Acta Cybernetica **27** (2025) 117–140.

Optimizing SAP Machine Learning-based Solutions through Custom API Integration

Georgina Asuah[ab], Arafat Md Easin[ac], and Tamás Orosz[ad]

Abstract

Rapid changes, dynamic consumer preferences, and evolving market trends are the hallmarks of the business environment. SAP HANA has emerged as a potent platform to meet this demand due to its resilient foundation for real-time data analytics and processing and in-memory processing architecture. This research aims to improve anomaly detection capabilities by integrating machine learning (ML) models into the SAP HANA Fiori web application. This will be achieved by developing a custom Application Programming Interface (API). The proposed solution integrates ML models with the SAP system using FastAPI, providing real-time insights and decision-making capabilities, by employing Local Outlier Factor (LOF) for anomaly detection. Multiple ML estimators were evaluated and the results indicate that LOF consistently outperforms other models, offering higher detection accuracy and computational efficiency. This research provides a practical framework for integrating machine learning-based anomaly detection into enterprise applications, addressing the limitations of SAP's built-in Predictive Analysis Library (PAL). To guarantee seamless performance and scalability, the API is deployed on Azure using Docker containers. This paper presents the capability of custom APIs to integrate ML models into enterprise systems, enhance operational efficiency, and establish a reliable framework for real-time anomaly detection as a practical solution. The article addresses challenges associated with API integration, scalability, and system configuration, providing valuable insights for enhancing the deployment of machine learning in enterprise applications. These findings offer valuable insights for organizations seeking to enhance their predictive analytics capabilities using modern AI-driven approaches.

Keywords: SAP HANA Fiori, machine learning, API integration, anomaly detection, Local Outlier Factor (LOF)

[a]Department of Data Science and Engineering, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
[b]E-mail: asuahgeorgina@inf.elte.hu, ORCID: 0009-0004-4390-7494
[c]E-mail: arafatmdeasin@inf.elte.hu, ORCID: 0000-0003-4014-9144
[d]E-mail: orosztamas@inf.elte.hu, ORCID: 0000-0003-0595-6522

DOI: 10.14232/actacyb.312225

1 Introduction

Organizations nowadays strive to derive actionable insights from their massive data sets, owing to prevalent digital technologies and big data [31]. The business world is characterized by rapid changes, evolving tastes of customers, and unpredictable market trends [35]. Companies in diverse sectors now consider real-time decision assistance a need rather than a luxury. SAP HANA emerged as a strong platform to address this need, due to its in-memory computing design and solid basis for processing and analytics of data in real-time [10].

However, a smooth integration of machine learning and artificial intelligence (AI) capabilities is necessary to fully explore SAP HANA's potential [9, 15]. Tasks like anomaly detection benefit greatly from this connection since ML models can detect variations in data trends, which can help spot vulnerabilities like fraud, system failures, or inventory shortages early. This link allows companies to quickly and easily conclude their data which will help them make better decisions. Unsupervised anomaly detection methods can be easily built using SAP HANA's Predictive Analysis Library (PAL) [22]. However, this approach has limitations, such as not letting the user modify the algorithm's settings or apply domain expertise for anomaly identification. This necessitates using custom APIs to enhance the precision and efficiency of anomaly detection.

Nowadays, ML algorithms utilize data analysis techniques to identify patterns and correlations in historical data, enabling the extraction of valuable information and the creation of algorithms [33]. Application Programming Interfaces are vital for connecting machine learning models to enterprise systems such as SAP HANA [27]. Organizations can efficiently tackle unique business difficulties by adapting ML models to their specific needs and requirements through these APIs [7]. Custom APIs offer flexibility in integrating specialized machine learning models tailored to the unique needs of a business, providing a means to optimize these models' deployment and scaling [26]. APIs act as connectors, enabling seamless communication between machine learning algorithms and SAP applications [16]. They allow data to flow efficiently between these systems, ensuring that ML models can be integrated without significantly disrupting existing workflows.

The capacity to promptly identify deviations from typical behavior is essential for the preservation of operational efficiency, security, and system performance in anomaly detection. The anomaly detection process can be automated and enhanced by the integration of machine learning models into SAP HANA through custom APIs, which can provide real-time insights that traditional rule-based systems may overlook [13]. This research investigates the potential of custom API integration to optimize SAP machine learning-based solutions, with a particular emphasis on the improvement of anomaly detection capabilities in enterprise environments.

The main motivation is derived from the constraints of the current SAP HANA PAL capabilities, which restrict customization and domain-specific tailoring. Anomaly detection is essential for identifying unusual patterns in data, including fraud, system malfunctions, and inventory discrepancies. Although SAP HANA's PAL provides fundamental anomaly detection capabilities, it is unable to integrate so-

phisticated ML models. In this study, we developed a custom API that incorporates sci-kit-learn's Local Outlier Factor (LOF) anomaly detection model to overcome these constraints and offer a more flexible anomaly detection solution. The aim of this research is the development of a resilient custom API using FastAPI to integrate machine learning models, specifically the LOF, with SAP Fiori Web, and the deployment of this solution on Azure Kubernetes Service (AKS) to guarantee scalability, security, and high availability. Improve the pace and precision of decision-making in real-time enterprise settings.

The main contributions of this study are summarized as follows:

- Developing a FastAPI-based model for the integration of machine learning anomaly detection with the SAP HANA Fiori application.
- Optimizing anomaly detection where various machine learning models were analyzed, revealing that the LOF exhibits enhanced accuracy, recall, and ROC AUC.
- Employing containerized cloud-based deployment through Docker and AKS to enhance scalability, security, and high availability.
- Designing an interactive interface for seamless integration of SAP HANA Fiori, enabling real-time anomaly detection within enterprise SAP applications.

2 Literature Review

Applying machine learning models for anomaly detection within SAP systems is becoming more popular as organizations strive to become more operationally efficient, reduce risks, and improve data-oriented decision-making [4, 12]. It has been noted that anomaly detection is important for any enterprise system as it involves detecting suspicious activity.

Anomaly detection methods are numerous, with some basic techniques extending to the use of artificial intelligence in application. Traditional statistical techniques such as Z-score and boxplot-based outlier detection are commonly used [18]. Techniques such as these establish a cutoff point based on mean-variance or other statistical moments such as quantiles. A Z-score, for example, tells how many standard deviations a given observation is away from the average, with higher scores meaning that they are closely related to anomaly activity [8]. However, these methods do not work satisfactorily with multi-dimensional or other complex data distributions. These methods are quite simple to adopt but the gutter lies on the prerequisite of a certain type of distribution which effectively dismisses them on dynamic or complex spheres.

Machine learning methodologies have become increasingly popular owing to their capacity to represent intricate patterns. For instance, Support Vector Machines (SVM) have proven effective in anomaly detection by identifying a hyperplane that separates typical instances from atypical ones [28]. Similarly, neural

networks, especially autoencoders, have found widespread application, as these models reconstruct input data, with significant reconstruction errors signaling the presence of anomalies [25]. Although these models are adaptable, they typically necessitate substantial datasets for training and may struggle with limited interpretability. Ensemble techniques, such as Isolation Forest and Random Forest, have also been employed to improve anomaly detection. Isolation Forest isolates anomalies through recursive partitioning, making it particularly effective for high-dimensional datasets [17]. While ensemble methods provide robustness and enhanced generalization, they also introduce increased computational complexity and often necessitate careful hyperparameter tuning to achieve the best performance.

