Anomaly Detection using ML in Real-Time Systems

Mamata Ganapati Naik, Assistant Professor Dr. Deeba K.

School of Computer Science and Applications REVA University, Bengaluru, India

DOI: https://doi.org/10.51244/IJRSI.2026.1306000240

Received: 14 June 2026; Accepted: 20 June 2026; Published: 03 July 2026

ABSTRACT

Modern day real-time systems have been applied in many domains like finance, healthcare, security, and industries, among others. Real-time systems are characterized by continuous creation of huge amounts of data, thus making manual detection of anomalies difficult. Anomaly detection is very essential as anomalies can be used to detect errors in real-time systems and even cyber attacks. In this study, an anomaly detection system using machine learning methods is presented. The proposed model adopts several methods like data preprocessing, feature extraction, supervised and unsupervised machine learning methods to efficiently detect anomalies. The system is able to analyze streams of data, therefore, providing fast and accurate detection of anomalies with minimal time delays.

Keywords: Machine Learning, Anomaly Detection, Real-Time Systems, Data Streams, Artificial Intelligence

INTRODUCTION

In the past few decades, real-time systems have emerged as a necessity for different industrial areas including health care, financial institutions, cybersecurity, and smart infrastructures. Real-time systems operate by processing information immediately without any delay. This includes banking where transactions take place instantaneously, and network security, which requires ongoing monitoring for suspicious activity. The continuous processing in real-time systems leads to the creation of large amounts of data at every second.

Due to the creation of such large volumes of data, it becomes challenging for anyone to monitor all of it at once. Even minor deviations might be overlooked, which could result in critical issues such as system malfunctioning, fraudulent activities, and cybercrimes. Here comes the need for anomaly detection. It refers to detecting any data point that does not conform to its normal pattern.

Anomaly detection in the past has been carried out using a set of fixed rules where predefined conditions have been put in place to detect abnormal activity. However, the rule-based approach has proven to be rigid in nature, hence ineffective in cases where new forms of anomaly occur since it does not adapt to changes in data.

Machine learning has proved to be a much more effective strategy that addresses the limitation of the traditional method. Machine learning algorithms work by learning patterns through training and hence do not rely on any fixed conditions. Machine learning algorithms adapt to changes in data and are thus effective in detecting anomalies regardless of the form.

Another critical factor involved in anomaly detection in real-time systems is the ability to detect the anomalies at high speed. Any delays in detecting anomalies could result in severe ramifications including monetary losses and system failures. Thus, not only should the system be able to detect accurately, but it should be able to do so at a relatively fast pace.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

The aim of this paper is to develop an approach for anomaly detection using machine learning. The objective will be to develop an anomaly detection system that is capable of handling data streams continuously, detecting any anomalies efficiently and promptly raising an alert.

PROBLEM STATEMENT

Real-time systems have found applications in areas ranging from e-commerce, networks monitoring to industry management, where huge amounts of data are produced all the time. It is difficult to monitor the data generated manually, and rule-based approaches usually fail when detecting unexpected abnormalities. This can cause delayed fault discovery, potential security breaches, and faulty decisions of the systems. Moreover, a delay in anomaly discovery may cause serious harm, ranging from financial loss to the failure of the systems themselves; meanwhile, too many false alarms may cause system unreliability. Hence, what is needed here is an intelligent mechanism which can automatically discover from the data, effectively process streaming data in real-time and detect abnormalities.

LITERATURE REVIEW

In recent years, there have been attempts made by many scientists in trying out various techniques in detecting anomalies, especially with the growing popularity of data-driven solutions. Initially, the most common technique involved the use of statistical approaches where behavior of normalcy is modeled mathematically and all deviations are deemed anomalies [1]. Though straightforward and convenient to use, they proved to be less efficient in handling complex situations.

With time, distance-based and clustering approaches emerged that offered better results than their predecessors [2]. Distance based and clustering methods rely on grouping similar data items and considering all ungrouped data points as anomalies. Even though they showed better performance results, they were still not efficient enough in tackling large volumes and dynamic data sets.

