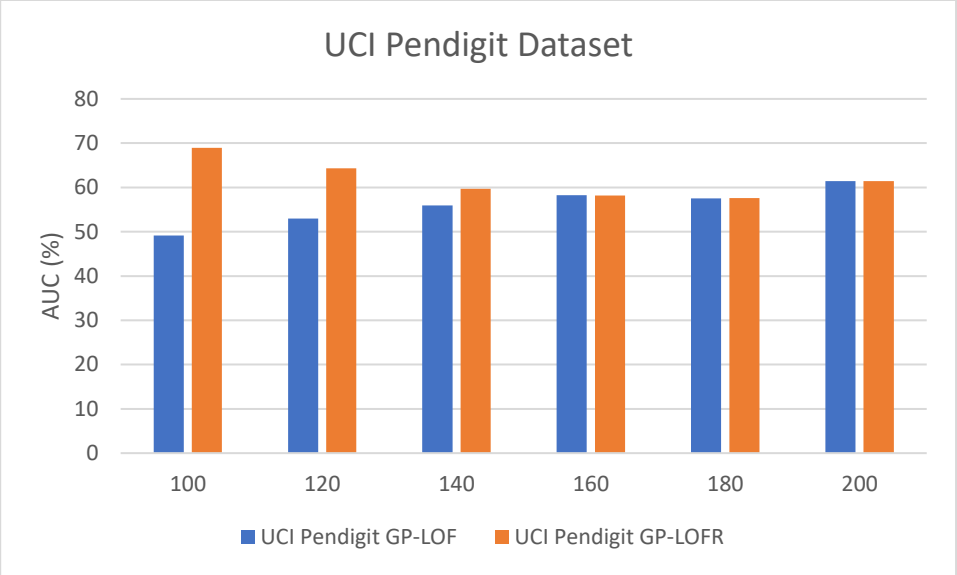


Figure 5.5 Comparison of accuracy result of outlier detection between the GP-LOFR and GP-LOF in the UCI Pendigit dataset.

5.3.1.2 Execution Time

The execution time results for all algorithms are presented for all the datasets, as illustrated in Figures 5.6 to 5.9. The time of execution was measured in seconds for the experiments. Both the GP-LOFR and the GP-LOF are close to each other in most of the *ws*. In the UCI Shuttle Dataset, GP-LOFR took 18.47 to 42.55 seconds, while the GP-LOF took 18.05 to 37.29 seconds. The GP-LOFR took from 0.86 to 1.21 seconds for the UCI Vowel Dataset, and the GP-LOF function took from 0.93 to 1.24 seconds. The GP-LOFR in the KDD Cup99 SMTP took 26.79 to 60.09 seconds, compared to the GP-LOF function, which took 27.18 to 57.82 seconds. For the UCI Pendigit, the GP-LOFR took from 57.56 to 68.94 seconds, while the GP-LOF took from 49.12 to 61.418 seconds.

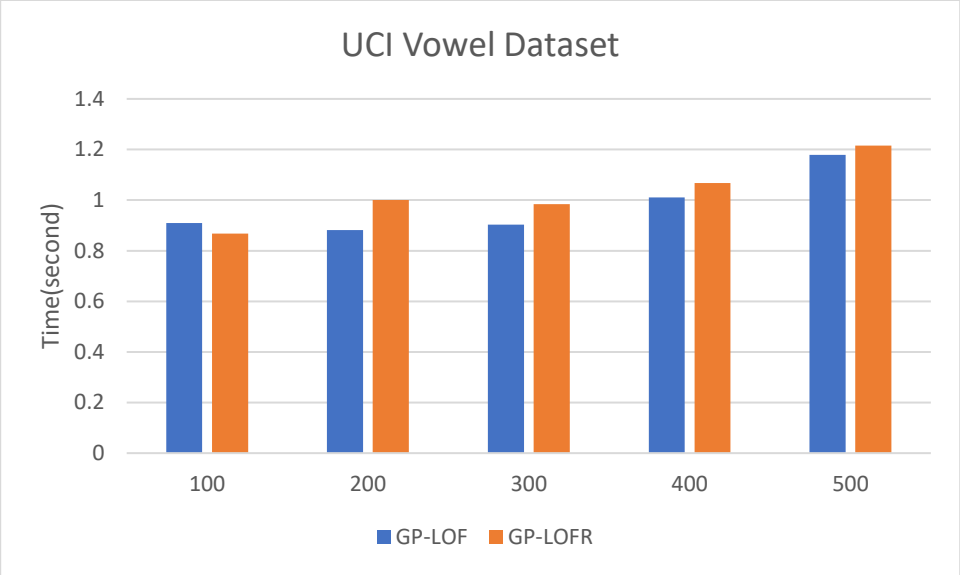


Figure 5.6 Comparison of Execution time between the GP-LOFR and GP-LOF in the UCI Vowel dataset.

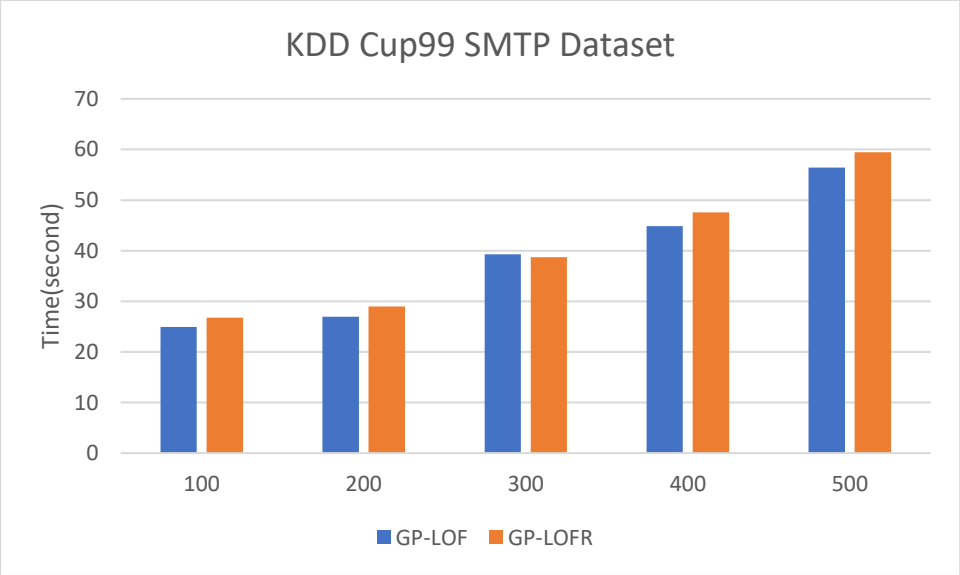


Figure 5.7 Comparison of Execution time between the GP-LOFR and GP-LOF in the KDD Cup99 SMTP dataset.

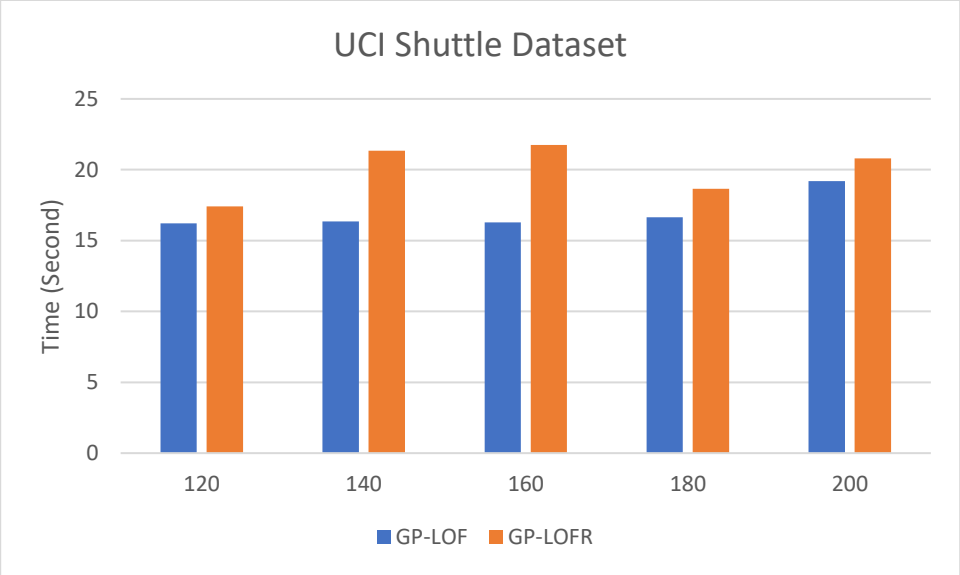


Figure 5.8 Comparison of Execution time between the GP-LOFR and GP-LOF in the UCI Shuttle dataset.

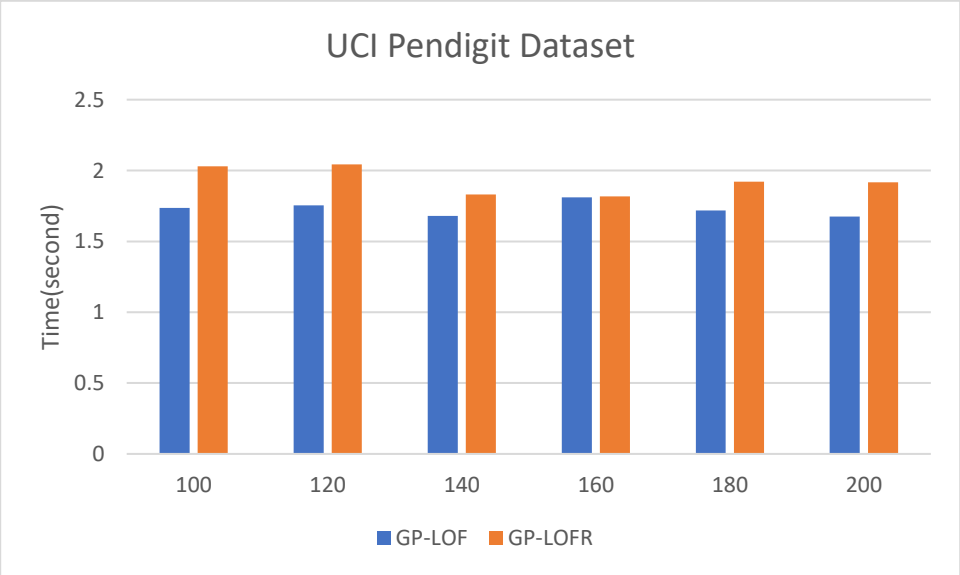


Figure 5.9 Comparison of Execution time between the GP-LOFR and GP-LOF in the UCI Pendigit dataset.

