

# ANOMALY DETECTION IN ALTAIR® AI STUDIO™: A COMPARATIVE APPROACH USING DBSCAN CLUSTERING AND ISOLATION FOREST (PYOD)

Khushi Kiran Sheelavantmath, Intern-Data Science, Analytics & IoT Support / Sreejay Sreedharan, Director – DA for BFSI, Analytics & IoT Support / July 24, 2025



## Introduction

The global banking industry is undergoing a significant digital transformation fueled by the widespread adoption of online and mobile banking services. While this evolution offers increased convenience and accessibility for consumers, it also exposes financial institutions to growing risks such as fraudulent transactions, irregular account activity, and compliance violations. Traditional rule-based systems and batch-processing approaches, which rely on predefined thresholds and historical fraud patterns, are increasingly inadequate in detecting subtle or previously unseen anomalies, especially in real time.

To address these challenges, we embarked on a project to identify hidden patterns, anomalies in financial transaction data, and practical applications of machine learning models within a low-code environment. This project employed unsupervised machine learning techniques—specifically density-based spatial clustering of applications with noise (DBSCAN) and Isolation Forest from the Python outlier detection (PyOD) library.

The entire anomaly detection pipeline is built using [Altair® AI Studio™](#), a low-code, visual platform that enables seamless development of machine learning workflows, ranging from data ingestion and feature engineering to model training and results visualization. With minimal coding, Altair AI Studio allows the creation of scalable, interpretable, and adaptive fraud detection systems. By combining the power of unsupervised learning with an intuitive platform, this project provided an effective solution for modern financial institutions seeking to proactively detect and mitigate emerging risks in digital banking.

## Dataset Description

A high-velocity transactional dataset was used and comprised 2,512 samples and 16 features (Figure 1), capturing a diverse range of user and transaction-related metrics. Key attributes include account ID, transaction amount, device ID, location, IP address, customer demographics, and timestamped activities. This rich combination of features made the dataset well-suited for detecting anomalies related to location shifts, device spoofing, and sudden behavioral changes — all common indicators of financial fraud and unauthorized account access.

Row No.	TransactionID	AccountID	Transaction...	Transaction...	Transaction...	Location	DeviceID	IP Address	MerchantID	Channel	CustomerAge	CustomerO...	Transaction...
1	TX000001	AC00128	14,060	Apr 11, 2023 ...	Debit	San Diego	D000380	162.198.216.92	M015	ATM	70	Doctor	81
2	TX000002	AC00455	376,249	Jun 27, 2023 ...	Debit	Houston	D000051	13.149.61.4	M052	ATM	86	Doctor	141
3	TX000003	AC00019	128,290	Jul 10, 2023 ...	Debit	Mesa	D000235	215.97.143.157	M009	Online	19	Student	56
4	TX000004	AC00070	184,500	May 5, 2023 ...	Debit	Raleigh	D000167	200.13.225.150	M002	Online	26	Student	25
5	TX000005	AC00411	13,450	Oct 16, 2023 ...	Credit	Atlanta	D000308	65.194.3.100	M091	Online	26	Student	198
6	TX000006	AC00383	92,150	Apr 3, 2023 ...	Debit	Oklahoma City	D000579	117.67.182.211	M054	ATM	18	Student	172
7	TX000007	AC00199	7,080	Feb 15, 2023 ...	Credit	Seattle	D000241	140.212.253...	M019	ATM	37	Doctor	139
8	TX000008	AC00089	171,420	May 6, 2023 ...	Credit	Indianapolis	D000500	92.214.76.157	M020	Branch	87	Retired	291
9	TX000009	AC00135	106,230	Mar 21, 2023 ...	Credit	Detroit	D000690	24.148.92.177	M035	Branch	51	Engineer	86
10	TX000010	AC00385	815,660	Mar 31, 2023 ...	Debit	Nashville	D000169	32.188.88.41	M007	ATM	55	Doctor	120
11	TX000011	AC00150	17,780	Mar 14, 2023 ...	Credit	Albuquerque	D000205	213.15.9.253	M073	Online	52	Engineer	59
12	TX000012	AC00459	190,020	Feb 6, 2023 ...	Debit	Memphis	D000589	116.175.11.222	M030	Online	21	Student	173
13	TX000013	AC00392	494,520	Jun 7, 2023 ...	Credit	Mesa	D000032	210.96.198.143	M057	Branch	24	Student	111
14	TX000014	AC00264	781,760	Nov 20, 2023 ...	Debit	Memphis	D000054	189.83.0.183	M025	ATM	26	Student	123
15	TX000015	AC00085	168,990	Feb 13, 2023 ...	Debit	Louisville	D000309	186.124.181.12	M017	Online	16	Student	134
16	TX000016	AC00270	465,450	Dec 12, 2023 ...	Debit	Denver	D000466	221.166.48.152	M025	ATM	36	Engineer	129