With the development of machine learning, more sophisticated algorithms were introduced. Supervised learning algorithms such as decision trees and support vector machines were used to classify the data points into two categories: normal and abnormal [5]. But supervised algorithms need labeled data, which may be challenging to acquire in practical scenarios.

This problem can be solved using unsupervised algorithms, including Isolation Forest [3] and Local Outlier Factor (LOF) [4]. Unsupervised algorithms do not need labeled data and are able to detect anomalies through understanding the characteristics of the data under normal conditions. These algorithms are widely used in applications such as fraud detection and cybersecurity. Moreover, it was proven that unsupervised algorithms were better suited for evolving and unknown datasets in comparative studies [17].

In addition to statistical techniques, recently deep learning methods, including autoencoders and neural networks, have been proposed as potential solutions for detecting anomalies [6]. Such methods are capable of discovering patterns that are too complex to analyze by traditional means, and thus they result in greater precision in their outputs. Replicator neural networks [12] and deep one-class classification [14] have become some of the more popular solutions for anomaly detection based on the above principles. At the same time, such algorithms require much more computational power than conventional methods.

Some recent works have also investigated anomaly detection from a real-time perspective, with streaming data being considered. The technologies used for processing streams of data include Apache Spark [8] and Apache Kafka [9], among others. Kernel recursive least squares have also been suggested as an approach to online detection of anomalies [11]. Although these solutions significantly improve the performance of anomaly detection systems, issues regarding their precision and minimizing errors still persist [7].

In addition, contemporary anomaly detection systems frequently utilize advanced packages and libraries, including Scikit-learn [10] and PyOD [19], which offer efficient implementations of different algorithms. This makes experimenting with various approaches more convenient for practical purposes. Moreover,

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

big data processing systems, such as MapReduce [13], have greatly contributed to making anomaly detection possible at scale.

The other key problem encountered in anomaly detection is the presence of imbalanced data sets, where there is an insufficient number of anomalies compared to normal data. Studies have indicated that conventional methods can be skewed toward the majority class, thus adversely affecting the detection process [20].

Furthermore, clustering-based anomaly detection systems have been considered to detect unusual activities on a network [18]. Such systems can be useful for cybersecurity applications as well.

In general, although many advances have been made regarding anomaly detection techniques, it remains necessary to develop a system that is fast, accurate, and adaptable enough for real-world use.

PROPOSED SYSTEM ARCHITECTURE

The system being suggested here is intended for detecting anomalies on real-time bases by continually analyzing data coming in through machine learning processes. The key point in the whole process will be building a system that would analyze data, find out the patterns in them, and then detect an anomaly immediately and independently of human input.

There are many modules in the system's architecture, each responsible for its own function. In this way, the system can provide uninterrupted flow of data analysis, which leads to speedy detection results.

1. Data Acquisition Layer

The first step for building such a system involves acquisition of data, which comes from various sources. This data may originate from different sensors, computer networks, financial transactions, and whatever real-time application you want. Because it comes in continuously, it is considered data stream as opposed to static data.

2. Data Preprocessing Module

Raw data from actual systems can be inaccurate or incomplete. Hence, data preprocessing plays an important role at this point. In this process, missing values in data are imputed, unwanted data is filtered out, and the data is formatted. This increases the efficiency of the machine learning model.

3. Feature Extraction and Selection

All features in data may not be relevant to finding anomalies in the system. Through this module, only necessary features are extracted that will enable anomaly detection. It enables faster detection through less data.

4. Machine Learning Model

This is the central component of the entire system. In this process, the machine learning algorithm learns what constitutes normal activity based on historical data. This training is then applied on live data coming into the system. Different algorithms can be used here, such as Isolation Forest, One-Class SVM, or Autoencoders. These models compare new data with learned patterns and detect deviations.