5.4 Conclusion

The main objective of this research paper is to further improve the accuracy of outlier detection on the GP-LOF algorithm when applied to data streams. GP-LOFR addresses the limitation of the LOF algorithm for processing data streams. Like GP-LOF, GP-LOFR works with limited memory by using a sliding window for summarizing the points. Also, it follows the same phrases used in GP-LOF for identifying outliers. GP-LOFR showed slight improvements in accuracy over GP-LOF on several real-world benchmark datasets. This improvement was achieved by using the Reachability Distance (Reach-Dist) to obtain the LOFR score.

Chapter 6: Improving the Outlier Detection Method in Concrete Mix Design by Combining the Isolation Forest and Local Outlier Factor

Alsini, R., Almakrab, A., Ibrahim, A. and Ma, X., 2021. Improving the outlier detection method in concrete mix design by combining the isolation forest and local outlier factor. *Construction and Building Materials*, *270*, p.121396.

6.1 Introduction

Throughout the Big Data era, numerous sources have generated massive data. One of the challenges in big data processing is how to quantify outliers. This is clearly significant in the concrete industry, especially that concrete is considered the second largest usable material on the globe after water. Concrete is a heterogeneous material, and its fresh and mechanical properties depend on various parameters (percentages of ingredients). Generally, the properties of concrete directly influence the stability and reliability of any construction project; these properties include compressive, flexural, and tensile strengths, and elastic modulus [140]. The quality of data in the construction industry may be limited and compromised. For example, field data collection might include some missing values, wrong measurements, or outliers [141]. Due to the efficiency of outlier detection in various areas, many outlier detection techniques have been developed to detect the anomaly known as an outlier.

The distance-based method is the most often used technique for outlier detection. Despite its accessibility, it results in poor accuracy when applied to multi-density data such as concrete mixtures input that have multiple variables. The density-based outlier method deals with multi-density data by the comparison of the density points with nearby local neighbors. The Local Outlier Factor (LOF) is the most practical procedure in the density-based approach [20]. The LOF handles dense data without assuming any underlying or predefined distribution. It also finds the dataset in heterogeneous densities [136,137]. However, the LOF faces some limitations. One limitation is that calculating the distance between points requires a large amount of memory, which affects execution time. In addition, the LOF is incapable of dealing with the sequence of outliers. Another technique for outlier detection, Isolation Forest (IF), solves the issues found in the LOF by isolating the outlier instead of processing the whole dataset. IF is an

unverified learning process for abnormality detection that depends on the principle of separating anomalies. Despite its accuracy, the IF method has a weakness when it comes to a local outlier. Cheng et al. [126] proposed pruning techniques by finding the outlier candidate set to calculate the outlier score. Thus far, this is the approach most successful in solving the limitations of both the IF and LOF methods.

To further improve the accuracy of both the IF and LOF in detecting the outlier, this paper introduces a new method called the Isolation Forest based on Sliding Window for Local Outlier Factor (IFS-LOF) detection. The IFS-LOF merges both methods (IF and LOF) with a sliding window to increase the rate of accuracy and to detect input with different window sizes (*ws*). The IFS-LOF was evaluated through a series of experiments that were performed through concrete mixtures with various ingredients. Based on the experimental results, the proposed algorithm demonstrated considerable improvement in accuracy when compared to the state-of-the-art standalone LOF method. The remainder of this chapter is organized as follows: Section 2 provides a review of related methods; Section 3 describes the Concrete material components; Section 4 presents the methodology of the IFS-LOF method; Section 5 provides the experimental results; and Section 6 states the conclusions.

6.2 Related Work

Outlier detection is a significant research issue in machine learning and in data mining for detecting a rare object in real applications, such as in the fields of finance, industry, health, and materials science. When it comes to the construction industry, outlier detection has been rarely used in evaluating the quality of the measured or collected data. According to [141], only very few articles have discussed outlier detection methods used for measuring the source of the data.

6.3 Context and methodology

6.3.1 Concrete Material Components and Dataset

Concrete is a primary component in the construction of various projects. Concrete ingredients have recently changed a lot by introducing multiple materials and admixtures that either added before or during mixing; most of these materials are waste by-products, and they are known as Supplementary

Cementitious Materials (SCMs). Fly ash, silica fume, and blast furnace slag are the most common types of SCMs used in the concrete industry. These waste by-products enhance short-term properties, such as compressive strength, tensile strength, and workability, and they significantly improve concrete durability over time [142–146]. Moreover, concrete is a multifarious material, and its properties are significantly affected by the individual properties of its constituents. In this paper, the concrete data collection was obtained from the University of California Irvine (UCI) machine-learning repository that was released in [147] and [129]. The data collection included the results of the compressive strength of 1030 concrete mixtures. Compressive strength is one of the important parameters used by the engineering community in structural concrete design that can be seen in structures, bridges, etc. Table 6.1 shows the range of ingredients that have been used in the 1030 concrete mixtures. Finding the outliers in each component could improve the quality and reliability of the data to be processed. Our task was to calculate the efficiency of the IFS-LOF method in the identification of the outliers in the UCI concrete data. The proposed algorithm generates better performance than state-of-the-art LOF algorithms. Finding the outliers in each component can improve the quality of the data to be processed. Our task was to calculate the efficiency of the IFS-LOF method in the identification of the outliers in the UCI Concrete Data Collection. The proposed algorithm generates better performance than state-of-the-art LOF algorithms.

Table 6.1 Ranges of the Concrete Components

Component	Minimum (kg/m^3)	Maximum (kg/m^3)	Average
Cement	71	600	232.2
Fly ash	0	175	46.4
Blast furnace slag	0	359	79.2
Water	120	228	186.4
Superplasticizer	0	20.8	3.5
Coarse aggregate	730	1322	943.5
Fine aggregate	486	968	819.9

6.4 Components and workflow of the method

6.4.1 The Isolation Forest (IF)

The IF is an unsupervised method used in the Ensemble-based model to isolate the anomalies by measuring the isolation score for all data points. The IF has the same concept of using the tree model as the Random Forest algorithm. Then it processes the data point into recurrent random splits that are dependent on the selecting features [147]. The main advantage of the IF algorithm is how it processes the data. Instead of processing all the data points, it uses a decision tree to isolate the outliers, which reduces the execution or processing time and its memory requirement [84]. The IF technique operates by partitioning the model into several segments that are required for the subsampling size, as illustrated in Fig. 6.1. An anomaly score is used to create a path length for the tree to isolate the outlier, as shown in Algorithm 1. The IF calculation begins with a certain data point value. Then, according to the selected value, it sets a range between the maximum and the minimum to determine the outlier score for each data point in the tree. The score is calculated to set a path length to isolate the outlier. For more details, the reader should refer to [148].

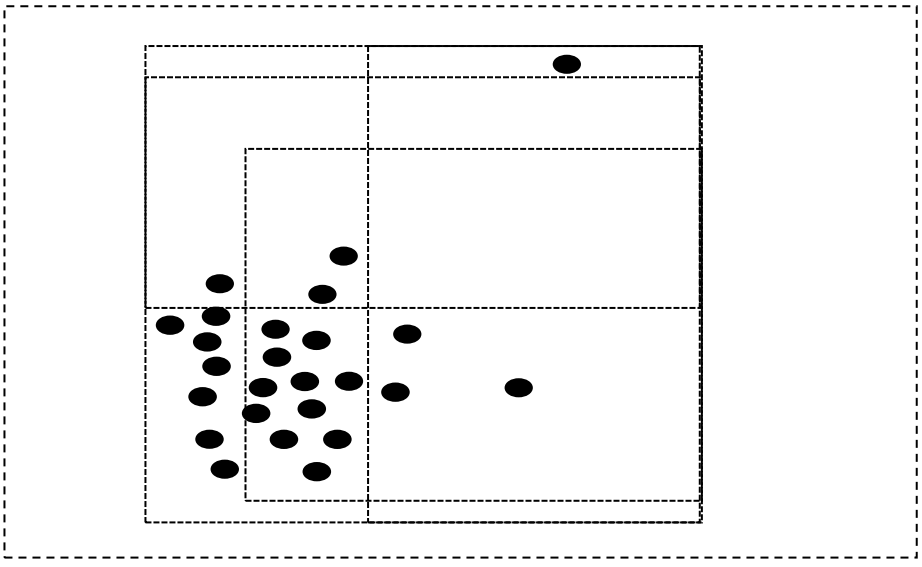


Figure 6.1 Illustration of the subsampling size in the isolation forest for processing data points.

Table 6.2: The Isolation Forest (IF) Algorithm

Algorithm 1: IForest (D, t, x)

Input: D – Input data set for the data points t – number of tree x– subsampling size s

Output: a set of t iTree

1 **Init** Forest

2 set height limit l – ceiling $log_2 x$

3 **For** i = 1 to t **do**

4 $D' \leftarrow$ sample D, x

5 Forest ← Forest ∪ iTreeD', 0, l

6 **End** for

7 **Return** Forest

6.4.2 Local Outlier Factor (LOF)

The LOF is an unsupervised approach in the density-based outlier detection to search the anomaly based on a score to determine if the data point is outlier or normal. LOF evaluates the data points according to a degree of measurement, i.e., the outlier factor regarding the density of the local neighbors. The definition of LOF is presented in figure 6.2.

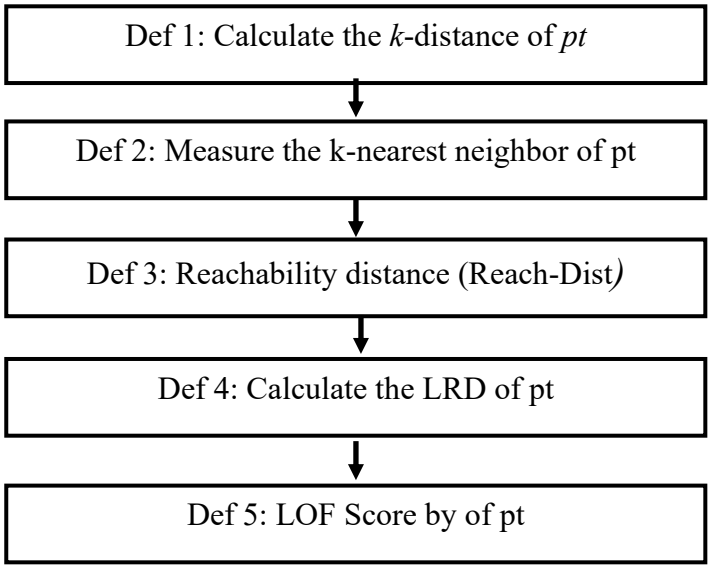


Figure 6.2 The key definitions of LOF algorithm.

