Stream Data Analysis and Processing Frameworks for Detecting Outliers in Human Activities: A Review

MOHAMMED SABHA, BULENT TUGRUL*
Department of Computer Engineering,
Ankara University, Golbasi, Ankara,
TURKEY

**Corresponding Author*

Abstract: - Every day, we generate enormous amounts of data from a wide range of personal devices. The rapid increase in data amount and velocity is pushing our limits to process and analyze them. Traditional machine learning and data analytics methods and algorithms use all historical data in the dataset to build their analyses and models. This may lead to processing and analyzing large amounts of historical data being computationally expensive and time-consuming, especially in real-time applications where speed is crucial. Furthermore, using all historical data may not account for changes in the models and dynamics underlying the data over time. This could lead to inaccurate forecasts or insights. Streaming analytics, on the other hand, processes each point of continuous data as it is received. It is more efficient than batch processing in certain cases. Real-time data processing using stream analytics allows organizations to make immediate and proactive decisions based on up-to-date information. This can be especially beneficial in time-sensitive industries, such as finance or logistics, where even a slight delay in data analysis can result in missed opportunities or costly errors. Additionally, stream analytics enables businesses to detect and respond to anomalies in real time, leading to enhanced operational efficiency and customer experiences. Statistically significant outliers are instances that don't follow the general trend of the data. Datasets may contain outliers for several reasons, such as mistakes made during data collection or the presence of extremely high or low values. Because of the potential impact of outliers on analysis, it is worthwhile to carefully consider whether or not they should be included. This is useful for spotting inconsistencies or discrepancies, as well as determining which parts of the data need more in-depth analysis. This study discusses topics related to stream data analysis. These topics include a variety of frameworks for processing and analyzing streaming data, methods for detecting outliers, and human activity detection.

Key-Words: - big data, human activity recognition, outlier detection, stream data.

Received: April 15, 2024. Revised: November 11, 2024. Accepted: December 13, 2024. Published: January 10, 2025.

1 Introduction

Remarkable developments all over the world have led to rapid technological growth. Many exciting concepts have emerged, such as the Internet of Things (IoT), one of the technologies behind many big data solutions, [1]. This has created new demands, such as high efficiency and real-time data processing. The challenges of processing and analyzing big data have led researchers to develop advanced algorithms and technologies that can handle large-scale and heterogeneous data in real time, [2]. The challenges of processing and analyzing big data include the overwhelming volume of data but also the variety and velocity at which it is generated. Traditional data processing techniques and tools are often inadequate to handle such massive and diverse datasets in real time. As a result, researchers have been working on developing parallel processing algorithms and technologies that can effectively extract insights and value from big data while maintaining high efficiency and accuracy, [3].

There are times when anomalous samples are more critical than normal ones, [4]. Detecting fraud in credit card transactions, machine fault diagnosis, and network intrusion detection are the most common examples. There are a number of factors that pose challenges for analysts to deal with. First, outlier detection requires storing infinitely massive amounts of data streams before processing, which results in high memory usage. Second, the curse of dimensionality results in data sets having a high number of dimensions, making the calculation of distance between points less effective and meaningless, [5]. Thirdly, because there are no labeled data in unsupervised learning, the model

must take into account hidden patterns from the available data. Finally, most anomaly detection algorithms have difficulty detecting anomalies in unbalanced data. This is due to the fact that they are based on an awareness of the behavior of more common, normal samples.

Stream data processing, however, presents challenges in terms of feature engineering and delivery to the machine learning model, which affects the model's overall performance. The complexity of the data itself presents a challenge for processing stream data. The nature of data makes it difficult to comprehend since it is constantly changing and moving. The consequences of this are significant for feeding data into a machine-learning

model, since each data point becomes less useful with time, [6]. It is essential to process each data point before the next one arrives to prevent delays as illustrated in Appendix in Figure 1 and Figure 2.

Considering the speed and volume of data, low throughput, and high latency challenges affect computation quality and data value. Low throughput limits the amount of data the pipeline can receive, while high latency restricts its ability to process it. Further, when data messages are streamed, they may encounter various scenarios. For example, messages may be lost or sent more than once, with clients having to read them again. To properly address these scenarios, organizations must have a comprehensive data latency and throughput strategy, including proper data replication and caching techniques.

A real time streaming pipeline for machine-learning models is challenging to set up. First, the lack of fault tolerance makes data delivery more likely to fail, which could cause data loss and stop the whole streaming pipeline process. Also, there is a lack of scalability, which means that the streaming pipeline will have to be redesigned every time there are more data producers or consumers. There is a chance that this could pose a burden and result in pipeline improvements being halted. The most critical problem with not being able to scale is not being able to decouple. This affects how long the pipeline lasts and how data flows between producers and consumers.

In this paper, we will discuss a few interconnected areas: frameworks for processing and analyzing streaming data, methods for identifying anomalies, and human activity recognition using only a smartphone. Figure 3 (Appendix) provides an overview of the system implementation. This study is broken down into three sections. The first section of the paper focuses on the infrastructure and software for processing streaming data. In the second section, we'll look at how various outlier detection algorithms deal with huge, multidimensional, and potentially infinite data streams. The final section of the study covers research into recognizing human activity using sensors embedded in mobile phones.

2 Big Data Stream Processing and Analysis

There has been a huge increase in open-source streaming framework availability recently. As a result, deciding on the right framework for streaming data use cases might be confusing. Stream processing techniques offer distinct advantages and disadvantages. For example, window-based processing allows for efficient time-based aggregation but does not support the handling of out-of-order events. On the other hand, event-oriented processing offers flexibility but may require more complex logic for time-dependent aggregation. It is, therefore, a necessity to decide which method will be used depending on the specific requirements of the particular use case before choosing a stream processing technique.

A majority of studies evaluated Apache Spark Streaming, Flink, and Storm, three of the most well-known stream processing engines. [7], developed a benchmarking application at Yahoo to simulate a real-world stream processing use case. Their study compares, evaluates, and analyzes respective engines. Additionally, [8] conducted an in-depth study to assess common stream processing frameworks. Each framework is analyzed in terms of performance, scalability, and resource utilization.

Alternatively, [9] compares Spark and Storm frameworks to determine how they differ in latency. He was interested in how well these frameworks performed when executing a variety of tasks, such as "Word Count". Due to the aggregation required for processing small logs, Storm can handle data much faster than Spark Streaming. Storm's performance decreases linearly with increasing record sizes. In contrast, Spark streaming performed very well in this scenario. It was six times faster at processing 1,000-byte records than Storm.

A further discussion of Apache Flink is found in [7]. They used streaming windows to ingest data from Kafka, count and aggregate it for live analysis of advertising campaigns. Flow motor performance was evaluated for 30 minutes. Storm and Flink displayed similar latency characteristics while Spark was faster due to its micro-batching design. Spark Streaming provides an advantage over other frameworks due

to its ability to change batch sizes on the fly. Additionally, Storm failed to handle the data volume when it reached 135,000 events per second, whereas Spark did not experience any problems.