Custom API integration is essential for embedding machine learning-based anomaly detection models within SAP systems. APIs are the interface between ML models and enterprise systems, allowing for smooth data transfer and enabling real-time predictions. According to [11], RESTful APIs are frequently used to expose ML models, providing a standardized method for communication between SAP systems and ML services. These APIs should accommodate various data formats (e.g., JSON, XML) and feature clear endpoints for model training, inference, and monitoring. Creating custom APIs for machine learning integration necessitates adherence to some fundamental principles. Best practices for API integration emphasize several key considerations as described by [5, 34]. Firstly, versioning is critical to guarantee backward compatibility as APIs progress. Secondly, adhering to RESTful principles and using JSON-based communication can make APIs more adaptable and easier to integrate with different applications. Lastly, security is essential, especially when integrating APIs into enterprise systems. Implementing encryption, authentication, and role-based access control mechanisms is crucial to protect sensitive information.

As machine learning models evolve, continuous deployment pipelines should be implemented to automate model updates in production environments [3]. In [21], the authors proposed a distributed and unified API service for machine learning models that helps ensemble multiple models. This results in better predictions and benefits such as wider availability, greater usability, and lesser resource constraints. [24] tackled the issue of developing user-friendly ML APIs, particularly for beginners. Their research centered on examining how the Kaggle community utilizes scikit-learn, a popular ML API. The work of [23], discussed a case study showing how they integrated an SAP ERP system with an external web service through API access, illustrating the use of algorithms and transactions within SAP ERP.

Enhancing SAP machine learning solutions for anomaly detection involves utilizing various methods, including custom API integration. Creating custom APIs is essential for linking machine learning models with enterprise systems, simplifying complicated processes, and improving user experience. Custom APIs will continue to be essential for achieving smooth and scalable ML integration in enterprise systems as organizations delve into AI-powered solutions.

3 Methodology

The proposed solution is intended to capitalize on the machine learning models available in the esteemed scikit-learn (sklearn) library. The SAP HANA Fiori solution's proposed implementation employs machine learning models from the sklearn library to improve predictive modeling and data analysis. The FastAPI framework is implemented to incorporate predictive capabilities within a scalable and accessible application programming interface. The Azure cloud service provider is selected for its seamless integration with FastAPI and robust infrastructure, which is where the API deployment is orchestrated. The API that has been finalized, functions as a connection between the SAP HANA Fiori web application and the machine learning model. It improves the web application's functionality by integrating intelligent decision-making capabilities that are based on the predictions of the ML model. A critical component of the operational strategy is integrating an anomaly detection API into the SAP HANA Fiori web application.

3.1 System Architecture

The system architecture integrates five key components: SAP HANA, SAP Fiori, FastAPI, Azure Kubernetes Service (AKS), and Azure Container Registry (ACR). This design ensures flawless interaction between enterprise data management, anomaly detection, and cloud-based deployment. Figure 1 provides a detailed view of the system's flow, showcasing how user interactions in SAP HANA Fiori trigger anomaly detection through the custom API. The entire process starts with SAP HANA, which stores and preprocesses the dataset before sending it via OData services to the FastAPI backend. A Local Outlier Factor (LOF) model is hosted by FastAPI to evaluate incoming data and produce anomaly predictions in real time that are returned in JSON format. Azure Container Registry manages containerized instances of the FastAPI application, guaranteeing version-control, and secure image storage. AKS coordinates scalable deployment, integrating HTTPS encryption and token-based authentication for secure operations and dynamically adjusting resources to meet demand.

Afterwards, the processed data are sent to SAP Fiori, which offers an easy-to-use, role-based interface for interactive anomaly analysis and prediction visualization. Using SAP platforms (HANA, Fiori) for data handling and user interaction and Azure Cloud Services (ACR, AKS) for robust infrastructure management, the architecture prioritizes modularity. It is secured by Role-based Access Controls (RBAC) and built for smooth scalability in enterprise settings.

3.1.1 SAP HANA

SAP HANA (High-performance Analytic Appliance) is an advanced in-memory database and application development platform designed for processing large volumes of real-time data [20]. In-memory processing stores data directly in the main memory of a system rather than on traditional disk storage, and this significantly

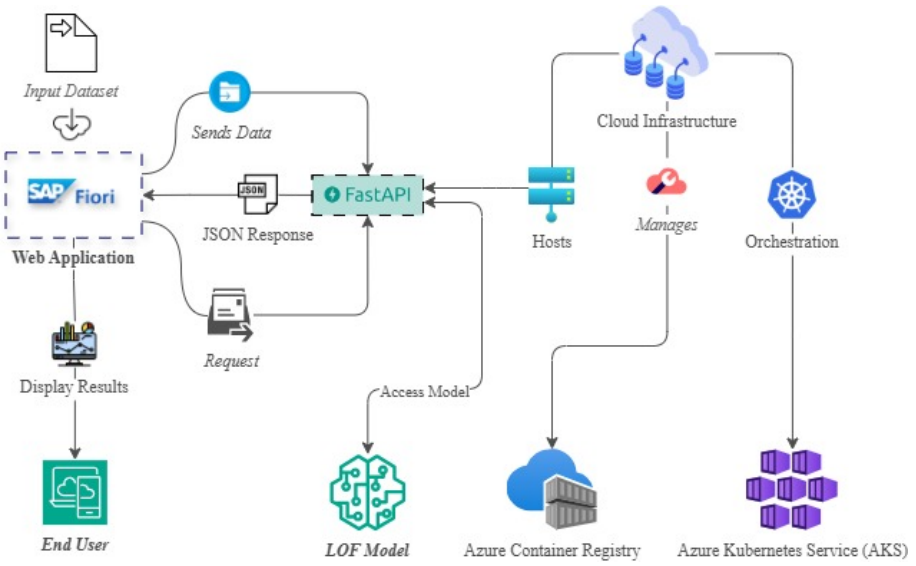

Figure 1: Overall architecture of the proposed system

enhances the processing speed for analytics and transactional workloads. It reduces the time taken to fetch data and accelerates computations by avoiding the latency related to disk I/O operations. Its Integrated Development Environment facilitates the creation of applications with the SAP HANA Deployment Infrastructure (HDI) containers, enabling smooth integration and data administration. Through its Predictive Analysis Library (PAL) [30], it facilitates predictive analytics, allowing developers to apply a variety of machine learning algorithms directly. The data source and preparation engine for this study is SAP HANA, which also prepares datasets and stores them in HDI containers for convenient access. It makes effective use of OData services to move data to external systems, such as the custom API.

3.1.2 SAP Fiori

SAP Fiori is a user experience (UX) platform for communicating intuitively and easily with enterprise systems through role-based interfaces [29]. It provides a responsive workflow for users on different devices through user-centered design, hence making laborious jobs less straining and hard business processes much easier. This work integrates it with the anomaly detection system using FastAPI, where the user can trigger the analysis in real time and in several directions with dynamic dashboards. Using SAPUI5 in the development, this platform provides much-needed customization possibilities according to organizational needs. The presented research extends the default functionality of SAP Fiori by incorporating new components, which interface directly with the anomaly detection API and showcase its adaptability in an advanced analytics context.

3.1.3 FastAPI

This study uses a scikit-learn LOF model to detect anomalies in real-time, and the real emotional core is a FastAPI-based backend. Incoming queries from SAP Fiori are processed by the API. It then uses the trained LOF algorithm to check the data for abnormalities and delivers predictions in standardized JSON format via RESTful endpoints. The system uses asynchronous request processing to maximize speed, guaranteeing low latency responsiveness and good scalability. This architecture allows anomaly scores to be dynamically shown on business dashboards by bridging the gap between machine learning algorithms and SAP Fiori's user interface.