Figure 1 – Dataset description: sample data

## Data Preparation

To enhance the dataset's ability to capture subtle, behavior-based anomalies, a lag attribute was engineered to track changes in key device and location features—specifically, device ID, location, and IP address—across consecutive transactions belonging to the same account ID. This was implemented using a loop operator (Figure 2) that sequentially compared each transaction with the immediately preceding one for the same account. If any of these attributes showed a change from the previous transaction, the lag attribute was flagged with a value of 1, indicating a potential anomaly; otherwise, it was set to 0.

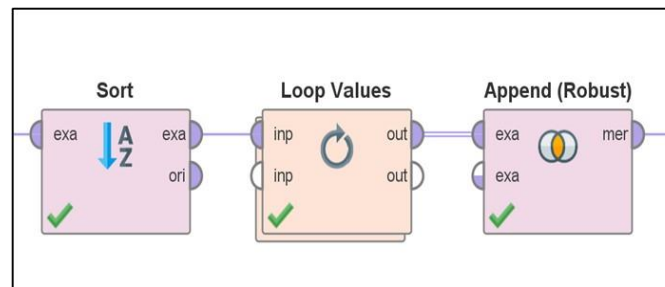


Figure 2 – Dataset preparation: Altair AI studio workflow

This lag-based feature (Figure 3) is especially important because fraudsters often attempt to disguise their activity by accessing accounts from new devices, unusual locations, or through spoofed IP addresses. By capturing these temporal shifts at the individual account level, the model gains insight into context-sensitive anomalies—those that are not evident when transactions are viewed in isolation but become suspicious when considered in sequence. For example, a sudden change in device or location following a history of stable usage patterns may signal account takeover or fraudulent access (Figure 4).

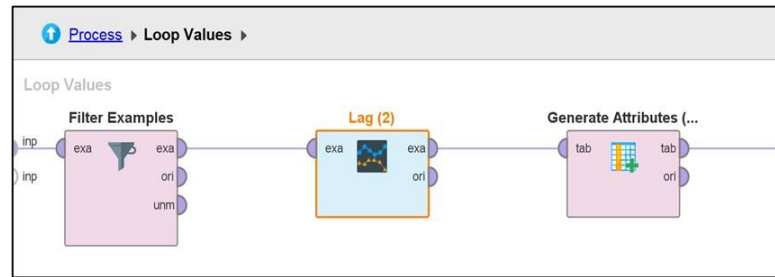


Figure 3 – Dataset preparation: Altair AI studio workflow inside loop values

individual attribute	lag
IP Address	1
DeviceID	1
Location	1

column name	function expressions
device_change	if([DeviceID-1] != DeviceID, 1, 0)
ip_change	if([IP Address-1] != [IP Address], 1, 0)
location_change	if([Location-1] != Location, 1, 0)

Figure 4 – Dataset preparation: Altair AI Studio lag operator and generate attributes operator parameters respectively

Incorporating this temporal dimension helps the anomaly detection algorithms distinguish between legitimate user behavior and suspicious activity more effectively, leading to improved detection accuracy. It also allows for a more nuanced understanding of how fraud manifests over time, rather than relying solely on static transactional features.

## Data Preprocessing and Feature Selection

Effective anomaly detection relies heavily on data preprocessing (Figure 5), using the most relevant features that capture meaningful patterns in the data (Figure 6). To achieve this, we first performed correlation analysis (Figure 5) to measure the relationships between individual features and potential anomaly indicators. This statistical method helped identify which attributes—such as transaction amounts, device changes, location shifts, and temporal activity—were most strongly associated with unusual behaviors in the dataset. By focusing on these significant features, we minimized noise from irrelevant or redundant variables that could otherwise dilute the model's ability to detect anomalies accurately.