6.4.3 The Isolation Forest based on Sliding window For the Local Outlier Factor (IFS-LOF)

This section describes the proposed IFS-LOF objective for finding the outlier. It proposes to increase the accuracy of detecting the outlier by using the sliding window for selecting the outlier candidates from the IF algorithm. The proposed IFS-LOF algorithm has two stages: the processing stage and detection stage, as shown in Figure 6.3 and Table 6.3

Table 6.3: The Isolation Forest-based on the Sliding window for Local Outlier Factor (IFS-LOF Algorithm)

Algorithm 2: IFS-LOF

Input: D – Input data set for the data points t – number of tree S-sliding windows k-number of nearest neighbor x -outlier candidate datapoint

Output: outlier score

1 **Init** Forest

2 x outlier candidate set → **Call Algorithm 1 with** *D, t, w, S*

3 **For j =1 to D do → Call LOF with k, x**

4 **If** the LOF for X-temp is > θ then

X-Temp is outlier

5 **End**

6 **End** for

7 **End**

Table 6.3 illustrates how the IFS-LOF algorithm process works. In the processing phase (lines 1-2), the isolation forest will process the concrete mixture data set, determine the number of tree to build, and determine the sampling size. Then, the sliding window is used as a window size (ws) to store the data points from the IF algorithm. In the detection phase, the LOF threshold θ is used to calculate the data input from the sliding windows to determine the outlier score. Any data points that exceed the threshold value are considered outliers (lines 3-5).

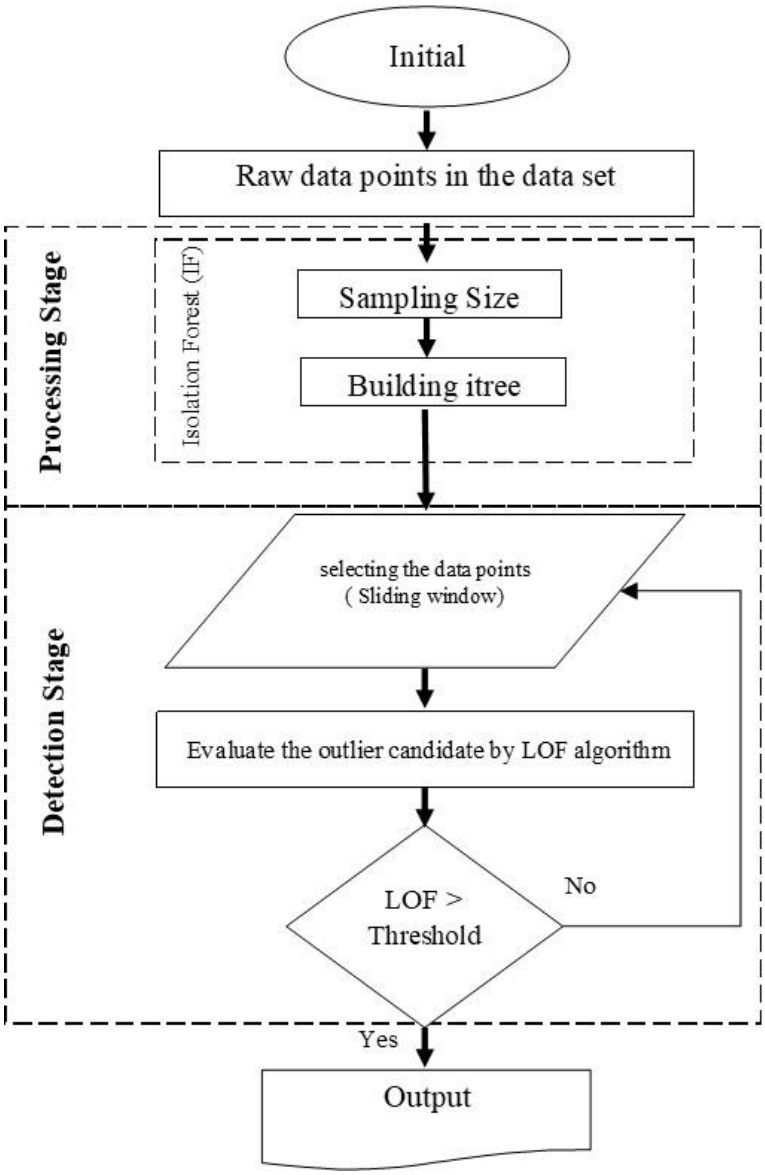


Figure 6.3 The structure of the IFS-LOF Algorithm.

6.5 Experimental Results and Discussion

This section describes the experimental results after comparing the IFS-LOF, LOF-Sliding window (LOF-SW), and LOF with different window sizes (*ws*). The purpose of IFS-LOF is to answer the following questions:

- Does IFS-LOF perform better than LOF and LOF-SW in the accuracy of outlier detection?
- Does the Sliding window improve the accuracy of outlier detection?

- Does IFS-LOF perform faster than LOF and LOF-SW in execution time?

6.5.1 Experiment Settings

All algorithms (LOF, LOF-SW, and IFS-LOF) were implemented in Java and operated on a machine that runs on operating system of Windows 10 (64-bit) with Intel Core (MT) i7-4940MX CPU, 16 GB RAM, and 1 TB SSD hard disk. The accuracy of the outlier detection for IFS-LOF, LOF-SW, and LOF was calculated by using the ROC Curve (AUC) method, as set out in [131,132]. In particular, AUC was used for the first question to obtain the accuracy rate. The sliding window strategy was adopted in the second question in order to compare the performance between the IFS-LOF and LOF-SW methods for the accuracy of outlier detection. The parameter of the KNN was set at 8 for all algorithms, including LOF, LOF-SW, and IFS-LOF. The IFS-LOF had a selecting feature that was set at 0.25 for the IF algorithm. Different sizes of windows were used to evaluate the performance of each algorithm, as is presented in Table 6.4. The window size (*ws*) has different values for the comparison between the algorithms: *ws*={100.200.300,400}.

6.5.2 Experiment Discussion

6.5.2.1 The Accuracy of Outlier Detection

The accuracy of outlier detection was assessed by applying AUC, as shown in Table 6.3 The IFS-LOF, LOF-SW, and LOF algorithms processed each element in the UCI concrete data set. Figures (6.4 to 6.11) illustrate the comparison of the accuracy rate with different window sizes (*ws*). Based on each element's result, we can illustrate the most suitable method to use for a greater accuracy rate. For the cement element, both IFS-LOF and LOF-SW had a higher accuracy rate for most window sizes compared to LOF. The reason is because of the size of the windows used to process the data in the memory. LOF performed better in the smaller sizes of the windows, while both LOF-SW and IFS-LOF had an advantage in the larger sizes of windows. The IFS-LOF method surpassed all other methods with an accuracy rate of 97.48% when it reached *ws*=400. Based on IFS-LOF's performance, it had a better accuracy rate for the superplasticizer element for all *ws*. The only weakness of IFS-LOF's performance in the concrete

elements was seen in the Blast Furnace Slag, Coarse Aggregate, and Water. LOF performed better than IFS-LOF in larger sizes of windows. In addition, in the Coarse aggregate element, the gap of the accuracy was noticeable when *ws* reached w= 100. However, when the size of the windows increased, IFS-LOF performed better than LOF when it reached *ws=200*. In the Coarse aggregate element, both IFS-LOF and LOF-SW performed low accuracy rates for the remaining *ws*. The IFS-LOF had a better accuracy rate than LOF, as illustrated in the blast furnace Slag element. The LOF-SW performance was better than LOF in the cement and Fine Aggregate elements when the ws reached *ws={300.400}*.

Table 6.4 The Accuracy rate of the LOF, LOF-SW and IFS-LOF for different windows sizes

W size / Component	100			200			300			400		
	LOF	LOF-SW	IFS-LOF	LOF	LOF-SW	IFS-LOF	LOF	LOF-SW	IFS-LOF	LOF	LOF-SW	IFS-LOF
Cement	90.89	89.01	90.44	96.68	92.10	95.40	94.49	95.38	92.66	94.67	96.55	97.48
Blast Furnace Slag	80.50	80.20	82.49	79.77	89.93	91.67	93.50	92.85	94.52	93.77	91.12	93.65
Fly Ash	67.80	72.05	67.36	93.80	85.54	93.68	94.70	92.45	94.28	93.81	93.77	94.28
Superplasticizer	77.39	78.01	90.91	84.41	80.69	93.13	88.57	86.21	88.97	82.94	89.56	89.20
Coarse Aggregate	98.46	85.58	85.83	92.08188	91.14	93.20	97.65234	95.23	96.44	97.28	94.59	96.51
Fine Aggregate	85.77	84.89	90.95	89.77501	87.21	90.40	93.09886	96.99	94.70	95.24	95.95	96.62
Age	82.83	86.46	88.56	90.03	93.10	86.46	94.0972	90.47	94.18	90.04	88.15	94.73
Water	85.13	62.63	90.53	96.23	92.78	94.69	91.35495	93.82	92.16	94.37	91.63	93.24

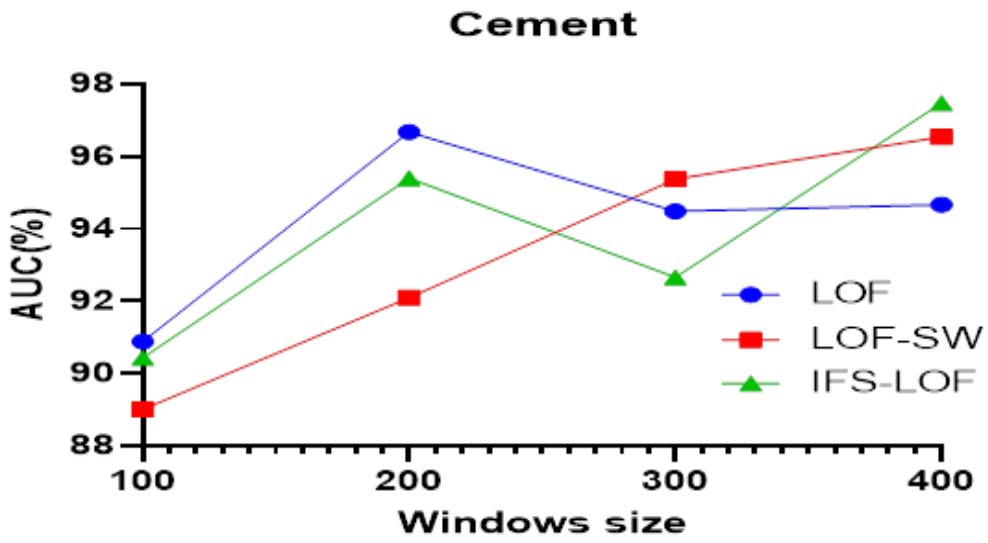


Figure 6.4 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Cement component.