3.1.4 Azure Container Registry

The centralized location for managing and storing Docker container images related to the FastAPI application is the Azure Container Registry. Immutable image tags provide strong version control, while integrated vulnerability assessment and role-based access restrictions guarantee safe deployment. ACR provides automated continuous integration and deployment (CI/CD) pipelines. It also and enables smooth connection with Azure Kubernetes Service by simplifying image distribution and authentication. This preserves adherence to company security rules while guaranteeing regular, auditable upgrades to the anomaly detection system.

3.1.5 Azure Kubernetes Service

The containerized FastAPI application is deployed and managed using Azure Kubernetes Service, which facilitates high availability and smooth scaling to satisfy business needs. Through auto-scaling capabilities, it automatically adjusts workloads in response to traffic changes, guaranteeing optimal resource use. To protect API endpoints and user interactions, AKS incorporates strong security mechanisms. These include token-based authentication using Azure Active Directory (AAD) and HTTPS encryption via ingress controllers. The platform supports zero-downtime upgrades and maintains a secure, auditable pipeline by utilizing Azure Container Registry for image retrieval and deployment. AKS is positioned as the foundation for production-ready anomaly detection processes because of its fault tolerance, scalability, and enterprise-grade security.

3.2 Data Preparation

The dataset[1] was the transactional sales data from SAP HANA (shows in Figure 2), preprocessed to remove null values and categorical anomalies. The sales transaction data from SAP HANA was deployed into a NativeDevelopment HDI container. This container functions as a repository for structured data, which is indispensable for developing and training machine learning models. The Multi-Target Application (MTA) paradigm was employed to import table definition files and construct

[1] https://webide.h08z.ucc.ovgu.de/watt/index.html

the MTA project, which initiates the configuration process. A `node.js` module with XSJS support was used to establish OData services, guaranteeing connectivity between the database and auxiliary services. The SAP HANA Fiori application enables data transfer to the API to facilitate real-time decision-making. Categorical variables, including Currency and Product, were encoded through Label Encoding, whereas numerical variables underwent standardisation via StandardScaler to maintain scale-invariance in model performance.

This study employed Stratified 10-Fold Cross-Validation to evaluate model performance in a robust and generalisable way. This approach guarantees that each fold maintains the same anomaly distribution as the complete dataset, which is essential in tasks involving imbalanced anomaly detection. Training and testing were conducted iteratively for each fold, with performance metrics averaged across the folds. A rigorous cross-validation process was employed on various anomaly detection models, such as Local Outlier Factor, One-Class SVM, Isolation Forest, and Robust Covariance, to determine the most effective estimator.

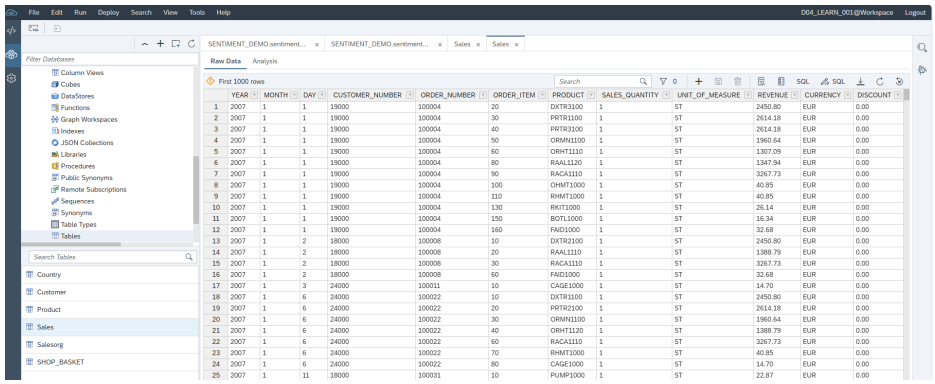

Figure 2: Snapshot of the sales dataset

3.3 Anomaly Detection Algorithms

In this paper, we have applied four machine-learning algorithms: Robust Covariance, Isolation Forest, One-Class Support Vector Machine, and Local Outlier Factor using the scikit-learn toolkit.

3.3.1 Robust Covariance

This method detects outliers by fitting data distribution through a robust estimation of covariance (e.g., Minimum Covariance Determinant) [1]. It calculates Mahalanobis distances to identify deviations from normality, making elliptical assumptions regarding data distributions. While performing well on low-dimensional Gaussian-like data, in high-dimensional data spaces, performance degrades since covariance estimation becomes unstable. The contamination parameter was set

to contaminatiset=0.05 which determines the predicted fraction of outliers and is important to its success. A mismatch in this parameter results in over- or under-detection. This scikit-learn method is intended to find outliers in datasets with a Gaussian distribution. It successfully detects outliers that depart from the center distribution by modeling the data and fitting an ellipse to it. When the data follows elliptical assumptions, this approach works well.

3.3.2 Isolation Forest

Isolation Forest is a tree-based ensemble method that effectively isolates anomalies by recursive partitioning. A process utilizing the fact that anomalies require fewer splits to be separated due to sparsity in the feature space [18, 32]. Though effective on high-dimensional data, it scales poorly with dataset size. The predicted percentage of outliers is indicated by the contamination parameter, set to contamination=0.05 in this implementation. The number of tren_estimators controls the number of trees which was set to 100 in this experiment to balance speed and accuracy.

3.3.3 One-Class SVM

This technique, which assumes anomalies are few and unique, finds anomalies by learning a decision border around normal data [19]. The choice of the kernel (radial basis function, for example) and hyperparameter tuning, specifically, nu (contamination estimate), kernel, and gamma, determine how successful it is. The implemented One-Class SVM algorithm uses the hyperparameters nu=0.05, kernel='rbf', and gamma='auto'. This is well suited for cases when the anomalies are well separated and kernel parameters match the inherent structure of the data.

3.3.4 Local Outlier Factor

This algorithm estimates the density of every point as a function depending on its k nearest neighbors. If the point's local density is lower compared to its k nearest neighbors, then it can be labeled as an anomaly. This reachability Distance to smoothen out this density notion can be formalized as the maximum actual distance between the two points, or the k^{th} nearest neighbor distance. [14]. Local Reachability Density (LRD) [2] is the inverse of the average reachability distance of a point's neighbors. LOF creates a score ratio between LRD for each point and an average of that same neighborhood around it. Scores greater than 1 would mark possible anomalies. The usefulness of LOF resides in the fact that it is a non-parametric technique with no assumptions of particular data distributions. This algorithm handles datasets with different densities more efficiently than global methods like the Z-score by evaluating the local density deviation of a data point about its neighbors. The n_neighbors (denoted as k) option adds versatility by enabling customization for various anomaly features. To find novel abnormalities in new data points The following hyperparameters were used: n_neighbors=80, contamination=0.05, metric='manhattan', and novelty=True.

The base algorithm, the LOF, was selected because of its versatility and performance. The ability of the LOF model to detect local density deviations and identify outliers in complex datasets led to its selection for deployment. Because it is non-parametric, it can adapt to different data distributions. When anomalies differ in degree from the norm, LOF performs exceptionally well. The pseudocode for the LOF algorithm is presented in Algorithm 1.