Attributes	TransactionAmount	CustomerAge	TransactionDuration	LoginAttempts	AccountBalance
TransactionAmount	1	-0.026	0.004	-0.008	-0.025
CustomerAge	-0.026	1	-0.018	0.008	0.320
TransactionDuration	0.004	-0.018	1	0.033	0.006
LoginAttempts	-0.008	0.008	0.033	1	0.015
AccountBalance	-0.025	0.320	0.006	0.015	1

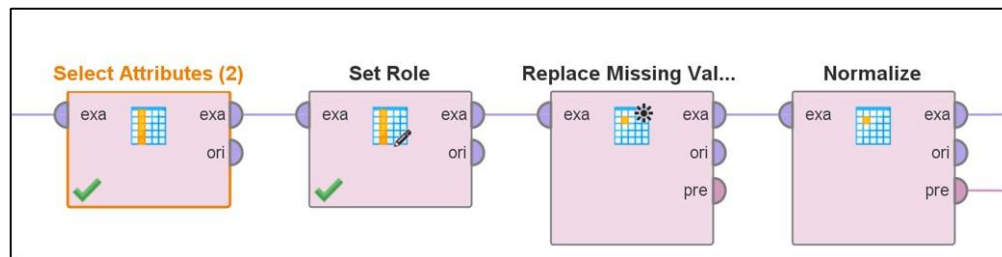


Figure 5 – Dataset preprocessing and feature selection: correlation matrix and Altair AI Studio workflow

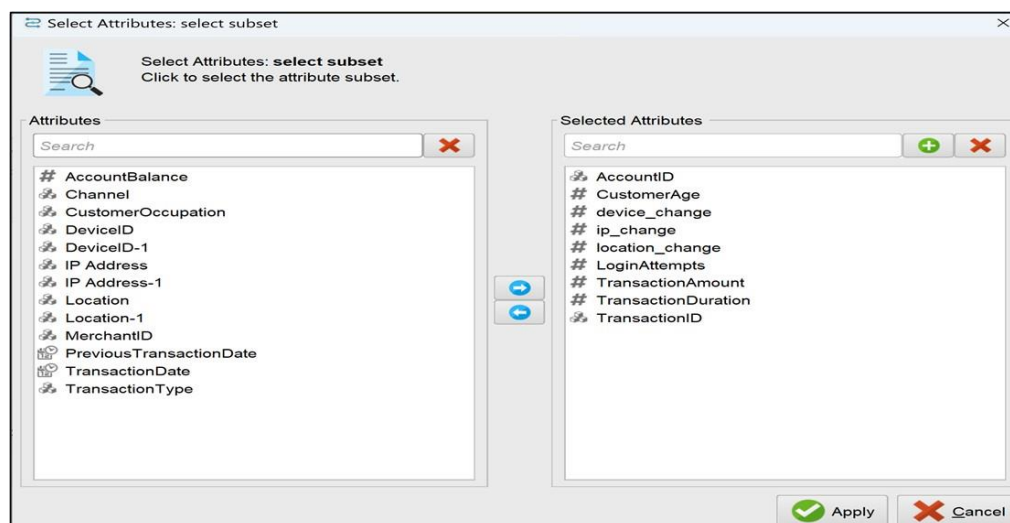


Figure 6 – Dataset preprocessing and feature selection: selected attributes

After selecting the key features, we applied normalization using the interquartile range (IQR) method. This technique standardizes feature values by scaling them relative to their spread between the 25th and 75th percentiles, making the data less sensitive to extreme values or outliers. Unlike simple min-max scaling or z-score normalization, the IQR method is robust to outliers, which are common in transactional datasets and could skew model training if not addressed properly (Figure 7).

By combining correlation-based feature selection with IQR normalization, we ensured that the anomaly detection models—DBSCAN and Isolation Forest—operate on a focused, consistent, and clean dataset. This preprocessing step improved model convergence, enhanced the detection of subtle anomalies, and contributed to more reliable and interpretable results (Figure 5).