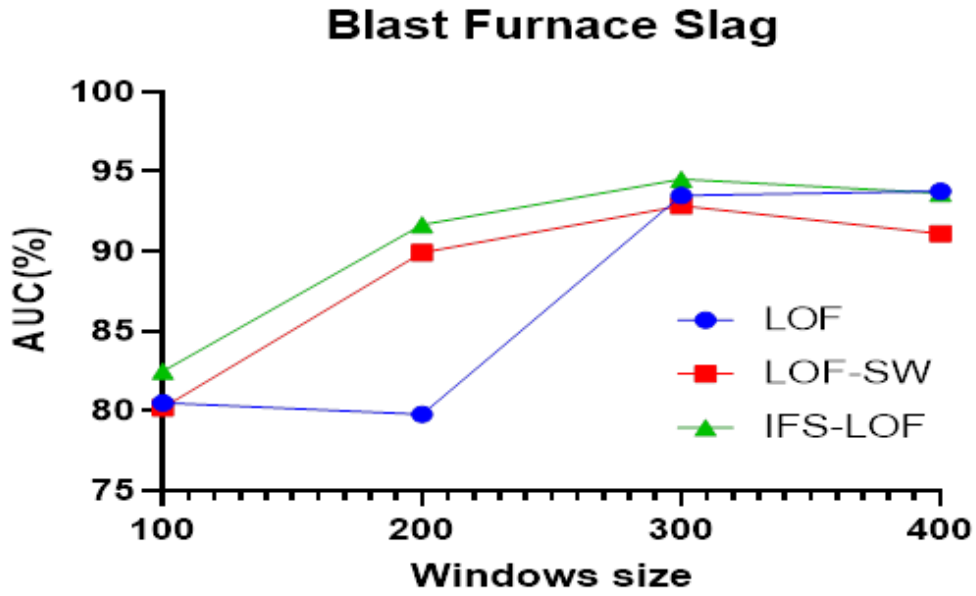


Figure 6.5 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Blast Furnace Slag component.

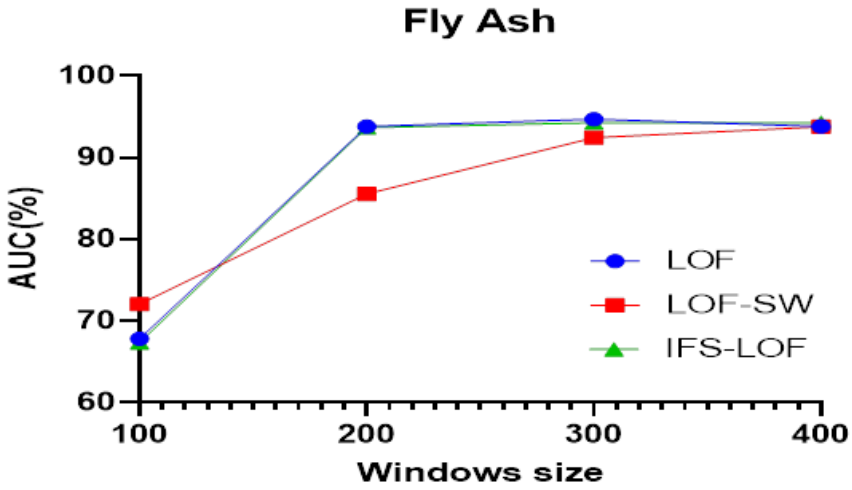


Figure 6.6 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Fly Ash component.

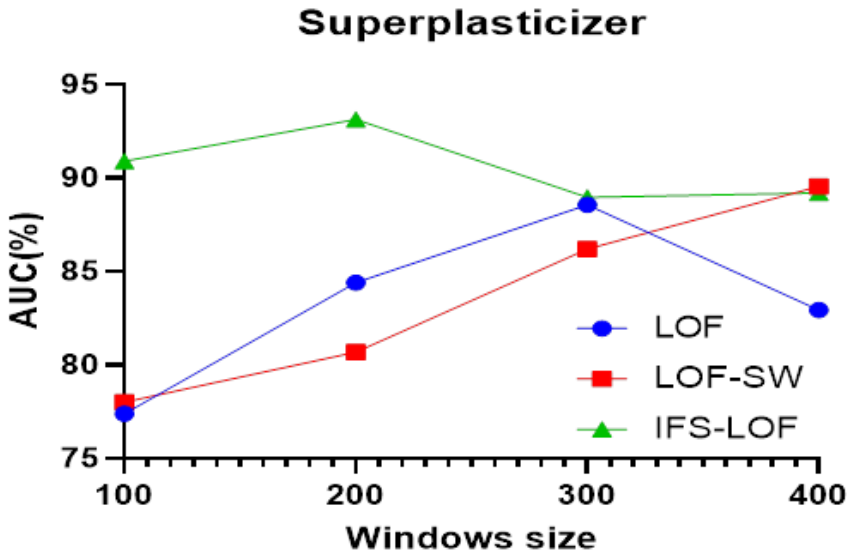


Figure 6.7 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Superplasticizer component.

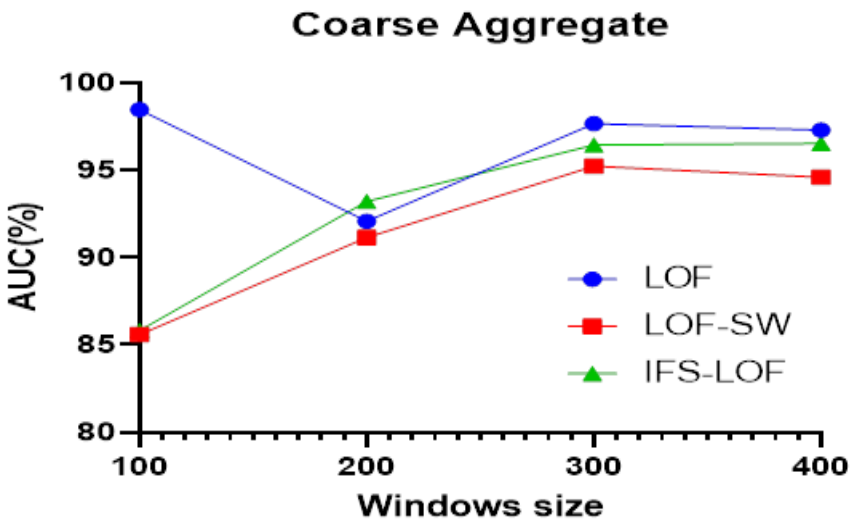


Figure 6.8 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Coarse Aggregate component.

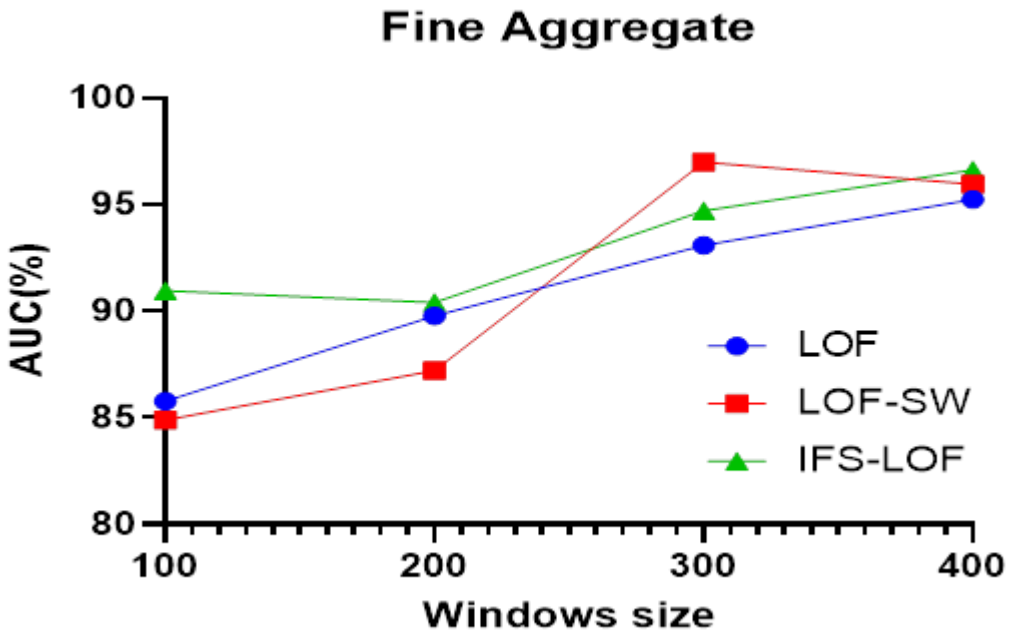


Figure 6.9 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Fine aggregate component.

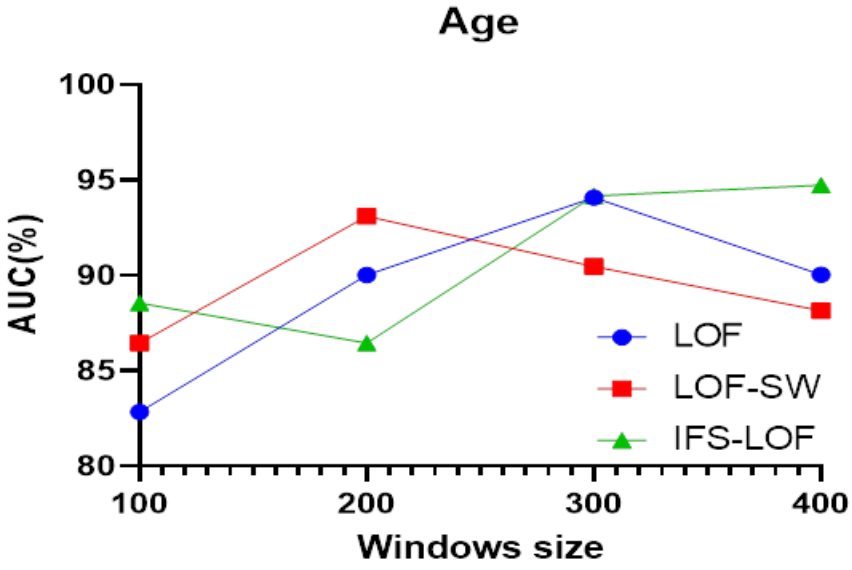


Figure 6.10 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Age component.