Similarly, [10] discussed novel ways to meet the growing demand for non-batch processing needs and Hadoop's incapacity to handle them. They concluded that Spark is a more efficient option for in-memory computing among the frameworks examined. It is also a suitable alternative for handling real-time and streaming demands. Apache Spark was built to address MapReduce's constraints, [11]. Spark [12] and Spark streaming [13] are key tools for building large data applications. Several studies have evaluated Spark's performance across different workloads and benchmarks, [14].

Spark streaming was introduced in [15] as a dependable and high-throughput API for live data streams. Spark focuses on making a single pass over the data utilizing data stream clustering.

Further research into Spark's potential to challenge MapReduce and Flink has been published. These works emphasize key points such as effective memory management, [16], [17], [18].

According to research such as [19], no single framework can handle all data kinds, sizes, and business models. The study found that Spark outperforms Flink in big graph processing by about 1.7 times. In contrast, Flink beats Spark by up to 1.5 times in scenarios such as batch and small graph workloads. As a result, it uses fewer resources and is easier to set up. Other studies, such as [20] and [21], revealed that Apache Spark's Stream outperformed Storm in terms of resource usage and peak throughput. This was verified by [7] that Storm struggled at high throughput. Storm, on the otherhand, performed better in latency.

Standard data mining techniques and machine learning models are challenging to apply to vast amounts of data at varying speeds, [22]. To meet this challenge, Apache Spark not only facilitates high-end data analytics and efficient general processing tasks but also allows machine learning algorithms. Spark has been a point of interest for various types of research where they use it as a large data stream processing engine to stream data in real-time in various fields. The results show that their system achieved exemplary performance and robustness, [23], [24], [25].

[26], proposed a study that divided the streaming pipeline into two stages (Reading and Processing) to analyze the impact of each stage on the end-to-end delay. The proposed study uses interconnects, a decoupling layer, to optimize the streaming pipeline by implementing a message queueing mechanism during the reading stage. Apache Kafka can considerably increase the overall performance of the streaming pipeline system. Furthermore, Apache Kafka is the most appropriate choice for employing Spark streaming as a data input phase. This is because of its dependability and compliance with Apache Spark streaming. Furthermore, it creates a separate queue for message delivery and guarantees correct sequencing.

Apache Kafka was utilized in many stream processing applications as a message broker system. For instance, [27] implemented a study to model the design characteristics of distributed-based frameworks such as Spark. Apache Spark processes a tuple faster than Storm using an average flat incremental clustering algorithm. This is because it utilizes Kafka as a message broker, ensuring proper order and delivery of messages.

Another study utilized Apache Kafka as a robust messaging broker system, [28]. The researchers built a real-time Twitter streaming and data analytics system. The framework is divided into three primary parts. Kafka is used as a data intake phase to extract data from Twitter and provide quick access to Kafka producers. The system analyzes and processes tweets in real time as possible. In contrast, Spark streaming was used as a stream processing engine throughout the processing step. At the end of the process, the data was visualized. Experiments have been conducted to evaluate Kafka's performance with traditional message brokers, notably known as scalable and efficient implementations of Advanced Message Queuing Protocol (AMQP), such as RabbitMQ, [29]. According to the study's findings, both systems can process messages with low latency. The Kafka protocol is better suited to applications that handle huge volumes of messages. Increasing Apache Kafka partitions can greatly boost throughput and producer/channel count. Besides, it increases RabbitMQ's performance.

[30], conducted a comparison of five message queueing techniques. The authors reported their findings using a replicable experimental setup and a defined comparison metric. The authors concluded the experimental analysis by demonstrating that RocketMQ has a shorter latency. Kafka has a higher throughput. According to [31], integrating Spark and Kafka can significantly reduce the time consumed and increase performance within streaming data analysis and processing applications. However, evaluating and recommending stream processing engines for streaming data with different use cases remains an ongoing research field. They are considering the massive demand for large data stream mining and processing platforms that integrate and use distributed

stream processing in a single open source. In addition, they want the ability to simultaneously apply machine learning techniques to the stream. Flink and Spark are cutting-edge distributed stream processing engines that provide these capabilities. However, more work is needed to integrate and extend analytical libraries within the same processing platform. For example, with all these huge advantages of Spark and this enormous ability to process data in real time, many machine learning algorithms still need to be supported on the platform within the structured streaming service.

3 Outlier Detection

In some contexts, outliers in data are more appealing than expected ones, so it is vital to recognize and identify them since they may contain critical information. There is currently a lot of interest in identifying outliers among researchers. By understanding and addressing these outliers, researchers can improve the accuracy and reliability of their findings, leading to more robust and meaningful conclusions.

Various studies provide a detailed analysis of state-of-the-art strategies for discovering outliers, dividing them into distinct approaches and categories, [32]. They were divided into statistical, distance, density, clustering, and ensemble methods. Furthermore, the authors investigated their primary strengths and limitations.

[33] highlighted the significance of detecting outliers, especially in time series data. The authors organized the topic around distinct data types and provided multiple outlier definitions. They also briefly defined the associated processes and reviewed a variety of applications where these strategies have been successfully used. The statistics-based technique [34] creates conventional distribution models by analyzing historical data. It looks for data points that deviate from the distribution of other data points. Later, these data points would be labeled anomalies. Most models, however, are based on a single variable. As a result, detecting anomalies becomes difficult when monitoring parameters are multi-dimensional. Furthermore, the original data, which contains some noisy data that greatly affects the distribution model building, is used to generate these models, [35].

The distance-based technique [36] computes the distance between two data sets. Two points are deemed “neighbors” when their distance is less than a threshold value. Suppose a group of data points are less than a threshold value. In that situation, they will be classified as abnormal. However, this strategy is inappropriate for instances where the data distribution is multi-cluster, [37]. When an anomaly occurs, numerous continuous abnormal resource metric data arise and are grouped together as neighbors. This approach, however, needs help to identify them.

Similarly, other studies like [38] employ windowing concepts and clustering algorithms to identify temporal data anomalies. Furthermore, to identify outliers in multivariate data, [39] applied an alternative model of One-Class SVM. It is, however, impossible to consider the method to be a practical approach for large-scale data due to two critical factors, space complexity and time complexity.

IoT sensors create large amounts of data that must be processed. Due to the streaming nature of this data, it is becoming increasingly important to identify trends. [40], proposed an approach for dealing with such data. The high computing cost of this technique reveals opportunities to optimize the process.

The majority of the previously outlined techniques have two key limitations. They demand a lot of computations during training or are intended to detect/recognize typical system characteristics. This may lead to the conclusion that discovering outliers in these systems is just a result of mismatched classification procedures. When such factors are examined, more than the possible benefits of current approaches may be necessary to overcome their limits and application risks. This is because their detection abilities are the consequence of an algorithm devised and tailored for a purpose other than detecting anomalies. These techniques could have been developed for various uses, such as clustering and classification challenges.