Algorithm 1: Local Outlier Factor (LOF) Algorithm

Input : Dataset $D = \{x_1, \ldots, x_N\}$; number of neighbors k; contamination level τ; distance metric d; novelty detection flag

Output: Anomaly labels $L = \{l_1, \ldots, l_N\}$, where $l_i \in \{\text{normal}, \text{outlier}\}$

foreach *point* $p \in D$ **do**
  Compute distance matrix M where $M_{ij} = d(p_i, p_j)$
end
foreach *point* $p \in D$ **do**
  Find k^{th} nearest neighbors: $\text{kNN}(p) \leftarrow \text{sort}(M_p)[1{:}k+1]$
end
foreach *point* $p \in D$ **do**
  foreach $q \in kNN(p)$ **do**
    Compute reachability distance:
    $\text{reach-dist}(p, q) \leftarrow \max(d(p, q), \text{distance}(q, q_k))$
  end
end
foreach *point* $p \in D$ **do**
  Compute local reachability density:
  $\text{LRD}(p) \leftarrow \frac{1}{\text{avg}(\{\text{reach-dist}(p,q) \mid q \in \text{kNN}(p)\})}$
end
foreach *point* $p \in D$ **do**
  Compute LOF score:
  $\text{LOF}(p) \leftarrow \frac{\text{avg}(\{\text{LRD}(q) \mid q \in \text{kNN}(p)\})}{\text{LRD}(p)}$
end
Determine threshold θ using contamination level τ
foreach *point* $p \in D$ **do**
  if $LOF(p) > \theta$ **then**
    $l_p \leftarrow$ outlier
  else
    $l_p \leftarrow$ normal
  end
end
return L

3.4 Performance Evaluation

The performance metrics and evaluation protocols used in this study to precisely assess the efficacy of the anomaly detection models are as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

The percentage of correctly categorized cases (both normal and abnormal) is quantified.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Evaluate the percentage of actual anomalies among all predicted anomalies to determine how well the model prevents detection errors.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

Assesses the model's ability to prevent false detections while identifying the majority of true anomalies.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Represents the balance between false positives and false negatives by taking the harmonic mean of precision and recall.

Execution Time: The Total amount of time (in seconds) needed for both model inference and training is essential for evaluating scalability.

Area Under the ROC Curve (AUC-ROC):

$$\text{AUC} = \int_0^1 \text{TPR}(x)\,dx \quad \text{or equivalently,} \quad \text{AUC} = \sum(\text{TPR} \times \Delta\text{FPR}) \tag{5}$$

The model's capability to distinguish anomalies from normal instances at various thresholds of classification.

$$\text{TPR} = \frac{TP}{TP + FN}; \quad \text{FPR} = \frac{FP}{FP + TN} \tag{6}$$

The assessment of anomaly detection models is predicated on several crucial terms: False Negatives (FN) are anomalies that the model failed to identify, False Positives (FP) are normal cases that were mistakenly identified as anomalies, True Negatives (TN) are normal instances that were correctly classified, and True Positives (TP) are anomalies that were accurately identified. Overall, anomaly classification based on threshold values determined by contamination levels. The evaluation employed a stratified 10-fold cross-validation approach to enhance robustness and generalizability. This method ensures that each fold preserves the distribution

of anomalies across the dataset, thereby mitigating bias in imbalanced learning tasks. Performance metrics, including accuracy, precision, recall, F1 score, and ROC AUC, were calculated using Scikit-learn's classification tools and averaged in all folds to obtain a reliable estimate of the effectiveness of each model.

4 Results and Discussion

This work involves the creation of a REST API using Python to enhance SAP ML-based solutions. The experimental results are shown and discussed below. This application consumes OData API/Service and loads data from the system. The experiment evaluated various machine learning models for anomaly detection, including Robust Covariance (RC), Isolation Forest (IF), One-Class SVM (OCSVM), and Local Outlier Factor (LOF) from the scikit-learn. To test the models for anomaly detection, the sales transaction data was used to evaluate the estimators. Table 1 compares the performance metrics of evaluated models, highlighting LOF's superior results. The hyperparameter specification for LOF includes k (number of neighbors): 80, Contamination: 0.05, metric: 'manhattan', Novelty: True. The Python pickle library saves the trained model for reuse via API.

4.1 Model Selection for Anomaly Detection

Our primary goal was to deliver robust outlier detection within SAP's inherently time-critical transaction environment, where prediction latency must remain minimal. Classical methods such as RC, IF, OCSVM, and LOF offer sub-second inference times and have demonstrated very strong detection performance in our benchmarks. We therefore prioritized these algorithms to ensure both speed and accuracy under production constraints.

Table 1: Comparison of different anomaly detection models evaluated on the full feature set using 10-fold cross-validation

Model	Acc	Precision	Recall	F1-Score	ROC AUC	Execution Time
RC	0.989	0.898	0.900	0.898	0.941	4.3s
IF	0.983	0.834	0.828	0.829	0.910	4.73s
OCSVM	0.969	0.674	0.756	0.711	0.868	**1.48s**
LOF	**0.991**	**0.911**	**0.920**	**0.914**	**0.958**	9.37s

Finding the optimum anomaly detection approach necessitates balancing sensitivity to class imbalance, computational efficiency, and accuracy [6]. Table 1 compares four popular anomaly detection models (RC, IF, OCSVM, and LOF) for all the features using 10-fold cross-validation, across accuracy, precision, recall, F1-score, ROC AUC, and execution time. Whereas, LOF achieves the best overall detection quality with the highest accuracy (0.991), precision (0.911), and recall

(0.920), yielding the top F1-score (0.914) and ROC AUC (0.958). However, this superior performance comes at the expense of speed: LOF is the slowest, requiring 9.37s to execute.

In contrast, OCSVM is the fastest (1.48s) but delivers the weakest detection metrics (accuracy 0.969, precision 0.674, recall 0.756, F1-score 0.711, AUC 0.868), making it less reliable for high-stakes anomaly identification. Both RC and IF offer more balanced trade-offs: RC provides strong recall (0.900) and accuracy (0.989) with moderate precision (0.898) and AUC (0.941) in 4.30s, while IF yields respectable accuracy (0.983) and precision (0.834) in 4.73s but slightly lower recall (0.828) and AUC (0.910).

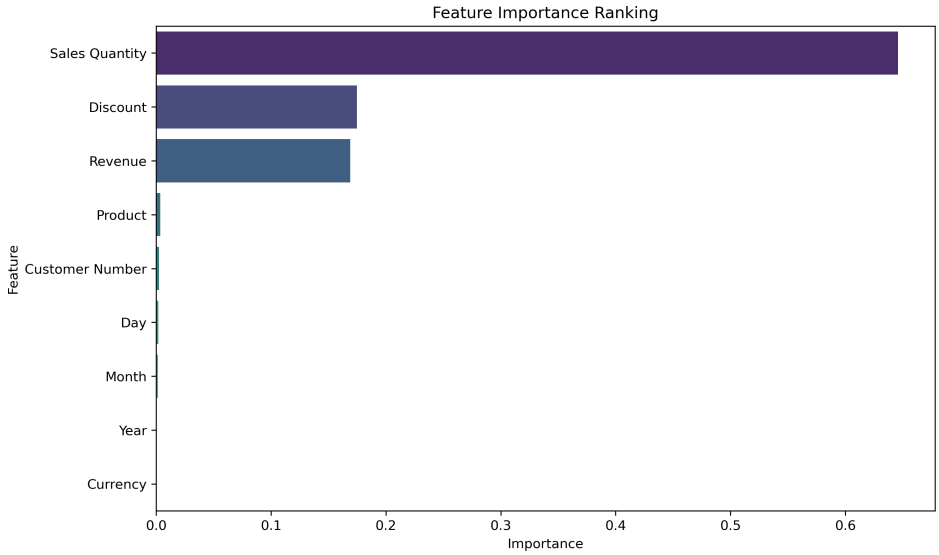

Figure 3: Feature importance from the complete feature set

After experimenting with the complete feature set on our anomaly-detection models, we utilized a RandomForestClassifier for feature-importance ranking to choose importance features. During the feature selection, three features were dropped including Order Number, Order Item, and Unit of Measure. As Figure 3 illustrates, Sales Quantity, Discount, and Revenue together contribute over 90% of the overall importance, whereas Product, Customer Number, Day, Month, Year, and Currency have only a little effect. Thus, we re-trained all our outlier detector models based on 9 features out of 12 features.