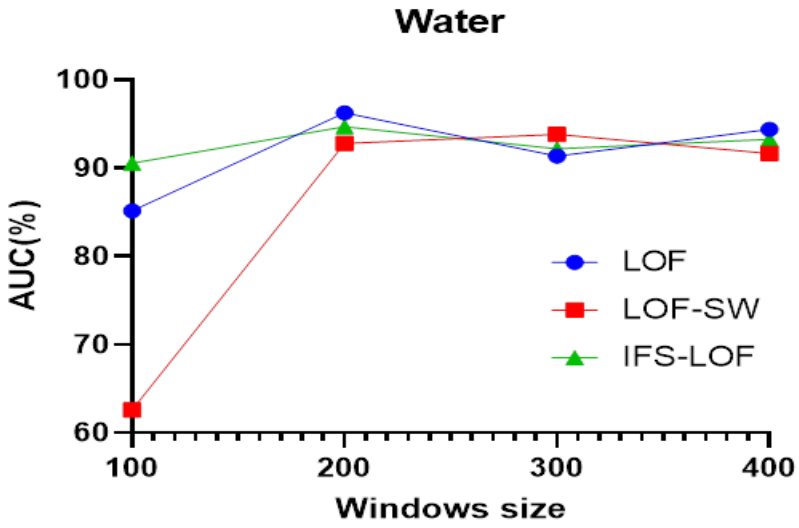


Figure 6.11 Comparison of accuracy results between LOF, LOF-SW, and IFS-LOF in the Water component.

6.5.2.2 Sliding Window Strategy for Improving the Outlier Detection

The sliding window strategy has improved the accuracy rate for most of the concrete components. Still, one of the drawbacks of the method is related to the size of the window used to process the data. For example, LOF-SW produced a lower accuracy rate, when the size of the window was increased. This is due to the amount of the data used in the sliding window, which has an impact on the results of the accuracy performance. To improve the accuracy rate in the sliding window technique, the IFS-LOF algorithm strengthens the accuracy output in the sliding window by using the IF algorithm. The IF algorithm enhances the sliding window by selecting the isolation data point instead of processing all of the data. IFS-LOF improves its usability in comparison with LOF-SW (Fig. 6.4 to 6.11). IFS-LOF presented most consistently higher accuracy in most of the *ws* compared to the other remaining algorithms.

6.5.2.3 Execution Time

Table 6.5 represents the execution time for the concrete data elements by comparing LOF, LOF-SW, and IFS-LOF. All algorithms were measured in seconds. In general, LOF was slightly better in execution time than either LOF-SW or IFS-LOF for most of the elements when it reached *ws*={100,200}. The Superplasticizer and coarse aggregate elements were executed faster than LOF-SW and IFS-LOF in all *ws*. For the remaining windows at *ws*= {300,400}, we noticed that both LOF-SW and IFS-LOF execution times were much lower than the LOF algorithm. The main reason was related to the sliding window technique. The LOF-SW algorithm was slightly better than the IFS-LOF in the execution time. However, it had a lower accuracy rate than LOF-SW.

Table 6.5 The Execution times of the LOF, LOF-SW and IFS-LOF for different windows size

Execution Time / Component	100			200			300			400		
	LOF	LOF-SW	IFS-LOF	LOF	LOF-SW	IFS-LOF	LOF	LOF-SW	IFS-LOF	LOF	LOF-SW	IFS-LOF
Cement	1.55	1.62	1.58	4.34	4.83	5.19	9.25	8.93	10.09	16.24	16.17	18.98
Blast Furnace Slag	8.05	8.02	8.24	7.97	8.99	9.16	9.35	9.28	9.45	9.37	9.11	9.36
Fly Ash	1.58	1.74	1.78	6.76	8.33	7.47	22.21	19.32	15.84	48.81	50.75	53.48
Superplasticizer	1.66	1.57	1.95	4.3	7.05	5.2	9.81	22.39	10.77	19.67	48.67	21.38
Coarse Aggregate	1.71	1.55	1.7	5.57	5.15	6.04	10.81	11.04	12.07	20.24	21.68	23.45
Fine Aggregate	1.7	1.44	2.08	4.4	4.45	5.62	9.72	9.15	10.72	17.39	17.1	20.03
Age	1.65	1.7	2.17	4.4	4.79	5.33	9.46	9.34	11.37	16.35	16.08	19.37
Water	2.38	1.8	2.19	8.88	9.28	8.14	31.77	18.29	23.2	40.36	37.95	50.91

6.5.2.4 Benefit of Using the Outlier Detection in the Concrete Mix Design

Outlier detection, which is part of data mining, aims to find a point or group of points in the dataset that deviates significantly in behavior from the rest of the data points. Different factors influence the recognition of an item as an outlier. One of the factors could be seen in the quality of the data. The main causes of the poor quality of data included defective data processing methods. The data is often generated from various heterogeneous sources; human or machine error may occur at data entry or processing. These issues may be found in practical applications. For example, the author in [149] illustrated the poor

quality of the data used because of the lack of reliability from the sensors used. Another benefit of outlier detection is that it can enhance the strength assessment of the construction process. The strength assessment is usually carried out at 7–28 days after the concrete has been poured. The quality assessment of concrete may include some unusual data. Using the IFS-LOF outlier detection method can improve the reliability of data processing during the concrete mixture design, which reduces the expenses and time.

6.6 Conclusion

In this article, the IFS-LOF algorithm was developed and compared with LOF and LOF-SW. A 1030 concrete mixtures dataset was used in the study to investigate the accuracy rate of the IFS-LOF. The concrete mixtures included various material proportions of water, cement, fine aggregate, coarse aggregate, fly ash, slag and superplasticizers. In addition, the concrete age of all mixtures was included in the analysis. The benefits and drawbacks have been analyzed in the concrete dataset to enhance the strength and workability of concrete mixtures by searching the outlier through measuring the accuracy rate and execution time. The main objective of IFS-LOF is to enhance the accuracy rate in each ingredient of the concrete data and solve the limitation of LOF. The outcome of the IFS-LOF demonstrated an improvement in the accuracy rate other than state-of-the-art LOF algorithms. Moreover, the popular LOF algorithm needs broad memory to hold all the data before identifying the local outlier.

Chapter 7: Conclusion and Future Research Direction

7.1 Introduction

Outlier detection is a method used in data mining and machine learning that indicates a data point's divergence from typical behavior in the dataset. Applied outlier detection research has evolved into a vast resource of algorithms used in network intrusion detection, fraud detection, and web analytics, to name just a few. Outliers can be categorized into two general types: global or local. This paper focuses on local outlier detection. The most popular methodology for local outlier detection is a density-based technique named the local outlier factor (LOF). There are numerous methods for detecting local outliers based on various algorithms; however, the majority of these methods were developed for a static environment and are not applicable for streaming data, which is the usual form of big data nowadays. Data streams pose a challenge for local outlier algorithms because they must cope with the high-speed stream and provide efficient analysis. The LOF is one of the most appropriate techniques used in the density-based method to determine the outlier. However, it faces some difficulties regarding data streams. First, LOF processes the data all at once, which is not suitable for data streams. Another issue occurs when a new data point arrives; significantly, all the data points need to be recalculated. Additionally, the LOF requires the whole dataset to be stored in memory. Therefore, it affects the execution time. This chapter summarizes the dissertation work and outlines interesting directions for future study.

7.2 Summary of main research goals and accomplishments

To solve the LOF constraint in data streams, new methods should be developed. As a consequence, each new method can achieve the key objective of calculating the LOF score by considering the following circumstances (as set out in [19]): (1) a portion of the dataset is stored in computer memory; (2) no previous knowledge regarding the distribution of data as outliers is detected; (3) the algorithm does not have any knowledge regarding future data points when it detects an outlier using the current dataset; and, (4) the algorithm should check an incoming data point to determine if it is either normal or an outlier pt.

We developed a new approach for finding local outliers to address the LOF's problems in the stream environment. Our methodology has three phases: preprocessing, processing, and detection. The preprocessing phase selects the first half of the data points to be processed in the next step. The data points are then divided, based on the grid method. The LOF algorithm processes each grid to obtain the outlier score. The detection phase ensures that any data points that surpass the threshold are deleted. Our method is called the grid partition-based local outlier factor (GP-LOF). The main objective of the GP-LOF is to find the outlier under the previously described conditions

The reachability distance, also known as the local outlier factor by reachability distance, is another method for calculating an outlier score (LOFR). The LOFR concept measures the outlier score without relying on the local reachability distance. Except for the lrd step, the LOFR uses, the LOF concepts and it generates a new LOFR ranking. The LOFR is taken from the data point (*pt*) reachability distance and is separated by the average of the neighbor's reachability distance, as discussed in chapter 5.In an attempt to improve the efficiency of the GP-LOF algorithm, we added the LOFR algorithm to the GP-LOF method and named it a grid-partition-based local outlier factor by reachability distance (GP-LOFR). The latest GP-LOFR calculation technique is presented in [150]. We also developed a new approach for outlier detection in for the construction industry. The concrete community requires such a framework to produce an efficient way of constructing concrete mixtures efficiently. Evaluation of measurements of samples that could involve humans or computer errors could result in outliers. Six research questions were addressed in our work, as presented in table 7.1. This section will summarize the responses to the research questions listed in Chapter 1. The research questions are repeated below to assist the reader:

Questions	Research Question	Chapters	Pages
1	How does the GP-LOF algorithm apply the LOF in processing the data stream?	4 & 5	57,71
2	How does the GP-LOF algorithm solve the memory consumption issue?	2 & 4	47,57
3	How does the GP-LOF algorithm deal with the incoming data points?	4	60
4	Does the GP-LOF algorithm perform better than the DILOF algorithm for the accuracy of outlier detection?	4 & 5	62, 74
5	Does the GP-LOF algorithm perform better than the DILOF algorithm in execution time?	4 & 5	65, 77
6	How does the new approach of the outlier detection method evaluate the concrete mixture?	6	87

In response to the first question, we solved the issue of the LOF by using the GP-LOF and GP-LOFR algorithms. For the second question, we summarized the dataset using a limited memory and the sliding window technique with a particular window size. We used two steps to resolve the third question; first, we took the first 50% of data points and keep the rest; second, we used the GP-LOF to obtain the outlier score from each grid. In addition, a threshold was used to determine if the data points were common or outliers. For the fourth and fifth questions, both the GP-LOF and the GP-LOFR showed an improved result for the accuracy of outlier detection for the majority of the dataset. However, as compared to the GP-LOF algorithm, the GP-LOFR had some drawbacks dealing with the dataset in terms of execution time. Chapters 4 and 5 address the analysis of the results of both the GP-LOF and the GP-LOFR in detail. For the last question, we developed a new approach for outlier detection in concrete mixtures by proposing a new method, called Isolation Forest (IF), based on the sliding window for the local outlier factor, which overcame both methods' limitations discussed in Chapter 6. The dissertation chapters are depicted in the diagram below (Figure 7.1).