In light of the scenarios described above, an algorithm not using the density or distance function is needed to advance anomaly detection significantly. In addition to offering reduced computation costs, this technique should also be able to handle big and multi-dimensional datasets and be less complex in terms of time. The available offline anomaly detection algorithms and techniques, such as density- and distance-based, as well as statistical approaches, may be inefficient at finding outliers due to memory constraints that require a single pass through the whole data. They are computationally demanding and may encounter overfitting because they require numerous passes through the data. They must also evaluate the complete dataset to find anomalies, [41]. As a result, a cutting-edge method was developed that does not assume prior knowledge of the system’s underlying dynamics. It assumes that anomalous cases are few and distinct from the rest of the data. As a result, they are more susceptible to isolation

procedures. This limits the number of such instances and brings them closer to the decision tree's root. In contrast, regular data items are more likely to be buried significantly deeper in the decision tree.

[42] suggested a fundamentally revised anomaly detection strategy prioritizing anomalous isolation above regular instance identification. Isolation forests (IF) are a highly successful solution to spot abnormalities. Tree separates anomalies closer to the tree's root than normal points using the few different characters of anomalies. IF have a low constant time complexity and minimal memory consumption, making them appropriate for large data sets. IF usually outperforms distance-based techniques with near-linear time complexity. Furthermore, it detects anomalies efficiently since it converges rapidly with a small ensemble size.

The IF technique has earned a reputation due to its extraordinary effectiveness in a wide range of high-dimensional, complex pattern identification challenges. [43] conducted a comparative research on outlier detection methods. Three different time series models are tested on 14 synthetic and real data sets. According to the paper, IF is better suited than other methods since it is an effective approach to efficiently detecting outliers while demonstrating exceptional scalability. It is memory efficient and can handle datasets of up to one million samples. [44] adopted IF for unmanned aerial vehicles. It was used in unsupervised learning. Using the Aero-Propulsion System Simulation dataset, the authors demonstrated the IF's suitability for various engineering applications. It outperforms all other alternative unsupervised distance-based algorithms considering their capacity to handle massive datasets. In addition, it results in lower linear time and total computing costs. [45] conducted an actual industry case study with IF. This was linked to one of the critical processes in semiconductor manufacturing known as etching. IF has been compared with univariate chart-based and multi-dimensional angle-oriented approaches. The results revealed that IF performed with higher accuracy than other multi-dimensional techniques.

[46] proposed a technique to detect credit card fraud with IF. They calculated various performance metrics such as accuracy, F1-score, ROC-score, and false positive rate of various approaches. During the experiment, IF proved to be very effective in detecting abnormalities in credit card transactions.

[47] used Spark streaming as a distributed computing platform to create a streaming application for time series data. Similarly, IF was tested on additional high-dimensional data in the author's implementation, [48]. The experiment was conducted using actual data acquired from eight separate CFM56-7B aero engines. A comparison of IF to other unsupervised anomaly detection algorithms revealed its scalability to high-dimensional data.

Another study [49] provided an anomaly detection solution based on an IF for streaming data, particularly time-series data. The study used a sliding window approach to analyze four real-world datasets from the UCI repository. In streaming data applications, the proposed approach successfully locates outliers with great scalability. Several published studies, [50], [51], [52], [53] have compared IF to other distance-based outlier detection methods in terms of performance measures such as precision, accuracy, and AUC. Performance findings reveal that the isolation forest-based approach outperforms other algorithms. Figure 4 (Appendix) summarizes the most common anomaly detection methods.

4 Human Activity Recognition

Health care and smart home applications can benefit from solutions that recognize human activity. Various sensors have been used to identify human activity, including wearable LED lights, cameras, and cell phones, [54]. Several Human Activity Recognition (HAR) systems integrate accelerometers to distinguish daily activities, including standing, walking, sitting, jogging, and lying. [55], searched for repeated behaviors using accelerometer data created from recordings of 30 participants doing routine activities daily. They tried to identify and prevent older adults from falling into a smart environment. Figure 5 (Appendix) depicts the components of a typical HAR system.

Human activity recognition tasks using wearable sensors have been examined in earlier studies. They indicated that wearable sensors could remarkably enhance human activity recognition accuracy, [56]. Some previous research has been applied to enhance recent wearable sensor-based HAR investigations. [57] developed a customized human activity recognition system. Using a multi-modal sensing device, they collected data on a variety of activities from a group of 28 volunteers ranging in gender, age, weight, and height. The data was then utilized to create a variety of hybrid activity models.

[58] utilized a wearable device to capture acceleration data based on a data set consisting of 20 computationally efficient features for human activity recognition. They achieved a 94% accuracy. In addition, [59] used three wearable accelerometers to gather data from 10 patients to track lower body

movements. A previous study, [60], presented eight activities; running, walking, standing, sit-ups, vacuuming, brushing teeth, stairs-down, and stairs-up. Those activities were detected with only one triaxial accelerometer worn near the pelvis.

Numerous studies have been conducted on the ability to use cellphones instead of wearable sensors to perform human activity identification tasks due to their high computing capabilities. Smartphones support several sensors, including gyroscopes and accelerometers. Moreover, its wireless connectivity capabilities make it valuable for recognizing and detecting human activity. Mobile devices incorporating built-in sensors have become a natural part of everyday life. Consequently, they can be considered a promising platform for HAR applications, [61]. [62] presented an online user-independent human activity classification approach using statistical features that include global and valuable properties of the time series. [63] combined a smartwatch and a smartphone sensor to identify 13 daily human activities.

Research shows that human activities can be accurately detected with a single accelerometer, based on earlier studies, [64]. A single accelerometer worn on the right waist achieved the highest recognition accuracy. [65] developed a sliding window approach to recognize physical activity for signal segmentation. A unified method is not available for identifying activities from sensor signals. Activity recognition can be achieved using a variety of algorithmic strategies. We want to examine and consider the processing power and time available based on the number of activities. Thus, depending on the approach used, the input to the HAR system's learning algorithms may alter.

Sensor signals are usually analyzed first to extract features before being sent to a classifier. These features are known as "handcrafted features" and represent raw signal data. The derived features from the original signal provide an appropriate description of the user's activity. Then, machine learning algorithms are applied to them. The "handcrafted features" technique is recommended in many applications since it improves classification accuracy. [66] analyzed many statistical parameters, including standard deviation, binned distribution, average, and duration between peaks. Furthermore, several classifiers were built based on these collected features to determine which behavior corresponds to which features. [67] combined classifiers used by

[66] using similar features and ensemble techniques to improve their findings.

Some studies extract features directly from the time-varying acceleration signal, [68]. The authors developed an independent human position activity identification system by extracting time-domain elements from raw data. [60] presented a study on identifying time-domain attributes. The authors then selected certain features, including standard deviation, mean, and energy. To classify the accelerometer signals, a classification algorithm was used, with the selected features as input. [58] suggested a method to identify human physical activities using a time-domain-based feature extraction methodology. The most acceptable features were investigated, including the mean value of min and max sums and root mean squared. With random forest as a classifier, activities were classified more accurately.

Several other researchers have worked on obtaining frequency-domain features from frequency analysis, [69], [70]. Moreover, [71] introduced an ensemble empirical mode decomposition (EEMD) based features extraction method to classify triaxial accelerometer signals for activity recognition. The Fast Fourier Transform (FFT) and Discrete Fourier Transform (DCT) coefficients are also used in various research studies, [64]. These methods can be considered frequency-based features. In addition, other features that have been utilized in experiments and yielded successful results include Principal Component Analysis (PCA) [72] and Haar filters [73].