Table 2 demonstrates the superiority of the LOF even when using only significant features with 10-fold cross-validation. LOF achieves the highest accuracy (0.993) and outperforms all other models in precision (0.927), recall (0.932), F1-score (0.928), and ROC AUC (0.964). These gains translate into a more reliable detection of anomalies with fewer false positives and false negatives, making LOF the strongest choice for scenarios where detection quality is paramount.

Table 2: Comparison of different anomaly detection models evaluated on the significant features using 10-fold cross-validation

Model	Acc	Precision	Recall	F1-Score	ROC AUC	Execution Time
RC	0.987	0.868	0.876	0.871	0.934	10.66s
IF	0.986	0.869	0.864	0.865	0.928	4.11s
OCSVM	0.972	0.699	0.784	0.737	0.882	**1.34s**
LOF	**0.993**	**0.927**	**0.932**	**0.928**	**0.964**	7.45s

IF and RC both hover around 98.6–98.7% accuracy which is closer to LOF but fall behind LOF in other key metrics. IF delivers a slightly faster execution time (4.11s vs. LOF's 7.45s) yet its precision (0.869), recall (0.864) and ROC AUC (0.928) are notably lower, indicating less consistent anomaly coverage. RC, while yielding solid accuracy (0.987), requires the longest runtime (10.66s) and offers lower recall (0.876), precision (0.868) and AUC (0.934) compared to LOF. This makes it less attractive for both speed-critical and high-performance use cases. OCSVM exhibits the fastest inference (1.34s) but at the cost of significantly reduced detection quality (accuracy 0.972, F1-score 0.737, AUC 0.882). Its poor balance between precision (0.699) and recall (0.784) underlines why it is unsuitable for applications demanding both reliability and robustness.

LOF strikes the optimal balance and achieves peak anomaly-detection performance across all major metrics while maintaining acceptable latency. Overall, if detection quality is paramount and latency is less critical, LOF is recommended. However, the LOF method is particularly well-suited for mission-critical applications where minimizing undetected anomalies is a top priority, such as fraud detection and system monitoring, given its improved recall and processing efficiency. Therefore, it was chosen as the base method for this study due to its balanced performance metrics, especially in scenarios that need both scalability and accuracy. For time-sensitive scenarios where some performance can be sacrificed, RC or IF may be preferable, with OCSVM reserved only for cases demanding the fastest inference despite lower accuracy.

Figures 4 and 5 present ROC-curve comparisons for our four anomaly detectors, including RC, IF, OCSVM, and LOF; first on the complete full-feature set and then on the reduced significant-feature set. In Figure 4 (all features, 10-fold CV), LOF's ROC curve consistently lies above the others, achieving an AUC of 0.9576. This steep rise toward the top-left corner reflects LOF's excellent true-positive rate at very low false-positive rates, confirming its superior recall and precision trade-off (highest F1-score). RC follows closely with AUC = 0.941, indicating strong overall discrimination but slightly less sensitivity at low FPR. IF attains AUC = 0.9096, demonstrating good but not top-tier performance, while OCSVM trails behind (AUC = 0.8682), consistent with its lower precision and recall.

After pruning the less important features, Figure 5 shows that all models' ROC curves tighten and AUCs improve except for RC: LOF increases to 0.9640, IF to

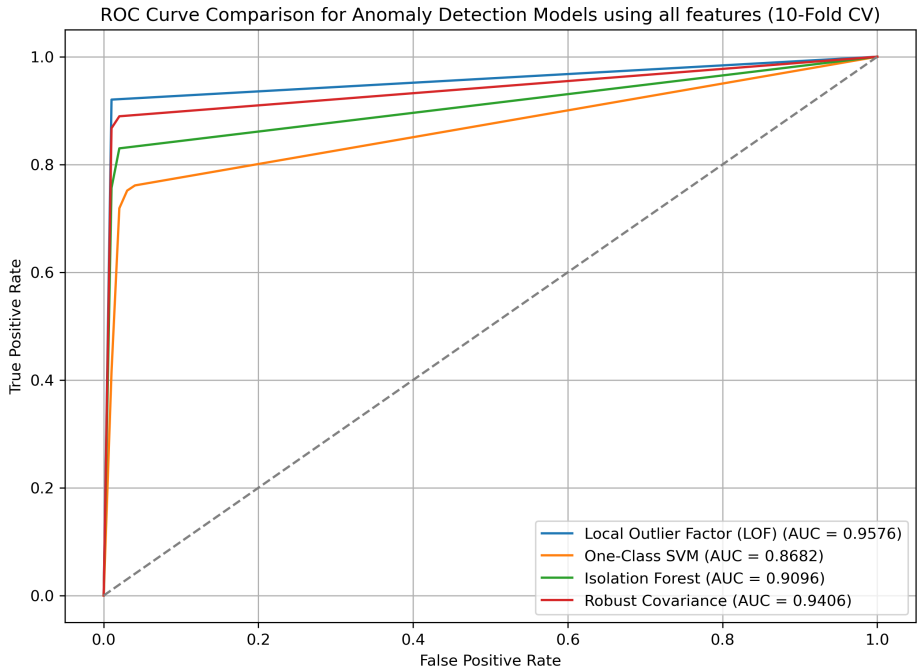

Figure 4: Comparison of ROC curves for different anomaly detection models with complete feature set

0.9284, and OCSVM to 0.8828 while RC decreases to 0.9346. Notably, LOF's curve becomes even sharper, underscoring that dimensionality reduction enhances its anomaly-separation power. IF benefits substantially as well, losing much of its gap to LOF while retaining fast inference. OCSVM also show modest gains, though OCSVM remains the weakest overall. However, RC shows a slight reduction in all performance metrics. These plots demonstrate that LOF is the top performer under both feature sets. Eliminating the three insignificant predictors further boosts all models' ability to distinguish anomalies from normal data except RC; especially improving LOF and IF in practical, real-time settings.

Figure 6 illustrates how a significant, relevant feature affects the model's functionality. Plotting the data demonstrates how changes in this characteristic correlate to changes in anomaly scores, hence improving the ability to distinguish between normal and anomaly cases. Notably, the graphic highlights the model's crucial role in reducing false negatives by showing that as the feature value rises, the model achieves increased detection accuracy and improved recall. Having all factors considered, the figure emphasizes how important the feature is for maximizing the anomaly detection procedure.