5. Real-Time Processing Engine

As the system operates on continuous data, a real-time processing component becomes essential for the implementation. This component helps process the data instantly upon receipt, minimizing any delays and enabling quick detection of the anomalies. Timely detection is vital in crucial applications.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

6. Decision and Alert Component

If the anomaly detector identifies an anomaly, this information is forwarded to the decision component. Based on a certain threshold, the decision is made about the abnormality of the identified behavior. In case the behavior is confirmed to be abnormal, an alert is triggered. Depending on the application, alerts may take the form of notifications, warning messages, and other outputs.

7. Feedback and Learning Component

For effective long-term operation, the system employs a feedback mechanism. The model is capable of changing depending on the input data. Such approach contributes to increased accuracy and reliability of the system.

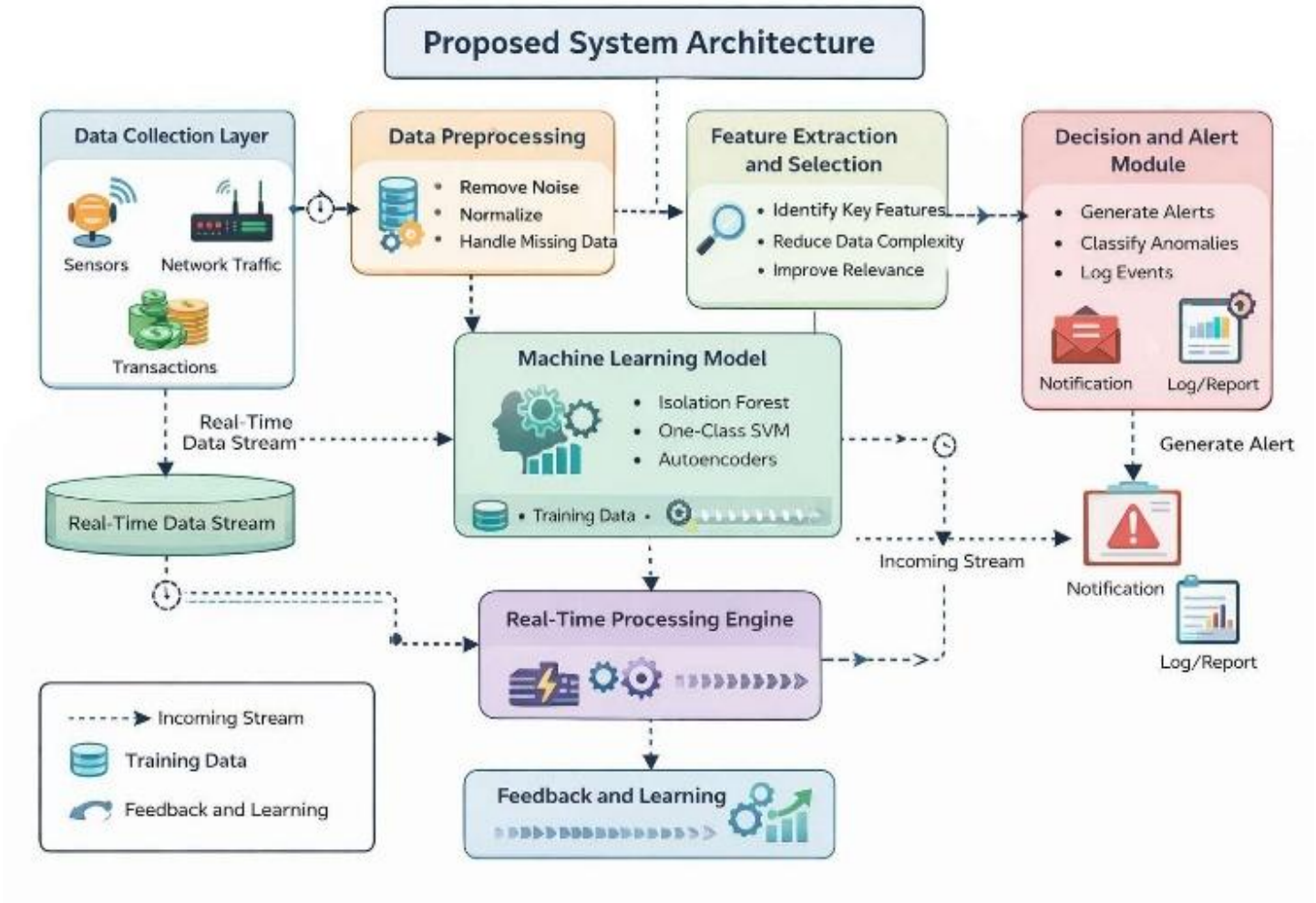

Fig 4.1 Proposed System Architecture

METHODOLOGY

The process of how the proposed system operates starts with the ongoing production of data from live feeds such as sensors or transaction information. As this is done at a very fast pace and there is usually much more information being processed, this data is initially run through the preprocessing step, where any extraneous noises and missing data are filtered out. This step is required due to the nature of real-world systems data which may not always be complete or consistent and its direct use could influence the model negatively.

After the preprocessing is done, data is further converted into a suitable format for analysis with only necessary features taken into consideration in order to facilitate faster processing and detection. Using historical data, the model is then trained by the system on how to detect normal behavior.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

In the aftermath of the training process, the system begins to analyze the data in real-time mode. For each new piece of data, it compares the observed behavior with the patterns that the algorithm learns. In case there is some unusual behavior detected in relation to what the system defines as 'normal', the data point is classified as an anomaly. Here comes into play a certain threshold system that helps to filter out insignificant anomalies.