An overview of human activity recognition research is shown in Figure 6 (Appendix). There are several factors to consider when planning to implement a project related to this area. A general description of these criteria includes types of recognition, approaches used, algorithms applied, data sources, and application areas.

5 Conclusion

Stream data is constantly generated by thousands of sources and sent simultaneously to the ingestion system in small sizes. However, processing and analyzing stream data poses several challenges due to its high velocity and volume. Real-time processing is required to handle the continuous flow of data, and algorithms need to be efficient enough to detect anomalies in real time without causing delays or bottlenecks in the system. Additionally, since stream data is constantly evolving, maintaining anomaly detection models' accuracy and relevance becomes a continuous and dynamic task. As a result, effective anomaly detection systems are required for fraud and cyberattack detection systems that need quick responses. For effective anomaly detection, it is critical to apply

big data analysis techniques like map-reduce. In contrast to previous review articles, we concentrated on big data tools and frameworks for processing and analyzing streaming data to recognize human activities. Three different architectures can be used for applications dealing with static, stream, or time-series data. Nevertheless, since not all data can be stored in memory, stream data analysis might pose particular difficulties when finding outliers. Outlier detection methods are appealing in stream data because they allow anomalies to be discovered in real time. This is especially useful for applications that need to react quickly to changes in data or find things that don't make sense automatically. Furthermore, the methods can be applied to large datasets and scaled up. Therefore, they are ideal for big data applications. They can also detect outliers even when the data is noisy or partially complete. Lastly, they can reveal patterns and trends in data that might otherwise remain hidden. Stream data can be analyzed and visualized with these methods, as well as interpreted based on their results.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used Wordtune for language editing. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References:

[1] Juarizo, Charles G and Rogelio, Jayson P and Balbin, Jessie R and Dadios, Elmer P, "IoT-based end-to-end monitoring of logistics and tracking of truck vehicles using Arduino microcontroller", *In Proceedings of The World Congress on Engineering*, pp. 3–5, 2019.

[2] Zhang, Huajie and Song, Lei and Zhang, Sen, "Parallel Clustering Optimization Algorithm Based on MapReduce in Big Data Mining", *IAENG International Journal of Applied Mathematics*, vol. 53, no. 1, 2023.

[3] Gazder, Uneb, "Studying Patterns of Rainfall and Topographical Clustering for Kingdom of Bahrain: An Application of Big Data", *Engineering World*, vol. 6, pp. 29–34, 2024, doi: 10.37394/232025.2024.6.5.

[4] Sathishkumar, E. N. and Thangavel, K., "A Novel Approach for Outlier Detection Using Rough Entropy", *WSEAS Transactions on Computers*, vol. 14, pp. 296–306, 2015.

[5] Aggarwal, Charu C. and Hinneburg, Alexander and Keim, Daniel A., "On the surprising behavior of distance metrics in high dimensional space", *In Database Theory—ICDT*, pp. 420–434, 2001, doi: 10.1007/3-540-44503-X_27.

[6] Zaharia, Matei and Das, Tathagata and Li, Haoyuan and Hunter, Timothy and Shenker, Scott and Stoica, Ion, "Discretized streams: Fault-tolerant streaming computation at scale", *In Proceedings of the Twenty-fourth ACM Symposium on Operating Systems Principles*, pp. 423–438, 2013, doi: 10.1145/2517349.2522737.

[7] Chintapalli, Sanket and Dagit, Derek and Evans, Bobby and Farivar, Reza and Graves, Thomas and Holderbaugh, Mark and Liu, Zhuo and Nusbaum, Kyle and Patil, Kishorkumar and Peng, Boyang Jerry and others, "Benchmarking streaming computation engines: Storm, flink and spark streaming", *In IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1789–1792, 2016, doi: 10.1109/ipdpsw.2016.138.

[8] Wissem Inoubli and Sabeur Aridhi and Haithem Mezni and Alexander Jung, "Big Data Frameworks: A Comparative Study", *CoRR*, vol. abs/1610.09962, 2016, doi: 10.48550/arXiv.1610.09962.

[9] Córdova, Patricio, "Analysis of real time stream processing systems considering latency", University of Toronto patricio@ cs. toronto. edu, 2015, [Online]. https://www.datascienceassn.org/sites/default/files/Analysis%20of%20Real%20Time%20Stream%20Processing%20Systems%20Considering%20Latency.pdf (Accessed Date: October 4, 2024).

[10] Shahrivari, Saeed, "Beyond batch processing: towards real-time and streaming big data", *Computers*, vol. 3, no. 4, pp. 117–129, 2014, doi: 10.3390/COMPUTERS3040117.

[11] Quinto, Butch, "Next-Generation Machine Learning with Spark: Covers XGBoost, LightGBM, Spark NLP, Distributed Deep Learning with Keras, and More", 2020, Apress, doi: 10.1007/978-1-4842-5669-5_1.

[12] APACHE Spark, "Spark Research", [Online]. https://spark.apache.org/research.html (Accessed Date: October 10, 2024).

[13] Anonymous, "Spark Structured Streaming", [Online]. https://spark.apache.org/streaming/

(Accessed Date: October 10, 2024).

[14] Xiao, Wen and Hu, Juan, "SWEclat: a frequent itemset mining algorithm over streaming data using Spark Streaming", *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7619–7634, 2020, doi: 10.1007/s11227-020-03190-5.

[15] Ghesmoune, Mohammed and Lebbah, Mustapha and Azzag, Hanene, "Micro-batching growing neural gas for clustering data streams using spark streaming", *Procedia Computer Science*, vol. 53, pp. 158–166, 2015, doi: 10.1016/j.procs.2015.07.290.

[16] Awan, Ahsan Javed and Brorsson, Mats and Vlassov, Vladimir and Ayguade, Eduard, "How data volume affects spark based data analytics on a scale-up server", *In Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, pp. 81–92, 2015, doi: 10.1007/978-3-319-29006-5_7.

[17] Veiga, Jorge and Expósito, Roberto R and Pardo, Xoán C and Taboada, Guillermo L and Tourifio, Juan, "Performance evaluation of big data frameworks for large-scale data analytics", *In IEEE International Conference on Big Data*, pp. 424–431, 2016, doi: 10.1109/BigData.2016.7840633.

[18] Shi, Juwei and Qiu, Yunjie and Minhas, Umar Farooq and Jiao, Limei and Wang, Chen and Reinwald, Berthold and Özcan, Fatma, "Clash of the titans: Mapreduce vs. spark for large scale data analytics", *Proceedings of the VLDB Endowment*, vol. 8, no. 13, pp. 2110–2121, 2015, doi: 10.14778/2831360.2831365.

[19] Marcu, Ovidiu-Cristian and Costan, Alexandru and Antoniu, Gabriel and Pérez-Hernández, María S, "Spark versus flink: Understanding performance in big data analytics frameworks", *In IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 433–442, 2016, doi: 10.1109/CLUSTER.2016.22.

[20] Lu, Ruirui and Wu, Gang and Xie, Bin and Hu, Jingtong, "Stream bench: Towards benchmarking modern distributed stream computing frameworks", *In IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 69–78, 2014, doi: 10.1109/UCC.2014.15.