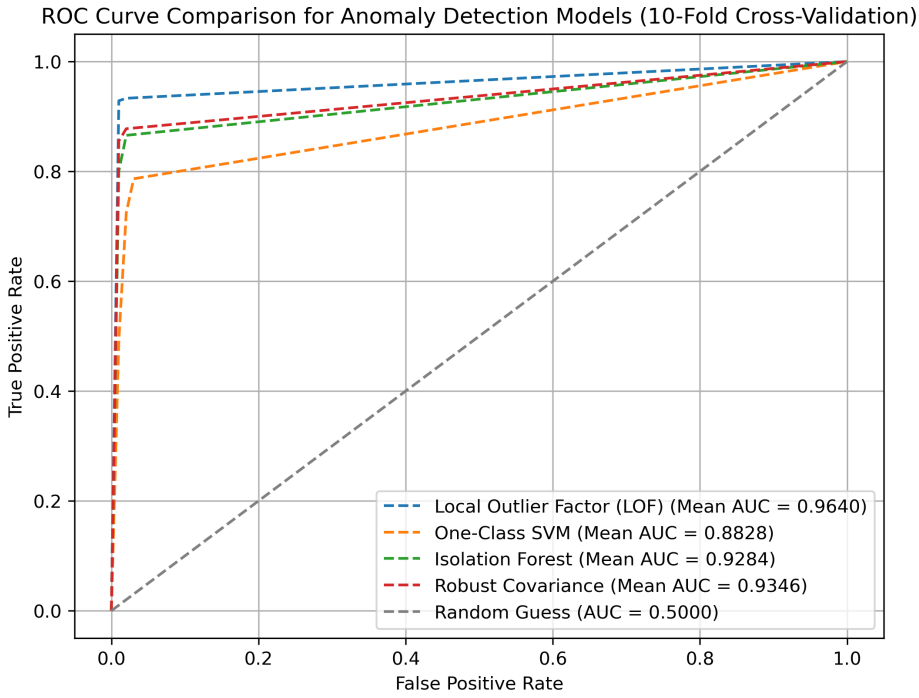

Figure 5: Comparison of ROC Curves for Different Anomaly Detection Models with significant features

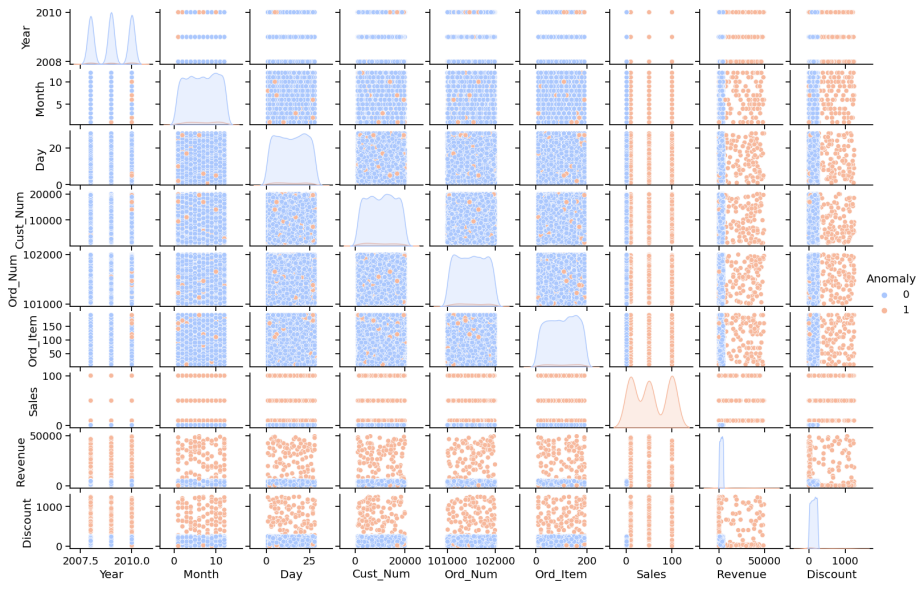

Figure 6: Visualizing sales patterns and anomalies

Figure 7 is a heatmap that shows a sales dataset's feature correlation matrix. It displays how strongly and in which direction linear correlations exist between two variables. The heatmap aids in uncovering potential correlations for feature engineering and further study. While execution time stays efficient, higher feature values are linked to increases in accuracy, recall, and the F1 score. This indicates that the function is essential for improving the model's detection power without compromising speed.

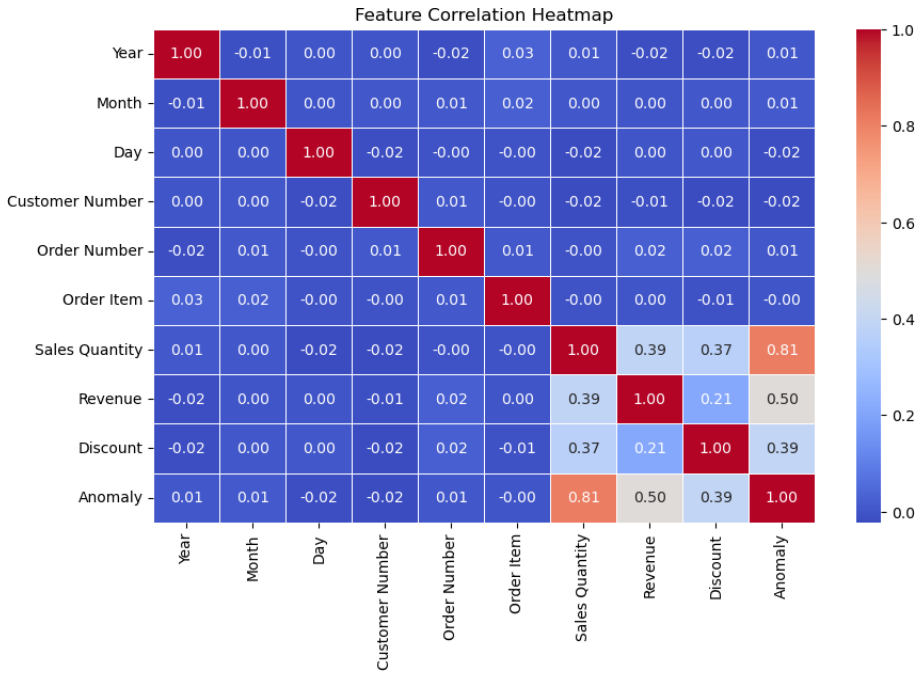

Figure 7: Sales feature correlation heatmap

4.2 API Development and Deployment

The FastAPI application encapsulates the predictive capabilities of the trained anomaly detection model. FastAPI facilitates effortless interaction with the anomaly detection model by providing automatic OpenAPI documentation and asynchronous support. A real-time API endpoint is established to expose the model's anomaly detection capabilities, accept input data, and return predictions.

To assure consistent deployment across varying environments, the FastAPI application is containerized using Docker. Azure Kubernetes Service is employed to deploy the containerized API on Azure, guaranteeing efficient and scalable administration. Security was guaranteed via HTTPS encryption and token-based authentication, while Azure's scalability accommodated high-throughput demand.

The LOF model's predictive capabilities were integrated into a custom API utilizing the FastAPI framework. The API enabled external systems, including the SAP HANA Fiori web application, to transmit real-time transaction data and obtain anomaly detection results in return. The API was deployed on Azure Cloud utilizing Docker containers, guaranteeing scalability and uniform performance across environments. FastAPI's automatic documentation and capacity to manage high-volume API requests rendered it an optimal framework for showcasing the machine learning model's functionalities. Additionally, the use of Azure's scalable cloud infrastructure ensured that the API could handle varying workloads without sacrificing performance. This was critical for ensuring that the system could meet real-time processing requirements in enterprise settings, providing a scalable and accessible platform for real-time anomaly detection.