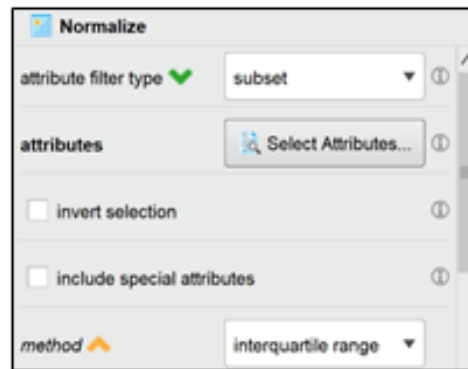


Figure 7 – Dataset preprocessing and feature selection: normalize operator parameters

### Anomaly Detection Using DBSCAN Clustering

DBSCAN is a density-based clustering algorithm used to identify clusters in data based on regions of high point concentration. Unlike partitioning algorithms, such as K-Means, DBSCAN does not assume any specific cluster shape and does not require the number of clusters to be specified beforehand. Instead, it uses a local density criterion to determine whether a point belongs to a cluster or should be classified as noise.

The algorithm operates by identifying core points with enough neighboring points within a defined radius. Clusters are formed by connecting core points that are density-reachable, meaning one can be reached from the other through a chain of neighboring core points. This process naturally expands clusters around dense areas in the dataset, while points lying alone in sparse regions are labeled as outliers.

DBSCAN's flexibility in handling arbitrary cluster shapes and ability to automatically detect noise make it suitable for analyzing datasets where traditional methods struggle. The algorithm's behavior is governed by two key parameters that control the notion of density: the neighborhood radius and the minimum number of points required to form a dense region. Despite its strengths, DBSCAN may face challenges in datasets with varying densities, where a single global parameter setting may not capture all cluster structures accurately. Nonetheless, it remains a powerful and intuitive tool for density-based clustering and anomaly detection

### Mathematical Foundation

DBSCAN requires two parameters:

- $\epsilon$  (epsilon): The radius within which points are considered neighbors
- MinPts: The minimum number of points required to form a dense region

Given a dataset  $D = \{x_1, x_2, \dots, x_n\}$ , for a point  $x_i$ :

The  $\epsilon$ -neighborhood of  $x_i$  is defined as all points  $x_j$  in  $D$  such that the distance between  $x_i$  and  $x_j$  is less than or equal to  $\epsilon$ .

$$\rightarrow N_\epsilon(x_i) = \{x_j \in D \mid \text{dist}(x_i, x_j) \leq \epsilon\}$$

- If the number of points in  $N_\epsilon(x_i)$  is greater than or equal to MinPts, then  $x_i$  is labelled as a core point
- Points that are within  $\epsilon$  distance of a core point are considered part of the same cluster
- Points that are not reachable from any core point are treated as noise or anomalies

### Implementation in Altair AI Studio

DBSCAN is an unsupervised learning algorithm well-suited for identifying patterns and outliers in transactional datasets. Unlike K-Means, DBSCAN does not require predefining the number of clusters and can detect arbitrarily shaped clusters. It works by evaluating density around data points using two parameters: epsilon ( $\epsilon$ ), the neighbourhood radius, and MinPoints, the minimum number of points to form a cluster. Points in low-density areas are labelled as noise or anomalies, which makes DBSCAN ideal for detecting irregularities in financial transaction data (Figure 8).

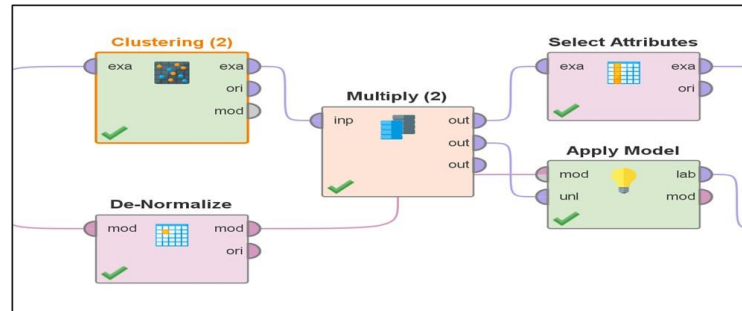
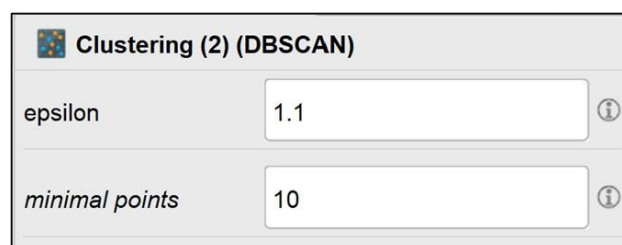