[21] Qian, Shilei and Wu, Gang and Huang, Jie and Das, Tathagata, "Benchmarking modern distributed streaming platforms", *In IEEE International Conference on Industrial Technology (ICIT)*, pp. 592–598, 2016, doi: 10.1109/ICIT.2016.7474816.

[22] Attigeri, Girija and Pai MM, Manohara and Pai, Radhika M, "Supervised Models for Loan Fraud Analysis using Big Data Approach", *Engineering Letters*, vol. 29, no. 4, 2021.

[23] Nair, Lekha R and Shetty, Sujala D and Shetty, Siddhanth D, "Applying spark based machine learning model on streaming big data for health status prediction", *Computers & Electrical Engineering*, vol. 65, pp. 393–399, 2018, doi: 10.1016/j.compeleceng.2017.03.009.

[24] Ed-daoudy, Abderrahmane and Maalmi, Khalil, "Application of machine learning model on streaming health data event in real-time to predict health status using spark", *In International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, pp. 1–4, 2018, doi: 10.1109/ISAECT.2018.8618860.

[25] Alnafessah, Ahmad and Casale, Giuliano, "Artificial neural networks based techniques for anomaly detection in Apache Spark", *Cluster Computing*, vol. 23, no. 2, pp. 1345–1360, 2020, doi: 10.1007/s10586-019-02998-y.

[26] Javed, M Haseeb and Lu, Xiaoyi and Panda, Dhabaleswar K, "Characterization of big data stream processing pipeline: a case study using Flink and Kafka", *In Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, pp. 1–10, 2017, doi: 10.1145/3148055.3148068.

[27] Solaimani, Mohiuddin and Iftekhar, Mohammed and Khan, Latifur and Thuraisingham, Bhavani and Ingram, Joey Burton, "Spark-based anomaly detection over multi-source VMware performance data in real-time", *In IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pp. 1–8, 2014, doi: 10.1109/CICYBS.2014.7013369.

[28] Yadranjiaghdam, Babak and Yasrobi, Seyedfaraz and Tabrizi, Nasseh, "Developing a real-time data analytics framework for twitter streaming data", *In IEEE International Congress on Big Data (BigData Congress)*, pp. 329–336, 2017, doi: 10.1109/BigDataCongress.2017.49.

[29] Dobbelaere, Philippe and Esmaili, Kyumars Sheykh, "Kafka versus RabbitMQ: A

comparative study of two industry reference publish/subscribe implementations: Industry Paper", *In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, pp. 227–238, 2017, doi: 10.1145/3093742.3093908.

[30] Fu, Guo and Zhang, Yanfeng and Yu, Ge, "A fair comparison of message queuing systems", *IEEE Access*, vol. 9, pp. 421–432, 2020, doi: 10.1109/ACCESS.2020.3046503.

[31] Tun, May Thet and Nyaung, Dim En and Phyu, Myat Pwint, "Performance evaluation of intrusion detection streaming transactions using apache kafka and spark streaming", *In International Conference on Advanced Information Technologies (ICAIT)*, pp. 25–30, 2019, doi: 10.1109/AITC.2019.8920960.

[32] Wang, Hongzhi and Bah, Mohamed Jaward and Hammad, Mohamed, "Progress in outlier detection techniques: A survey", *IEEE Access*, vol. 7, pp. 107964–108000, 2019, doi: 10.1109/ACCESS.2019.2932769.

[33] Gupta, Manish and Gao, Jing and Aggarwal, Charu C and Han, Jiawei, "Outlier detection for temporal data: A survey", *IEEE Transactions on Knowledge and data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2013, doi: 10.1109/TKDE.2013.184.

[34] Chandola, Varun and Banerjee, Arindam and Kumar, Vipin, "Anomaly detection: A survey", *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009, doi: 10.1145/1541880.1541882.

[35] Khoa, Nguyen Lu Dang and Chawla, Sanjay, "Robust outlier detection using commute time and eigenspace embedding", *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 422–434, 2010, doi: 10.1007/978-3-642-13672-6_41.

[36] Ramaswamy, Sridhar and Rastogi, Rajeev and Shim, Kyuseok, "Efficient algorithms for mining outliers from large data sets", *In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 427–438, 2000, doi: 10.1145/342009.335437.

[37] Angiulli, Fabrizio and Basta, Stefano and Pizzuti, Clara, "Distance-based detection and prediction of outliers", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, pp. 145–160, 2005, doi: 10.1109/TKDE.2006.29.

[38] Khaleghi, Azedeh and Ryabko, Daniil and Mari, Jeremie and Preux, Philippe, "Consistent algorithms for clustering time series", *Journal of Machine Learning Research*, vol. 17, no. 3, pp. 1–32, 2016.

[39] Khan, Muhammad Aamir and Khan, Aunsia and Khan, Muhammad Nasir and Anwar, Sajid, "A novel learning method to classify data streams in the internet of things", *In National Software Engineering Conference*, pp. 61–66, 2014, doi: 10.1109/NSEC.2014.6998242.

[40] Giannoni, Federico and Mancini, Marco and Marinelli, Federico, "Anomaly detection models for IoT time series data", *arXiv preprint arXiv:1812.00890*, 2018, doi: 10.48550/arXiv.1812.00890.

[41] Aggarwal, Charu C, "An introduction to outlier analysis", *In Outlier Analysis*, pp. 1–34, 2017, Springer, doi: 10.1007/978-3-319-47578-3_1.

[42] Liu, Fei Tony and Ting, Kai Ming and Zhou, Zhi-Hua, "Isolation forest", *In Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008, doi: 10.1109/ICDM.2008.17.

[43] Domingues, Rémi and Filippone, Maurizio and Michiardi, Pietro and Zouaoui, Jihane, "A comparative evaluation of outlier detection algorithms: Experiments and analyses", *Pattern Recognition*, vol. 74, pp. 406–421, 2018, doi: 10.1016/j.patcog.2017.09.037.

[44] Khan, Samir and Liew, Chun Fui and Yairi, Takehisa and McWilliam, Richard, "Unsupervised anomaly detection in unmanned aerial vehicles", *Applied Soft Computing*, vol. 83, pp. 105650, 2019, doi: 10.1016/j.asoc.2019.105650.

[45] Susto, Gian Antonio and Beghi, Alessandro and McLoone, Seán, "Anomaly detection through on-line isolation forest: An application to plasma etching", *In 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pp. 89–94, 2017, doi: 10.1109/ASMC.2017.7969205.

[46] Ounacer, Soumaya and El Bour, Hicham Ait and Oubrahim, Younes and Ghoumari, Mohamed Yassine and Azzouazi, Mohamed, "Using Isolation Forest in anomaly detection: the case of credit card transactions", *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 6, no. 2, pp. 394–400, 2018.

[47] Weng, Yu and Liu, Lei, "A collective anomaly detection approach for

multidimensional streams in mobile service security", *IEEE Access*, vol. 7, pp. 49157–49168, 2019, doi: 10.1109/ACCESS.2019.2909750.