Upon detecting an anomaly, the system generates immediate feedback in the form of alerting messages. It makes the timely response possible, especially when talking about cases such as fraud detection. The system continues analyzing data and learning about various patterns; therefore, the detection process improves with time.

HARDWARE AND SOFTWARE COMPONENTS

Anomaly Detection Using ML for Real-Time Systems comprises several components that help in monitoring, controlling, and automating the agricultural activity. All these components have been carefully chosen considering their effectiveness and ease of implementation.

A. Hardware Components

Table I Hardware Components Used in the Proposed System

Sl. No	Component	Purpose
1	Computer System / Workstation	Used for developing, training, and running the machine learning model
2	Processor	Minimum Intel i5 or equivalent
3	RAM	At least 8 GB
4	Storage	Minimum 256 GB SSD for faster data processing
5	Network Connectivity	Required for handling real-time data streams and cloud integration

1. ComputerSystem/Workstation

A standard laptop or desktop system is sufficient for development and testing. For large-scale real-time applications, a high-performance workstation or cloud server is recommended.

2. Processor (CPU)

A multi-core processor such as Intel i5/i7 or AMD Ryzen is preferred to handle parallel data processing and model execution efficiently.

3. Memory (RAM)

A minimum of 8 GB RAM is required, while 16 GB or higher is recommended for handling large datasets and real-time streaming without performance issues.

4. Storage (SSD/HDD)

SSD storage (256 GB or above) is preferred for faster data access and processing. It helps improve system performance during model training and data handling.

5. *NetworkConnectivity*

A stable internet connection is required for real-time data streaming, especially when using distributed systems or cloud-based services.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

B. Software Components

Table II Software Components Used in the Proposed System

Sl. No	Component	Purpose
1	Operating System	Windows / Linux / macOS
2	Programming Language	Python (Implementing ML models and logic)
3	Machine Learning Libraries	Scikit-learn, TensorFlow
4	Data Processing Libraries	Pandas, NumPy
5	Streaming Frameworks	Apache Kafka, Apache Spark
6	Development Environment	VS Code / Jupyter Notebook
7	Visualization Tools	Matplotlib / Power BI

1. Operating System

The system can run on Windows, Linux, or macOS. Linux is often preferred for large-scale and server-based deployments.

2. Programming Language

Python is used due to its simplicity, readability, and strong support for machine learning and data processing.