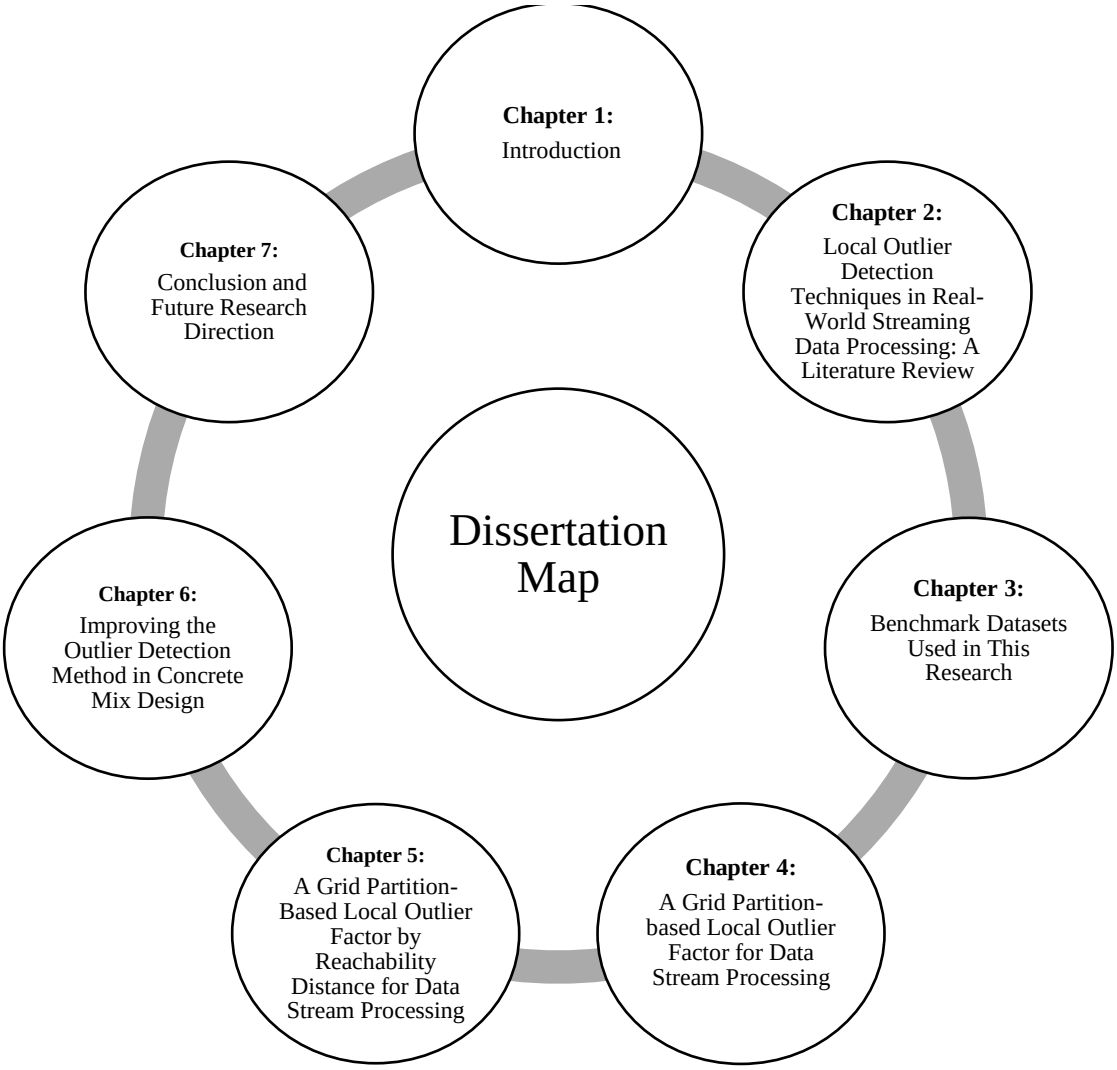


Figure 7.1 The map of the dissertation chapters.

7.3 Conclusion and Future Direction

Many applications require methods of dealing with unexpected results and identifying unusual results in the big data era. One of the goals of an outlier detection method is to discover suspicious or strange patterns. For example, extracting information from the dataset may lead to wrong conclusions because the information is inaccurate. Recently, outlier detection has received a lot of attention due to the increased complexities of data streams. This dissertation proposed a new way of identifying local outliners by developing two methods for data streams; they are called the Grid Partition-based Local

Outlier Factors (GP-LOF) and the Grid Partition-based Local Outlier Factors by the Reachability Distance (GP-LOFR). Both approaches perform well when compared with the DILOF algorithm.

For potential future work, other conventional local outlier algorithms can now be developed to function in the data stream. The above techniques, such as the GP-LOF and the GP-LOFR algorithms, can be used to implement the conventional algorithms in a data stream. The recent use of a Genetic Algorithms (GA) has demonstrated a possible solution for dealing with the problem of the local outlier in a data stream. It can also be combined with other common methods for detecting local outliers, such as COF, LOCI, aLOCI, and LoOP. Other EC methods for processing the data stream can be developed. Another future path may be to pair the LOF algorithm with another robust method to improve local outlier performance, such as the Isolation Forest that is discussed in chapter 7. This dissertation dealt with LOF problems and challenges in the stream environments and presented new methods for increasing the efficiency of local outlier detection.

7.4 Achievement and Award

7.4.1 List of Publications:

- As the main author

1) Alsini, R. and Ma, X., 2019. Data Streaming. Encyclopedia of Big Data; Schintler, L., McNeely, C., Eds

Research paper Title: Data Streaming

Types: Book chapter.

Explanation: present a general information about the Data stream. It is published online by Springer in the encyclopedia of big data.

2) 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.

Research paper Title: A Grid Partition-based Local Outlier Factor for Data Stream Processing

Types: Conference Paper

Explanation: a new method is proposed to solve the LOF algorithm's issue in the data stream. It is accepted at the 4th International Conference on Applied Cognitive Computing by Springer.

3) 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.

Research paper Title: A Grid Partition Based Local Outlier Factor by Reachability Distance for Data Stream Processing

Types: Conference Paper

Explanation: This research paper aims to improve the GP-LOF algorithm's accuracy in detecting the outlier in the data stream environment. It has been accepted at International Conference on Computational Science and Computational Intelligence (CSCI).

4) Alsini, R., Almakrab, A., Ibrahim, A. and Ma, X., Improving the outlier detection method in concrete mix design by combining the isolation forest and local outlier factor. *Construction and Building Materials*, *270*, p.121396.

Research paper Title: Improving the outlier detection method in concrete mix design by combining the isolation forest and local outlier factor.

Types: Journal

Explanation: This research paper aims to detect the outlier in the concrete mixture design. It introduces a new approach of combining the Isolation Forest (IF) and Local Outlier Factor (LOF) to solve both method issues. It is published online in the construction and building material journal under science direct.

5) Alsini, R., Alghushairy, O. Almakrab, A., Soule, T. and Ma, X., 2021. Local Outlier Detection Techniques in Real-World Streaming Data Processing: A Literature Review (Under Review).

Research paper Title: Local Outlier Detection Techniques in Real-World Streaming Data Processing: A Literature Review.

Types: Journal

Explanation: This research paper aims to overview the recent progress in the local outlier detection techniques. It also illustrates the most common application apply such as static and stream environment. Also, it addresses the issue of the current process of the local outlier detection in the stream environment.

- As co-author

6) 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).

Research paper Title: A Genetic-Based Incremental Local Outlier Factor Algorithm for Efficient Data Stream Processing.

Types: Conference Paper

Explanation: This research paper aims to solve the local outlier detection in processing the data stream. It is publish at 4th international conference in compute and data analysis (ICCDA 2020), at the ACM.

7) 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).

Research paper Title: Improving the Efficiency of Genetic based Incremental Local Outlier Factor Algorithm for Network Intrusion Detection.

Types: Conference Paper

Explanation: This research paper aims to improve the GILOF algorithm by introducing a new calculation method called Local Outlier Factor by Reachability distance (LOFR). It is accepted at the 4th International Conference on Applied Cognitive Computing by Springer.

8) 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.

Research paper Title: An Efficient Local Outlier Factor for Data Stream Processing: A Case Study.

Types: Conference Paper

Explanation: This research paper aims to address the LOF algorithm's issue in processing the data stream. It has been accepted at International Conference on Computational Science and Computational Intelligence (CSCI).

9) 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.

Research paper Title: A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams

Types: Journal

Explanation: This research article provides a review of the local outlier factor in data stream processing. It is published online at the MDPI journal.

- Awards:

1 A grant from the department of computer science to participate the CSCI 2020.

2 A grant from the college of graduate studies (GPSA) for paper publication.

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 (ICCPCCT) (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. Cios, K.J., Pedrycz, W. and Swiniarski, R.W., 2012. *Data mining methods for knowledge discovery* (Vol. 458). Springer Science & Business Media.

6. Knorr, E.M. and Ng, R.T., 1998, August. Algorithms for mining distance-based outliers in large datasets. In *VLDB* (Vol. 98, pp. 392-403).

7. Garofalakis, M., Gehrke, J. and Rastogi, R. eds., 2016. *Data stream management: processing high-speed data streams*. Springer.

8. Fan, W. and Bifet, A., 2013. Mining big data: current status, and forecast to the future. *ACM SIGKDD explorations newsletter*, *14*(2), pp.1-5.

9. Hawkins, D.M., 1980. *Identification of outliers* (Vol. 11). London: Chapman and Hall.

10. Jiang, M. F.; Tseng, S. S.; Su, C. M. Two-phase clustering process for outliers detection. Pattern Recognition Letters 2001, 22, 691–700.

11. Muthukrishnan, S., Shah, R. and Vitter, J.S., 2004, June. Mining deviants in time series data streams. In Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004. (pp. 41-50). IEEE.

12. Aggarwal, C.C. and Philip, S.Y., 2005. An effective and efficient algorithm for high-dimensional outlier detection. The VLDB journal, 14(2), pp.211-221.

13. Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), pp.1-58.

14. Wang, H., Bah, M.J. and Hammad, M., 2019. Progress in outlier detection techniques: A survey. IEEE Access, 7, pp.107964-108000.

15. Boukerche, A., Zheng, L. and Alfandi, O., 2020. Outlier Detection: Methods, Models, and Classification. ACM Computing Surveys (CSUR), 53(3), pp.1-37.

16. Saxena, S. and Rajpoot, D.S., 2019. Density-Based Approach for Outlier Detection and Removal. In Advances in Signal Processing and Communication (pp. 281-291). Springer, Singapore.

17. Alsini, R. and Ma, X., 2019. Data Streaming. Journal: Encyclopedia of Big Data, pp.1-4.

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

19. 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.

20. 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).

21. 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.