[48] Zhong, Shisheng and Fu, Song and Lin, Lin and Fu, Xuyun and Cui, Zhiquan and Wang, Rui, "A novel unsupervised anomaly detection for gas turbine using isolation forest", *In IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–6, 2019, doi: 10.1109/ICPHM.2019.8819409.

[49] Ding, Zhiguo and Fei, Minrui, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window", *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013, doi: 10.3182/20130902-3-CN-3020.00044.

[50] Jain, Prarthi and Jain, Seemandhar and Zaïane, Osmar R and Srivastava, Abhishek, "Anomaly detection in resource constrained environments with streaming data", *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 3, pp. 649–659, 2021, doi: 10.1109/TETCI.2021.3070660.

[51] Fan, Linchang and Ma, Jinqiang and Tian, Junjing and Li, Tonghan and Wang, Hao, "Comparative Study of Isolation Forest and LOF algorithm in anomaly detection of data mining", *In International Conference on Big Data, Artificial Intelligence and Risk Management (ICBAR)*, pp. 1–5, 2021, doi: 10.1109/ICSSIT53264.2022.9716541.

[52] Zadafiya, Narendra and Karasariya, Jenish and Kanani, Parthkumar and Nayak, Amit, "Detecting Credit Card Frauds Using Isolation Forest And Local Outlier Factor-Analytical Insights", *In 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1588–1594, 2022.

[53] Vamsi, P Raghu and Chahuan, Anjali, "Machine learning based hybrid model for fault detection in wireless sensors data", *EAI Endorsed Transactions on Scalable Information Systems*, vol. 7, no. 24, pp. e6–e6, 2020, doi: 10.4108/eai.13-7-2018.161368.

[54] Shahrim, Khairunnisa Ahmad and Abd Rahman, Abdul Hadi and Goudarzi, Shidrokh, "Hazardous Human Activity Recognition in Hospital Environment Using Deep Learning", *IAENG International Journal of Applied Mathematics*, vol. 52, no. 3, 2022.

[55] Anguita, Davide and Ghio, Alessandro and Oneto, Luca and Parra Perez, Xavier and Reyes Ortiz, Jorge Luis, "A public domain dataset for human activity recognition using smartphones", *In Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 437–442, 2013.

[56] Gyllensten, Illapha Cuba and Bonomi, Alberto G, "Identifying types of physical activity with a single accelerometer: evaluating laboratory-trained algorithms in daily life", *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2656–2663, 2011, doi: 10.1109/TBME.2011.2160723.

[57] Hong, Jin-Hyuk and Ramos, Julian and Dey, Anind K, "Toward personalized activity recognition systems with a semipopulation approach", *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 101–112, 2015, doi: 10.1109/THMS.2015.2489688.

[58] Casale, Pierluigi and Pujol, Oriol and Radeva, Petia, "Human activity recognition from accelerometer data using a wearable device", *In Iberian Conference on Pattern Recognition and Image Analysis*, pp. 289–296, 2011, doi: 10.1007/978-3-642-21257-4_36.

[59] Krishnan, Narayanan C and Colbry, Dirk and Juillard, Colin and Panchanathan, Sethuraman, "Real time human activity recognition using tri-axial accelerometers", *In Sensors, Signals and Information Processig Workshop*, vol. 2008, pp. 3337–3340, 2008.

[60] Ravi, Nishkam and Dandekar, Nikhil and Mysore, Preetham and Littman, Michael L, "Activity recognition from accelerometer data", *In Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence*, pp. 1541–1546, 2005.

[61] bin Abdullah, Mohd Fikri Azli and Negara, Ali Fahmi Perwira and Sayeed, Md Shohel and Choi, Deok-Jai and Muthu, Kalaiarasi Sonai, "Classification algorithms in human activity recognition using smartphones", *International Journal of Biomedical and Biological Engineering*, vol. 6, no. 8, pp. 362–369, 2012.

[62] Ignatov, Andrey, "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks", *Applied Soft Computing*, vol. 62, pp. 915–922, 2018, doi: 10.1016/j.asoc.2017.09.027.

[63] Shoaib, Muhammad and Bosch, Stephan

and Scholten, Hans and Havinga, Paul JM and Incel, Ozlem Durmaz, "Towards detection of bad habits by fusing smartphone and smartwatch sensors", *In IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 591–596, 2015, doi: 10.1109/PERCOMW.2015.7134104.

[64] Xue, Yang and Jin, Lianwen, "A naturalistic 3D acceleration-based activity dataset & benchmark evaluations", *In IEEE International Conference on Systems, Man and Cybernetics*, pp. 4081–4085, 2010, doi: 10.1109/ICSMC.2010.5641790.

[65] Noor, Mohd Halim Mohd and Salcic, Zoran and Kevin, I and Wang, Kai, "Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer", *Pervasive and Mobile Computing*, vol. 38, pp. 41–59, 2017, doi: 10.1016/j.pmcj.2016.09.009.

[66] Kwapisz, Jennifer R and Weiss, Gary M and Moore, Samuel A, "Activity recognition using cell phone accelerometers", *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011, doi: 10.1145/1964897.1964918.

[67] Catal, Cagatay and Tufekci, Selin and Pirmit, Elif and Kocabag, Guner, "On the use of ensemble of classifiers for accelerometer-based activity recognition", *Applied Soft Computing*, vol. 37, pp. 1018–1022, 2015, doi: 10.1016/j.asoc.2015.01.025.

[68] Torres-Huitzil, Cesar and Nuno-Maganda, Marco, "Robust smartphone-based human activity recognition using a tri-axial accelerometer", *In IEEE 6th Latin American Symposium on Circuits & Systems (Lascas)*, pp. 1–4, 2015, doi: 10.1109/LASCAS.2015.7250435.

[69] Wang, Changhai and Zhang, JianZhong and Wang, Zhicheng and Wang, Jian, "Position-independent activity recognition model for smartphone based on frequency domain algorithm", *In Proceedings of 3rd International Conference on Computer Science and Network Technology*, pp. 396–399, 2013, doi: 10.1109/ICCSNT.2013.6967138.

[70] Sun, Lin and Zhang, Daqing and Li, Bin and Guo, Bin and Li, Shijian, "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations", *In International Conference on Ubiquitous Intelligence and Computing*, pp. 548–562, 2010, doi: 0.1007/978-3-642-16355-5_42.

[71] Wang, Zhelong and Wu, Donghui and Chen, Jianming and Ghoneim, Ahmed and Hossain, Mohammad Anwar, "A triaxial accelerometer-based human activity recognition via EEMD-based features and game-theory-based feature selection", *IEEE Sensors Journal*, vol. 16, no. 9, pp. 3198–3207, 2016, doi: 10.1109/JSEN.2016.2519679.

[72] Fu, Zhongzheng and He, Xinrun and Wang, Enkai and Huo, Jun and Huang, Jian and Wu, Dongrui, "Personalized human activity recognition based on integrated wearable sensor and transfer learning", *Sensors*, vol. 21, no. 3, pp. 885, 2021, doi: 10.3390/s21030885.