4.3 SAP HANA Fiori Integration

The SAP HANA Fiori web application is seamlessly integrated with the API, which allows for the real-time detection of anomalies within the user interface. The web application is connected to the API by configuring a service destination in the SAP HANA Cloud Platform Cockpit. To facilitate communication between the web application and the API, an extension to the OData service is developed that specifies input parameters and response structures.

The web application was configured to interact with the anomaly detection (shown in Figure 8). This application consumes the OData service and loads data from the system. The app is configured to show the sample data and make the external API call. The system facilitated smooth communication between the SAP HANA database, which held the transactional data, and the deployed ML model through the API. The configuration involved setting up OData services, defining input parameters, and establishing routes for API calls within the Fiori interface. This integration enabled end-users to interact with the model predictions seamlessly. The SAP HANA Fiori web application was successfully integrated with the anomaly detection API, allowing real-time interaction and decision-making based on the model's predictions Figure 9.

A user-provided service is created on SAP HANA XS Advanced Cockpit and the custom API endpoint credentials are assigned. Afterwards, the OAfterwardcation is assigned to this user-provided service in *mta.yaml* file. The route of service to OData has been used in *outbound.controller.js*. This is defined in the *xs-app.json* file.

The proposed environment offers significant advantages by incorporating FastAPI for API development and scikit-learn's ML models to improve anomaly detection in the SAP HANA Fiori web application. This integrated system utilizes powerful algorithms, offering immediate insights for well-informed decision-making. Moreover, the adaptability of customized APIs effectively overcomes the constraints of SAP HANA's PAL, guaranteeing the streamlined identification of irregularities. This methodology could be applied to other enterprise systems requiring anomaly detection, such as network security or operational monitoring. Despite its bene-

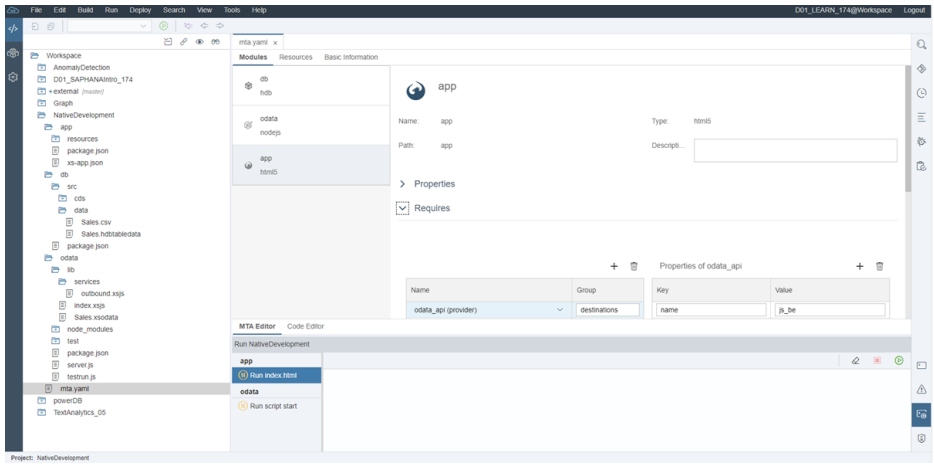

Figure 8: Overview of SAP HANA Fiori web configuration

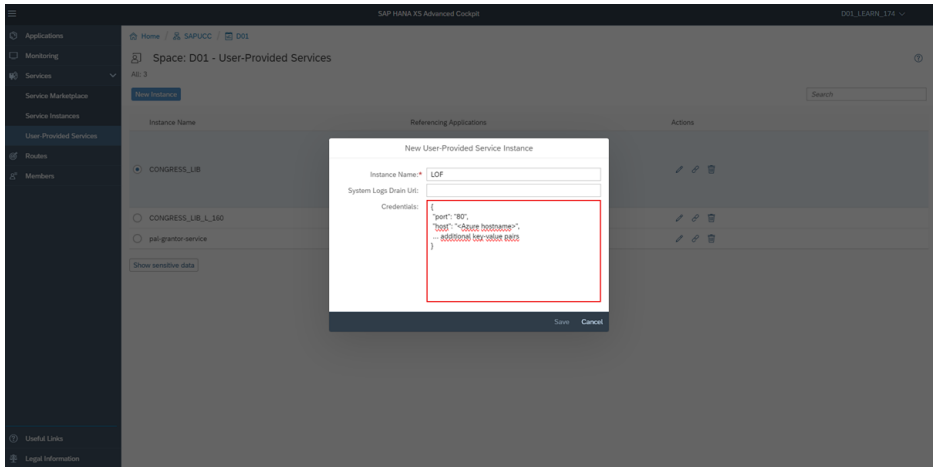

Figure 9: Creating a user-provided service on SAP HANA XS advanced cockpit and assigning the custom API endpoint credentials

fits, the challenges included ensuring seamless integration of the machine learning model, the API, and the SAP HANA Fiori web application. This incurs additional costs and effort for maintenance and the possibility of scaling issues as the volume of data grows. Future research could concentrate on optimizing the model for even larger datasets and automating updating the model as new transaction data becomes available.

5 Conclusions

In this study, an anomaly detection system inside the SAP HANA Fiori Web Tool was successfully implemented. The experimental results indicate that LOF surpasses other anomaly detection methods, including Isolation Forest and One-Class SVM, in terms of both accuracy and processing speed. The model utilized transactional sales data from SAP HANA, confirming its relevance to practical enterprise contexts. The constraints of SAP HANA's PAL were addressed by integrating a custom API-based solution powered by scikit-learn's LOF model, thus creating a more reliable anomaly detection method. Moreover, FastAPI allowed a high-performance API interface, boosting the LOF model's usability. Azure deployment guaranteed scalability and dependability by employing Docker and Kubernetes. This solution easily links scikit-learn, FastAPI, Azure, and SAP HANA Fiori to create a robust and unified predictive analytics system that tackles technical issues and improves functionality, adaptability, and real-time anomaly detection in business environments. Therefore, many technical challenges are addressed using this solution. This approach proved to be effective, but there are still many areas that need improvement. Maintaining the performance of the anomaly detection system as data quantities becomes a challenge. Future studies might look into using anomaly detection algorithms based on evaluating some lightweight neural network alternatives (e.g., shallow autoencoders or one-layer graph networks). Once we have validated that their computational overhead remains compatible with SAP's real-time requirements, which could offer even more accuracy and flexibility. This study particularly contributes to the field of enterprise AI integration by presenting a scalable and efficient solution for real-time anomaly detection in SAP HANA environments. The findings highlight the significance of integrating machine learning, cloud technologies, and API-driven architectures to improve enterprise analytics capabilities.

Data Availability Statement

The dataset utilized in this study was obtained from the SAP HANA platform. While it is not publicly available, access may be granted upon formal request through the appropriate official channels. The source code and supplementary materials can be accessed at the following URL: https://github.com/Georgina-asuah/SAPML.

References

[1] Agyemang, E. F. Anomaly detection using unsupervised machine learning algorithms: A simulation study. *Scientific African*, 26:e02386, 2024. DOI: 10.1016/j.sciaf.2024.e02386.