22. 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.

23. 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.

24. 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).

25. He, Z., Xu, X. and Deng, S., 2003. Discovering cluster-based local outliers. *Pattern Recognition Letters*, *24*(9-10), pp.1641-1650.

26. 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).

27. Goldstein, M. Anomaly Detection in Large Datasets. Ph.D. thesis, University of Kaiserslautern, Kaiserslautern, Germany, 2016.

28. 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.

29. Alsini, R., Almakrab, A., Ibrahim, A. and Ma, X., 2021. Improving the outlier detection method in concrete mix design by combining the isolation forest and local outlier factor. *Construction and Building Materials*, *270*, p.121396.

30. Goldstein, M. and Uchida, S., 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, *11*(4), p.e0152173.

31. 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.

32. 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.

33. Goldstein, M., 2012, November. FastLOF: An expectation-maximization based local outlier detection algorithm. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)* (pp. 2282-2285). IEEE.

34. 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.

35. Guan, H., Li, Q., Yan, Z. and Wei, W., 2015, September. SLOF: identify density-based local outliers in big data. In *2015 12th Web Information System and Application Conference (WISA)* (pp. 61-66). IEEE.

36. Liu, J. and Wang, G., 2016, May. Outlier detection based on local minima density. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference* (pp. 718-723). IEEE.

37. 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.

38. 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.

39. 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).

40. 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.

41. 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.

42. Du, H., Zhao, S., Zhang, D. and Wu, J., 2016, April. Novel clustering-based approach for local outlier detection. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 802-811). IEEE.

43. Thang, V.V., Pantiukhin, D.V. and Nazarov, A.N., 2016, November. FLDS: fast outlier detection based on local density score. In *2016 International Conference on Engineering and Telecommunication (EnT)* (pp. 137-141). IEEE.

44. 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.

45. Babaei, K., Chen, Z. and Maul, T., 2019. Detecting point outliers using prune-based outlier factor (plof). *arXiv preprint arXiv:1911.01654*.

46. 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.

47. 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.

48. Miao, D., Qin, X. and Wang, W., 2015. Anomalous cell detection with kernel density-based local outlier factor. *China Communications*, *12*(9), pp.64-75.

49. Du, H., 2015, November. Robust local outlier detection. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)* (pp. 116-123). IEEE.

50. Tang, B. and He, H., 2017. A local density-based approach for outlier detection. *Neurocomputing*, *241*, pp.171-180.

51. Wang, L. and Deng, X., 2017, July. Multimode process fault detection method based on variable local outlier factor. In *2017 9th International Conference on Modelling, Identification and Control (ICMIC)* (pp. 175-180). IEEE.

52. 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.

53. Wang, R.; Zhu, Q. LSOF: Novel Outlier Detection Approach Based on Local Structure. 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) 2019.

54. 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.

55. 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.

56. 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).

57. Gao, K., Shao, F.J. and Sun, R.C., 2010, July. n-INCLOF: A dynamic local outlier detection algorithm for data streams. In *2010 2nd International Conference on Signal Processing Systems* (Vol. 2, pp. V2-179). IEEE.

58. Pokrajac, D., Reljin, N., Pejcic, N. and Lazarevic, A., 2008, September. Incremental connectivity-based outlier factor algorithm. In *Visions of Computer Science-BCS International Academic Conference* (pp. 211-223).

59. 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.

60. Wang, Z., Zhao, Z., Weng, S. and Zhang, C., 2015. Incremental multiple instance outlier detection. *Neural Computing and Applications*, *26*(4), pp.957-968.

61. Kalliantzis, I., Papadopoulos, A., Gounaris, A. and Tsichlas, K., 2019. *Efficient Distributed Outlier Detection in Data Streams* (Doctoral dissertation, Aristotle University of Thessaloniki; 54124 Thessaloniki; Greece).

62. Liu, F., Yu, Y., Song, P., Fan, Y. and Tong, X., 2020. Scalable KDE-based top-n local outlier detection over large-scale data streams. *Knowledge-Based Systems*, *204*, p.106186.

63. Yang, Y., Chen, L. and Fan, C., 2021. ELOF: fast and memory-efficient anomaly detection algorithm in data streams. *Soft Computing*, *25*(6), pp.4283-4294.

64. 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.

65. 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.

66. Gao, J., Ji, W., Zhang, L., Li, A., Wang, Y. and Zhang, Z., 2020. Cube-based incremental outlier detection for streaming computing. Information Sciences, 517, pp.361-376.

67. 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.

68. 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.

69. 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).

70. 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.

71. 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.

72. 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).

73. 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.

74. 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).

75. 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.

76. 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).

77. Huang, J.W., Zhong, M.X. and Jaysawal, B.P., 2020. TADILOF: Time Aware Density-Based Incremental Local Outlier Detection in Data Streams. *Sensors*, *20*(20), p.5829.

78. 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.

79. Agrawal, S. and Agrawal, J., 2015. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, *60*, pp.708-713.

80. Ahmed, M., Mahmood, A.N. and Islam, M.R., 2016. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, *55*, pp.278-288.

81. Markou, M. and Singh, S., 2003. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, *83*(12), pp.2481-2497.

82. Markou, M. and Singh, S., 2003. Novelty detection: a review—part 2:: neural network based approaches. *Signal processing*, *83*(12), pp.2499-2521.

83. Hodge, V. and Austin, J., 2004. A survey of outlier detection methodologies. *Artificial intelligence review*, *22*(2), pp.85-126.

84. 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.

85. Chen, L., Gao, S. and Cao, X., 2020. Research on real-time outlier detection over big data streams. *International Journal of Computers and Applications*, *42*(1), pp.93-101.

86. 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.

87. 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*.

88. Pimentel, M.A., Clifton, D.A., Clifton, L. and Tarassenko, L., 2014. A review of novelty detection. *Signal Processing*, *99*, pp.215-249.

89. Park, C.H., 2019. Outlier and anomaly pattern detection on data streams. *The Journal of Supercomputing*, *75*(9), pp.6118-6128

90. 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.

91. 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.

92. Singh, A., Thakur, N. and Sharma, A., 2016, March. A review of supervised machine learning algorithms. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 1310-1315). Ieee.

93. Lodhia, Z., Rasool, A. and Hajela, G., 2017. A survey on machine learning and outlier detection techniques. *IJCSNS*, *17*(5), p.271.

94. Quinlan, R., 1993. 4.5: Programs for machine learning morgan kaufmann publishers inc. *San Francisco, USA*.

95. Mehrotra, K., Mohan, C.K. and Ranka, S., 1997. *Elements of artificial neural networks*. MIT press.

96. Gao, J., Cheng, H. and Tan, P.N., 2006, April. Semi-supervised outlier detection. In *Proceedings of the 2006 ACM symposium on Applied computing* (pp. 635-636).

97. Daneshpazhouh, A. and Sami, A., 2013, May. Semi-supervised outlier detection with only positive and unlabeled data based on fuzzy clustering. In *The 5th Conference on Information and Knowledge Technology* (pp. 344-348). IEEE.

98. Ruff, L., Vandermeulen, R.A., Görnitz, N., Binder, A., Müller, E., Müller, K.R. and Kloft, M., 2019. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*.

99. Fitriani, S., Mandala, S. and Murti, M.A., 2016, November. Review of semi-supervised method for intrusion detection system. In *2016 Asia Pacific Conference on Multimedia and Broadcasting (APMediaCast)* (pp. 36-41). IEEE.

100. Heller, K., Svore, K., Keromytis, A.D. and Stolfo, S., 2003. One class support vector machines for detecting anomalous windows registry accesses.

101. Zoppi, T., Ceccarelli, A. and Bondavalli, A., 2018, December. On algorithms selection for unsupervised anomaly detection. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 279-288). IEEE.

102. Ngai, E.W., Hu, Y., Wong, Y.H., Chen, Y. and Sun, X., 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, *50*(3), pp.559-569.

103. Zhao, M. and Chen, J., 2020, June. A Review of Methods for Detecting Point Anomalies on Numerical Dataset. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (Vol. 1, pp. 559-565). IEEE.

104. 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.

105. Jain, A.K. and Dubes, R.C., 1988. *Algorithms for clustering data*. Prentice-Hall, Inc..

106. 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.

107. Han, J.; Pei, J.; Kamber, M.Data Mining: Concepts and Techniques; Elsevier: Amsterdam, The Netherlands, 2011.

108. Krempl, 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.

109. Ikonomovska, E., Loskovska, S. and Gjorgjevik, D., 2007, September. A survey of stream data mining. In *Proceedings of 8th National Conference with International participation, ETAI* (pp. 19-21).

110. Younas, M., 2019. Research challenges of big data.

111. 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.

112. Namiot, D., 2015. On big data stream processing. *International Journal of Open Information Technologies*, *3*(8).

113. O'Reilly, C., Gluhak, A., Imran, M.A. and Rajasegarar, S., 2014. Anomaly detection in wireless sensor networks in a non-stationary environment. *IEEE Communications Surveys & Tutorials*, *16*(3), pp.1413-1432.

114. Aggarwal, C.C., 2015. Outlier analysis. In *Data mining* (pp. 237-263). Springer, Cham.

115. 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).

116. 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).

117. Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsichlas, 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.

118. Bai, M., Wang, X., Xin, J. and Wang, G., 2016. An efficient algorithm for distributed density-based outlier detection on big data. *Neurocomputing*, *181*, pp.19-28.

119. Aggarwal, C.C., Philip, S.Y., Han, J. and Wang, J., 2003, January. A framework for clustering evolving data streams. In Proceedings 2003 VLDB conference (pp. 81-92). Morgan Kaufmann.

120. 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.

121. Guha, S., Meyerson, A., Mishra, N., Motwani, R. and O'Callaghan, L., 2003. Clustering data streams: Theory and practice. IEEE transactions on knowledge and data engineering, 15(3), pp.515-528.

122. Aggarwal, C.C., Han, J., Wang, J. and Yu, P.S., 2004, August. A framework for projected clustering of high dimensional data streams. In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30 (pp. 852-863).

123. Aggarwal, C.C., 2012. A segment-based framework for modeling and mining data streams. Knowledge and information systems, 30(1), pp.1-29.

124. Assent, I., Kranen, P., Baldauf, C. and Seidl, T., 2012, April. Anyout: Anytime outlier detection on streaming data. In International Conference on Database Systems for Advanced Applications (pp. 228-242). Springer, Berlin, Heidelberg.

125. Salehi, M., Leckie, C.A., Moshtaghi, M. and Vaithianathan, T., 2014, May. A relevance weighted ensemble model for anomaly detection in switching data streams. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 461-473). Springer, Cham.