3. Machine Learning Libraries

Libraries such as Scikit-learn are used for implementing anomaly detection algorithms like Isolation Forest and LOF. TensorFlow or PyTorch can be used for advanced deep learning models.

4. Data Processing Libraries

Pandas and NumPy are used for data cleaning, preprocessing, and numerical operations.

5. Streaming And Big Data Frameworks

Apache Kafka is used for real-time data ingestion, while Apache Spark enables distributed data processing and low-latency analytics.

6. Development Tools

IDEs such as Visual Studio Code or Jupyter Notebook are used for coding, debugging, and experimentation.

RESULTS

The proposed real-time anomaly detection system was tested using simulated streaming data to evaluate its performance under different conditions. During the testing phase, the system continuously received incoming data and processed it using the trained machine learning model.

Table III Observed System Response During Testing

Parameter	Condition	System Response
Data stream rate	High incoming data	System maintained stable performance
Anomaly score	Above threshold	Anomaly detected and alert generated
Normal data pattern	Within the expected range	No action taken (normal operation)
Noise in data	Minor fluctuations	Filtered during preprocessing

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Sudden data spike	Abnormal spike detected	Immediate anomaly flag raised
Model Response Time	Continuous streaming	Low latency maintained
Data Volume	Large-scale input	Scalable processing without delay
Pattern Variation	Changing data behaviour	Model adapted and detected anomalies

However, it was noted that the model managed to detect any outliers in the data in real-time as the data passed through the model. Once there were any instances of abnormality in the data set, the model managed to detect the abnormalities and generate notifications instantly without experiencing any delays.

The reaction time of the model remained minimal despite an increase in the amount of data being used. This implies that the model can be applied in scenarios where there is a need to process massive amounts of data in real-time without interfering with the efficiency of the system.

Lastly, it is evident that the model behaved consistently during its operations despite using different input data sets.

From an overall perspective, the findings indicate that the system under consideration operates efficiently for anomaly detection. It minimizes the risks of late detection and can be used in real-world applications, including fraud detection, network monitoring, and sensor networks.

ADVANTAGES

Some of the significant benefits of the proposed real-time anomaly detection framework include the following. First, the framework can instantly detect anomalies while processing the data. The immediate detection and alert will enable stakeholders to take timely measures and prevent any risk of fraud or system failure.

Second, the use of machine learning models enables the framework to be highly accurate in detecting any anomalies. Through learning from the provided data, the accuracy of anomaly detection increases gradually, hence reducing any false alarms.

Finally, the real-time anomaly detection architecture is scalable. In other words, it is clear that no matter what volume of data is being used by the framework, its efficiency will not be reduced.

In addition, the implementation of real-time processing architectures guarantees that there is going to be low latency when it comes to the detection of anomalies. In fact, it should be said that the promptness of such an analysis is very important in some cases.

It is also essential to say that the system under discussion can be adapted to any area, whether it is cybersecurity, health care monitoring, or even industry-related processes.

LIMITATIONS

However, the real-time anomaly detection system suggested does have some limitations despite giving efficient results. First of all, the issue lies with high computational requirements. The system works with data that is provided constantly in real-time, which means it might require strong computing power or cloud computing support.

The other problem is that of the dependency on the quality of the data. If there is noise or missing information in the data, the model might give inaccurate results.The performance of the system also depends on the selection and tuning of machine learning models. Choosing inappropriate parameters or algorithms may reduce detection accuracy or increase false alarms. This requires some level of expertise and experimentation.

Furthermore, the real-time systems might encounter problems in coping with abrupt changes in the

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

patterns within the dataset, called concept drift. Failure to update the model on time could lead to reduced efficiency in the long run.

Moreover, incorporating the two can lead to an increase in the complexity of the system. The maintenance of such a system could demand technical know-how, which could prove difficult for novices.

CONCLUSION

The current study focused on developing an anomaly detection system using machine learning for use with dynamic data streams. Such a system was developed based on its ability to handle continuous streams of data to detect abnormality within a very short period of time – a requirement for several contemporary applications including fraud detection, cyber security, and IoT.