[2] Albtsoh, L. and Omar, M. Textguard: Identifying and neutralizing adversarial threats in textual data. *International Journal of Informatics, Information System and Computer Engineering (INJIISCOM)*, 6(2):212–224, 2025. URL: https://ojs.unikom.ac.id/index.php/injiiscom/article/view/15232.

[3] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. Software engineering for machine learning: A case study. In *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice*, pages 291–300. IEEE, 2019. DOI: 10.1109/ICSE-SEIP.2019.00042.

[4] Baur, B. *Machine Learning mit SAP HANA*. Espresso Tutorials GmbH, 2022. URL: https://www.vitalsource.com/products/machine-learning-mit-sap-hana-benedict-baur-v9783960121688.

[5] Baylor, D. et al. Tfx: A TensorFlow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1387–1395, 2017. DOI: 10.1145/3097983.3098021.

[6] Birihanu, E. and Lendák, I. Optimal sensor data resampling for anomaly detection in industrial control systems. In *The International Conference on Recent Innovations in Computing*, pages 697–710. Springer, 2023. DOI: 10.1007/978-981-97-3442-9_49.

[7] Buitinck, L. et al. API design for machine learning software: Experiences from the scikit-learn project. arXiv preprint, 2013. DOI: 10.48550/arXiv.1309.0238.

[8] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009. DOI: 10.1145/1541880.1541882.

[9] Easin, A. M. and Orosz, T. Enhancing SAP ecosystem: Harmonizing open-source technologies for integration and innovation. In *Proceedings of the 14th Conference of PhD Students in Computer Science*, page 7, 2024. URL: https://www.inf.u-szeged.hu/~cscs/pdf/cscs2024.pdf.

[10] Easin Arafat, M., Asuah, G., Saha, S., and Orosz, T. Empowering real-time insights through LLM, LangChain, and SAP HANA integration. In *Proceedings of the International Conference on Recent Innovations in Computing*, pages 483–495. Springer, 2023. DOI: 10.1007/978-981-97-3442-9_33.

[11] García, Á. L. et al. A cloud-based framework for machine learning workloads and applications. *IEEE Access*, 8:18681–18692, 2020. DOI: 10.1109/ACCESS.2020.2964386.

[12] Gole, V. and Shiralkar, S. *Empower decision makers with SAP analytics cloud*. Springer, 2020. DOI: 10.1007/978-1-4842-6097-5.

[13] Jin, J. Anomaly detection and exploratory causal analysis for SAP HANA. Master's thesis, Karlsruher Institut für Technologie, 2019. DOI: 10.5445/IR/1000089289.

[14] Khasawneh, M. A., Daraghmeh, M., Awasthi, A., and Agarwal, A. Multilevel learning for enhanced traffic congestion prediction using anomaly detection and ensemble learning. *Cluster Computing*, 28(3):160, 2025. DOI: 10.1007/s10586-024-04871-z.

[15] Kohli, M. Using machine learning algorithms on data residing in SAP ERP application to predict equipment failures. *International Journal of Engineering & Technology*, 7(2.28):312–319, 2017. DOI: 10.14419/ijet.v7i2.28.12952.

[16] Lavin, A. et al. Technology readiness levels for machine learning systems. *Nature Communications*, 13(1):6039, 2022. DOI: 10.1038/s41467-022-33128-9.

[17] Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008. DOI: 10.1109/ICDM.2008.17.

[18] Mahmud, J. S., Birihanu, E., and Lendak, I. A semi-supervised framework for anomaly detection and data labeling for industrial control systems. In *Proceedings of the Conference on Information Technology and its Applications*, Volume 872 of *Lecture Notes in Networks and Systems*, pages 149–160. Springer, 2023. DOI: 10.1007/978-3-031-50755-7_15.

[19] Mahmud, J. S. and Lendak, I. Enhancing one-class anomaly detection in unlabeled datasets through unsupervised data refinement. In *Proceedings of the 2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics*, pages 000497–000502. IEEE, 2024. DOI: 10.1109/SISY62279.2024.10737577.

[20] Mogîldea, A. SAP HANA – SAP high-performance analytic appliance. Tehnica UTM, 2016. URL: http://repository.utm.md/handle/5014/808.

[21] Nandigramwar, H., Mittal, A., Bhatnagar, A., and Rashid, M. A distributed and unified API service for machine learning models. In *Proceedings of the 2021 2nd International Conference on Intelligent Engineering and Management*, pages 480–485. IEEE, 2021. DOI: 10.1109/ICIEM51511.2021.9445348.

[22] Oliveira, J. P. and Sousa, R. D. Unsupervised anomaly detection of retail stores using predictive analysis library on SAP HANA XS Advanced. *Procedia Computer Science*, 181:882–889, 2021. DOI: 10.1016/j.procs.2021.01.243.

[23] Peksa, J. Autonomous data-driven integration into ERP systems. In *Design, Simulation, Manufacturing: The Innovation Exchange*, pages 223–232. Springer, 2021. DOI: 10.1007/978-3-030-77719-7_23.

[24] Reimann, L. and Kniesel-Wünsche, G. Improving the learnability of machine learning APIs by semi-automated API wrapping. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, pages 46–50, 2022. DOI: 10.1145/3510455.3512789.

[25] Sakurada, M. and Yairi, T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014. DOI: 10.1145/2689746.2689747.

[26] SAP HANA Machine Learning Overview, 2021. URL: https://help.sap.com/doc/ae101ea615324a41addb8b9552805f20/2.0.05/en-US/SAP_HANA_Machine_Learning_Overview_Guide_en.pdf.

[27] Sarferaz, S. *Embedding Artificial Intelligence into ERP Software*. Springer, 2024. DOI: 10.1007/978-3-031-54249-7.

[28] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001. DOI: 10.1162/089976601750264965.

[29] Settibathini, V. S. Enhancing user experience in SAP Fiori for finance: A usability and efficiency study. *International Journal of Machine Learning for Sustainable Development*, 5(3):1–13, 2023. URL: https://ijsdcs.com/index.php/IJMLSD/article/view/467.

[30] Shaik, M. and Siddque, K. Q. Predictive analytics in supply chain management using SAP and AI. *Journal of Computer Sciences and Applications*, 11(1):1–6, 2023. DOI: 10.12691/jcsa-11-1-1.

[31] Shi, Z. and Wang, G. Integration of big-data ERP and business analytics (BA). *The Journal of High Technology Management Research*, 29(2):141–150, 2018. DOI: 10.1016/j.hitech.2018.09.004.

[32] Sivakumar, V., Prasad, M. R., Vadivel, M., Prasad, S. T., Aranganathan, A., and Murugan, S. Isolation forests integration for proactive anomaly detection in Augmented Reality-enhanced Tele-ICU systems. In *Proceedings of the 2024 6th International Conference on Energy, Power and Environment*, pages 1–6. IEEE, 2024. DOI: 10.1109/ICEPE63236.2024.10668946.

[33] Smith, D., Khorsandroo, S., and Roy, K. Machine learning algorithms and frameworks in ransomware detection. *IEEE Access*, 10:117597–117610, 2022. DOI: 10.1109/ACCESS.2022.3218779.

[34] Subramanian, H. and Raj, P. *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Packt Publishing Ltd, 2019. ISBN: 9781788998581.

[35] Yang, K. *Quality in the Era of Industry 4.0: Integrating Tradition and Innovation in the Age of Data and AI.* John Wiley & Sons, 2024. ISBN: 9781119932468.