[73] Hanai, Yuya and Nishimura, Jun and Kuroda, Tadahiro, "Haar-Like Filtering for Human Activity Recognition Using 3D Accelerometer", *In 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*, pp. 675-678, 2009, doi: 10.1109/DSP.2009.4786008.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

In this study, all authors contributed equally, from formulation of the problem to solution and analysis. Follow: www.wseas.org/multimedia/contributor-role- instruction.pdf

Sources of funding for research presented in a scientific article or scientific article itself

The authors did not receive support from any organization for the submitted work.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article

APPENDIX

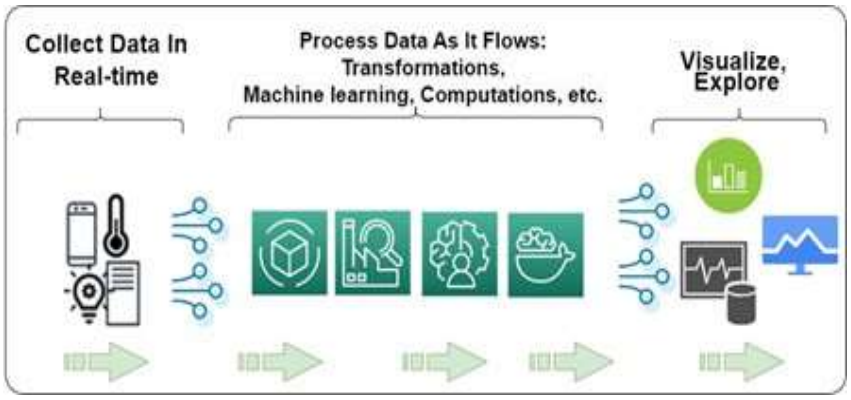

Fig. 1: Stream processing

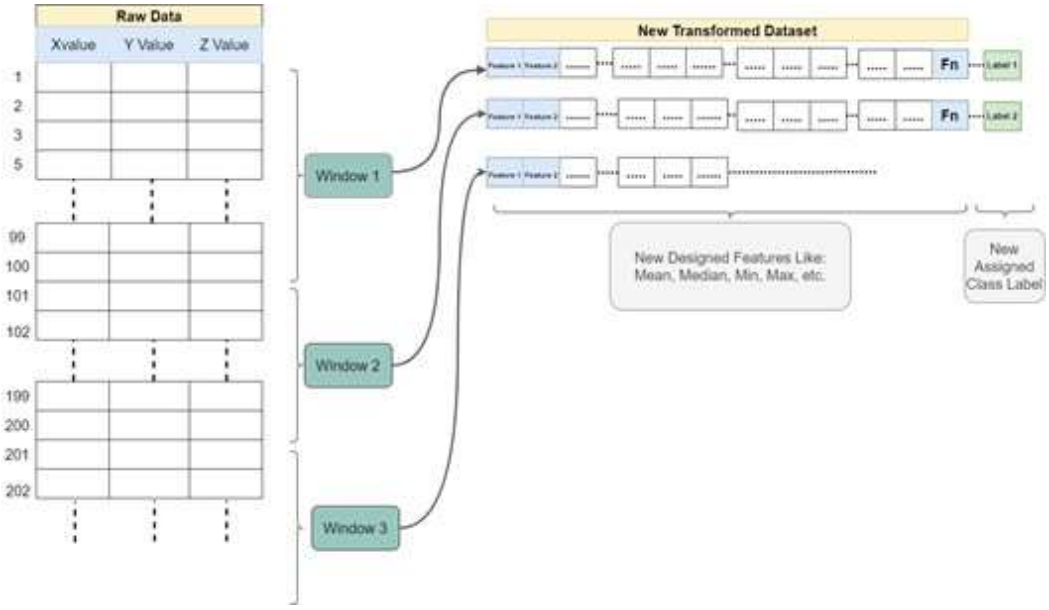

Fig. 2: Data transformation and feature engineering

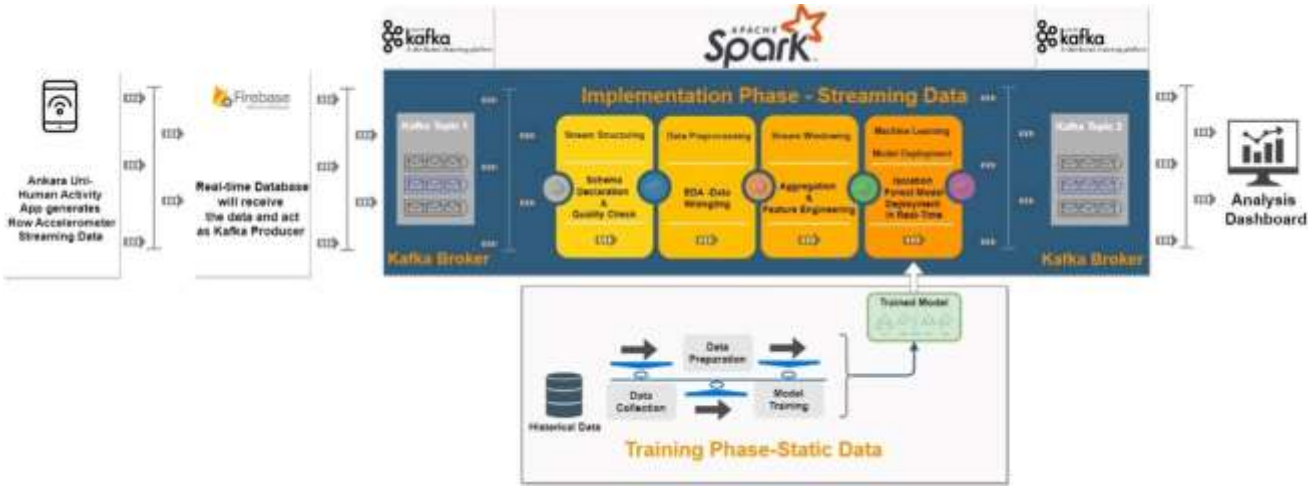

Fig. 3: An overview of the system implementation

Approach	Algorithms	Advantages	Disadvantages
Statistical	• Parametric methods - Gaussian Mixture - Regression • Non parametric methods - Kernel Density Estimation	• Easy to implement in spite of their limitations • Detect outliers efficiently if the correct probability distribution model is defined well	• The use of the Gaussian distribution incorrectly can produce false outliers • Effective when local outliers need to be detected • Computing costs are high when multivariate data is involved • Not applicable in a multidimensional scenario • Using non-parametric statistical methods in real-time is expensive
Density	• Connective-based Outlier Factor • Local Correlation Integral • Local Outlier Factor	• More suitable for local anomaly detection	• Choosing the right size and number of neighbors could be challenging • High computational cost • Not an ideal choice in data stream use cases • Suffer from the curse of high dimensionality
Distance	• k-nearest neighbor (KNN) • Abstract-C • Extract-n	• Tend to scale better in multidimensional space • Compute-efficient compared with statistical methods • Easy to comprehend • Don't rely on a specific distribution to fit the data	• High-dimensional data decreases their performance • Cannot deal with data streams • High CPU cost and CPU performance instability • Memory consumption is high
Clustering	• k-means clustering • DBSCAN • D-Stream • D-Cluster	• Robust to different data types. • Effective in collective outlier detection usually. • Can operate in an unsupervised mode. • DBSCAN has the ability find clusters of arbitrary shape.	• Rely on specifying number of clusters in advance • Optimized to find cluster in general anomaly • If some points don't create any cluster, it fails • Computationally expensive • Unsuitable for high-dimensional datasets
Isolation	• Isolation Forest	• Capable of detecting outliers in high dimensional data • Applicable for streaming data • Simple, fast, and efficient • Low memory requirement	• Hard to adapt with categorical data