126. Cheng, Z., Zou, C. and Dong, J., 2019, September. Outlier detection using isolation forest and local outlier factor. In Proceedings of the conference on research in adaptive and convergent systems (pp. 161-168).

127. Staerman, G., Mozharovskyi, P., Clémençon, S. and d'Alché-Buc, F., 2019, October. Functional isolation forest. In Asian Conference on Machine Learning (pp. 332-347). PMLR.

128. Ding, Z. and Fei, M., 2013. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proceedings Volumes, 46(20), pp.12-17.

129. 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.

130. Shebuti Rayana (2016). ODDS Library [http://odds.cs.stonybrook.edu]. Stony Brook, NY: Stony Brook University, Department of Computer Science.

131. 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.

132. 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.

133. Aggarwal, C.C. and Sathe, S., 2015. Theoretical foundations and algorithms for outlier ensembles. Acm sigkdd explorations newsletter, 17(1), pp.24-47.

134. 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.

135. 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*.

136. 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).

137. 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).

138. 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).

139. 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.

140. Zhang, M., Li, M., Shen, Y., Ren, Q. and Zhang, J., 2019. Multiple mechanical properties prediction of hydraulic concrete in the form of combined damming by experimental data mining. *Construction and Building Materials*, *207*, pp.661-671.

141. Yan, H., Yang, N., Peng, Y. and Ren, Y., 2020. Data mining in the construction industry: Present status, opportunities, and future trends. *Automation in Construction*, *119*, p.103331.

142. Hassan, E.C. and Ibrahim, A., 2013. The performance of high-strength flowable concrete made with binary, ternary, or quaternary binder in hot climate. *Construction and Building Materials*, *47*, pp.245-253.

143. Mahmoud, E., Ibrahim, A., El-Chabib, H. and Patibandla, V.C., 2013. Self-consolidating concrete incorporating high volume of fly ash, slag, and recycled asphalt pavement. *International Journal of Concrete Structures and Materials*, *7*(2), pp.155-163.

144. Ibrahim, A., Mahmoud, E., Khodair, Y. and Patibandla, V.C., 2014. Fresh, mechanical, and durability characteristics of self-consolidating concrete incorporating recycled asphalt pavements. *Journal of materials in civil engineering*, *26*(4), pp.668-675.

145. Ibrahim, A., El-Chabib, H. and Eisa, A., 2013. Ultrastrength flowable concrete made with high volumes of supplementary cementitious materials. *Journal of materials in civil engineering*, *25*(12), pp.1830-1839.

146. Ibrahim A., and Mahmoud E., and Ali, T. (2013). Macroscopic Compressive Strength of High-Strength Self-Consolidating Concrete with High Volume of Cementitious Materials Based on Real Digital Image

147. Yeh, I.C., 1998. Modeling of strength of high-performance concrete using artificial neural networks. Cement and Concrete research, 28(12), pp.1797-1808.

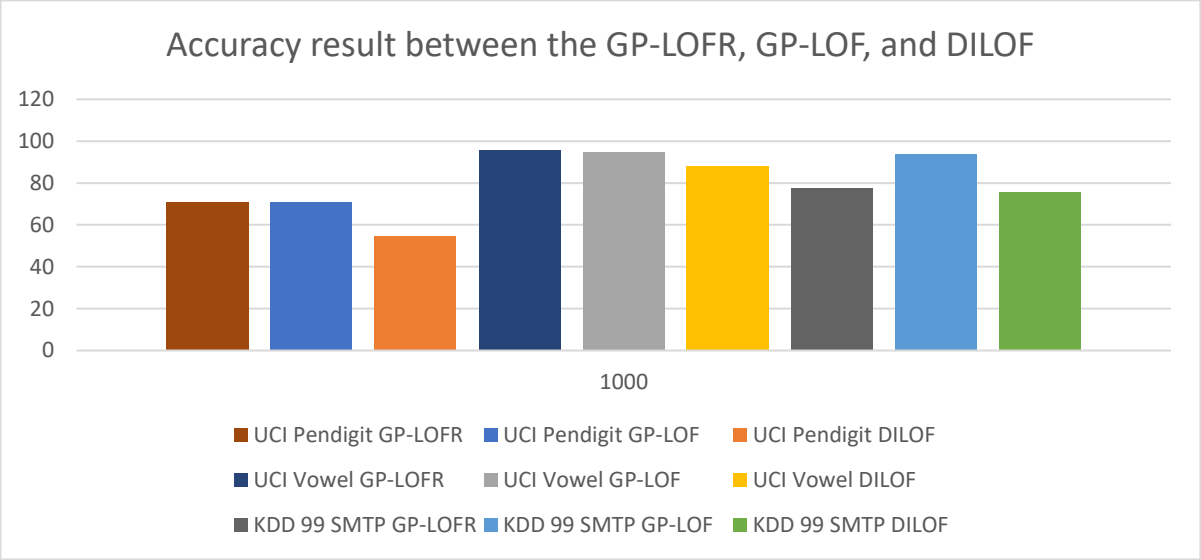
148. Liu, F.T., Ting, K.M. and Zhou, Z.H., 2008, December. Isolation forest. In 2008 eighth ieee international conference on data mining (pp. 413-422). IEEE.

149. Yu, Z.J., Haghighat, F. and Fung, B.C., 2016. Advances and challenges in building engineering and data mining applications for energy-efficient communities. Sustainable Cities and Society, 25, pp.33-38.

150. 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.

Appendix A - Other Experiment tests between the GP-LOF, GP-LOFR, DILOF, IFS-LOF

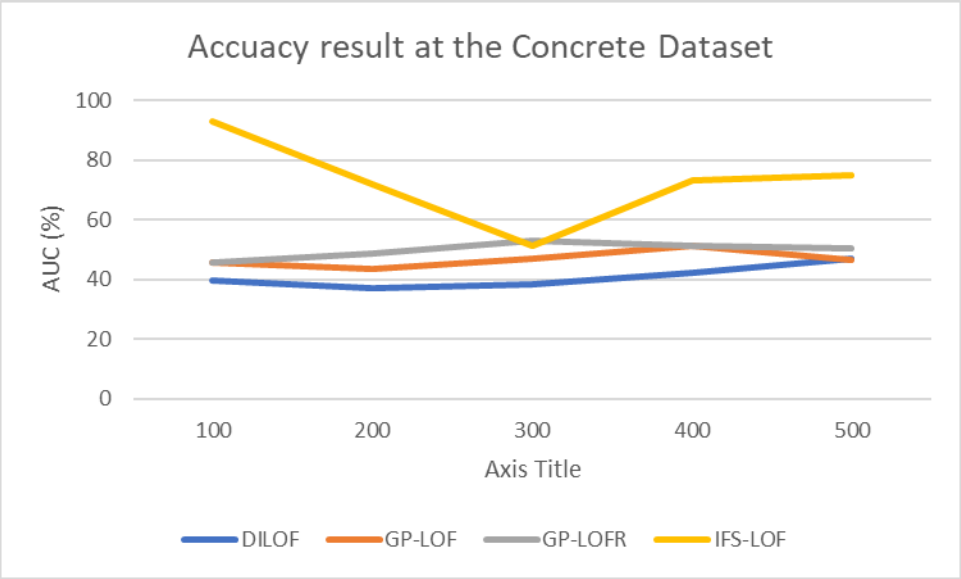
- The tables and figure below show the performance results of the algorithms between the GP-LOF, DILOF algorithms in the window size 1000.

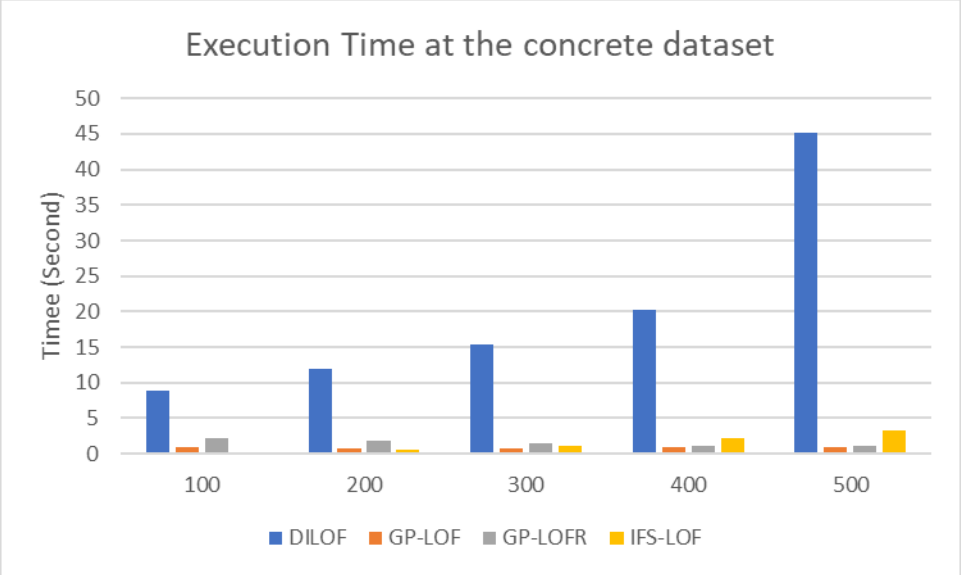


Window size	UCI Pendigit			UCI VOWEL			KDD 99 SMPT		
	GP-LOF	GP-LOFR	DILOF	GP-LOF	GP-LOFR	DILOF	GP-LOF	GP-LOFR	DILOF
1000	70.91	71.01	54.93	94.6384	95.54	88.19	94.01	77.43	75.82

Window size	UCI Vowel			KDD99 SMPT			UCI Pendigit		
	GP-LOF	Dilof	GP-LOFR	GP-LOF	Dilof	GP-LOFR	GP-LOF	Dilof	GP-LOFR
1000	1.815	10.3391	1.925	130.11	1155.23	219.347	5.49373	4.645	5.575

- The tables and figure below show the performance results of the algorithms in the concrete dataset between the GP-LOF, DILOF and IFS-LOF algorithms in all window sizes.





- **Accuracy result:**

window size	DILOF	GP-LOF	GP-LOFR	IFS-LOF
100	39.8082	45.80337	45.6479	92.84
200	37.3153	43.65678	48.62021	71.8625
300	38.5663	47.06702	53.05968	51.33333
400	42.2612	51.36927	51.51733	73.2
500	47.1796	46.42113	50.24005	74.8296

- **Execution time:**

window size	DILOF	GP-LOF	GP-LOFR	IFS-LOF
100	8.7898	0.9	2.215	0.172
200	11.9397	0.787	1.837	0.512
300	15.42	0.747	1.391	1.034
400	20.1695	0.885	1.162	2.101
500	45.1401	0.993	1.156	3.215