The machine learning-based approach used in this system provided it with sufficient balance in terms of speed and accuracy when dealing with the continuous stream of data. As a result, the system managed to detect any anomalies with relatively low latency while working efficiently with high volumes of data.

The results demonstrate that the system can operate reliably in real-time environments and adapt to different data conditions. It reduces the dependency on manual monitoring and provides a smarter way to identify potential risks at an early stage.

FUTURE SCOPE

The proposed anomaly detection framework can be improved by incorporating additional machine learning models like Autoencoders and Recurrent Neural Networks (RNN). These algorithms will enable the framework to learn complex and time-dependent relationships in the datasets, improving the overall efficiency of anomaly detection. Additionally, cloud computing platforms can play an important role in scaling and maintaining the proposed system due to its ability to store, process, and distribute massive amounts of information. In this case, the framework will be able to analyze large volumes of streaming data in real-time without impacting system performance.

Finally, another crucial step to improve the system is to implement a real-time dashboard that will provide visual feedback to the user on how the system behaves. It will also facilitate a user-friendly interaction experience where one can observe anomalies within the data. Continuous learning and updating of models can ensure stable results in the long run since there might be slight changes in the dataset over time. Lastly, connecting the system to IoT devices may be a potential way to increase the number of application domains.

REFERENCES

1. **V. Chandola, A. Banerjee, and V. Kumar**, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3,
2. pp. 1–58, 2009.
3. **C. C. Aggarwal,** *Outlier Analysis*, 2nd ed., Springer, New York, NY, USA, 2015.
4. **F. T. Liu, K. M. Ting, and Z.-H. Zhou**, "Isolation Forest," *IEEE International Conference on Data Mining (ICDM)*, pp. 413–422, 2008.
5. **M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander**, "LOF: Identifying Density-Based Local Outliers," *ACM SIGMOD International Conference on Management of Data*, pp. 93–104, 2000.
6. **M. Ahmed, A. N. Mahmood, and J. Hu**, "A Survey of Network Anomaly Detection Techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.
7. **I. Goodfellow, Y. Bengio, and A. Courville**, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.
8. **D. Pimentel, D. Clifton, L. Clifton, and L. Tarassenko**, "A Review of Novelty Detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.
9. **Apache Software Foundation**, "Apache Spark: Unified Analytics Engine for Big Data

Processing," Available: https://spark.apache.org
10. **Apache Software Foundation**, "Apache Kafka: Distributed Event Streaming Platform," Available: https://kafka.apache.org
11. **Scikit-learn Developers**, "Machine Learning in Python,"
12. Available: https://scikit-learn.org
13. **T. Ahmed, M. Coates, and A. Lakhina**, "Multivariate Online Anomaly Detection Using Kernel Recursive Least Squares," *IEEE INFOCOM*, 2007.
14. **S. Hawkins, H. He, G. Williams, and R. Baxter**, "Outlier Detection Using Replicator Neural Networks," *Data Warehousing and Knowledge Discovery*, 2002.
15. **J. Dean and S. Ghemawat**, "MapReduce: Simplified Data Processing on Large Clusters," *OSDI*, 2004.
16. **L. Ruff et al**., "Deep One-Class Classification," *International Conference on Machine Learning (ICML)*, 2018.
17. **Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau**, "Autoencoder-Based Anomaly Detection," *IEEE Conference*, 2017.
18. **A. Patcha and J. M. Park**, "An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.
19. **M. Goldstein and S. Uchida**, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms," *Pattern Recognition*, vol. 46, no. 5, pp. 1469–1483, 2013.
20. **K. Leung and C. Leckie**, "Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters," *ACSAC*, 2005.
21. **Y. Zhao, Z. Nasrullah, and Z. Li**, "PyOD: A Python Toolbox for Scalable Outlier Detection," *Journal of Machine Learning Research*, vol. 20, pp. 1–7, 2019.
22. **H. He, J. Yang, and S. Wang**, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.