Fig. 4: A comparison of the most common anomaly detection methods

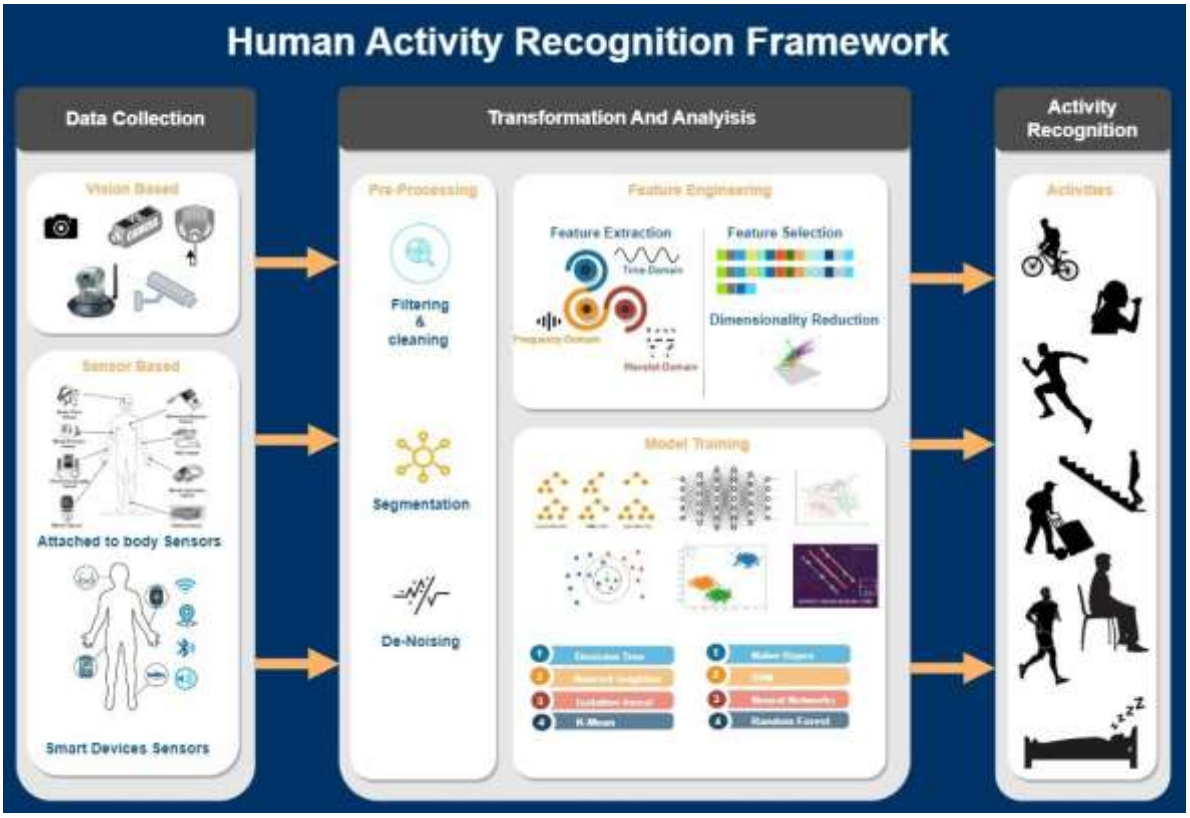

Fig. 5: Human activity recognition framework design

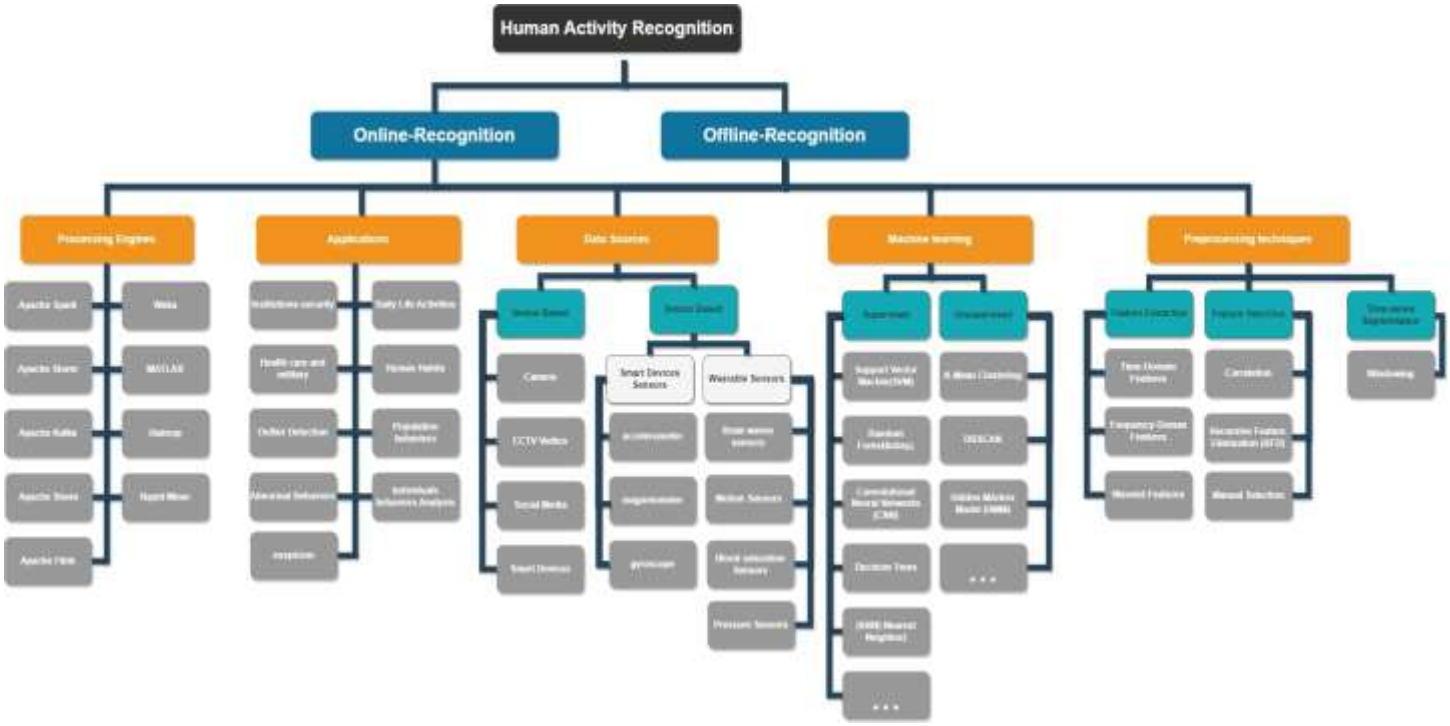

Fig. 6: Human activity recognition framework design