Figure 8 – Anomaly detection using DBSCAN clustering: Altair AI Studio workflow

In this project, DBSCAN was applied using Altair AI Studio, with epsilon set to 1.1 and the minimum points to 10 (Figure 9), based on empirical tuning. These parameters allowed the algorithm to capture dense clusters of regular transaction behavior while identifying sparse or isolated points as potential anomalies. After clustering, the data was de-normalized to restore it to its original scale for ease of interpretation. This end-to-end pipeline enabled a robust clustering mechanism that uncovered hidden patterns and deviations in the dataset, providing valuable insights into anomalous transactional activity without relying on labelled data.



The screenshot shows the configuration window for the 'Clustering (2) (DBSCAN)' operator. It includes two input fields:

- epsilon**: Set to 1.1
- minimal points**: Set to 10

Each field has an information icon (i) to the right.

Figure 9 – Anomaly detection using DBSCAN clustering: DBSCAN operator parameters

We were able to analyze the data as to which was marked as regular transaction and potential fraud (Figure 10), and a scatter plot was created using customer age and transaction amount to visualize the clustering results and identify potential anomalies (Figure 11). This plot effectively illustrates the separation between regular transactions and outliers detected by the DBSCAN algorithm. While most transactions formed dense clusters indicative of typical behavior, the anomalies appeared as isolated points scattered away from the main grouping. This visual representation helped validate the clustering model qualitatively, offering an intuitive understanding of how certain transactions deviated from normal customer patterns and potentially signaled fraudulent activity.





Figure 10 - Anomaly detection using DBSCAN clustering: anomaly found after clustering



Figure 11 - Anomaly detection using DBSCAN clustering: scatter plot visualization in AI Studio and Tooltip in Python platy

After applying DBSCAN clustering, the results were further analyzed by aggregating key transaction-related attributes across the identified anomaly groups (Figure 12). This post-processing step revealed distinct behavioral differences between regular and anomalous transactions. Fraudulent or irregular entries often exhibited elevated login attempts, higher average transaction amounts, and longer transaction duration. These patterns reflected unusual activity typically associated with suspicious behavior. Aggregating features in this manner added clarity to the model output (Figure 13), helping to validate and interpret the clusters by highlighting meaningful deviations from typical customer behavior. The model performance was evaluated using silhouette score (Figure 14).

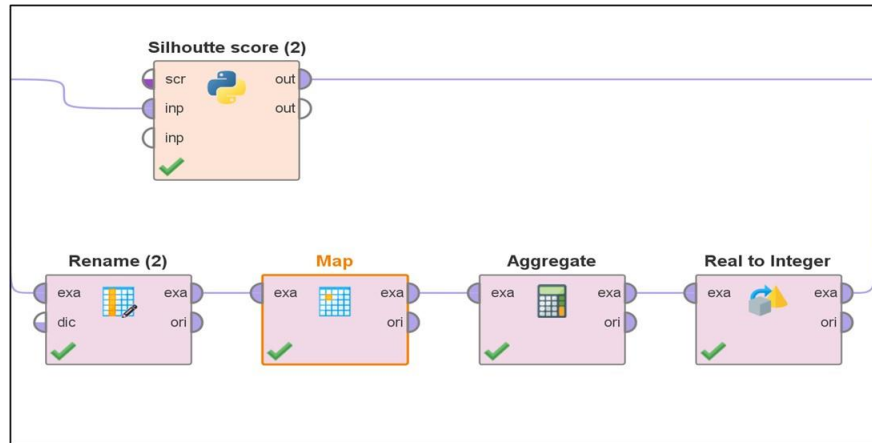


Figure 12 - Anomaly detection using DBSCAN clustering: Altair AI Studio workflow for output

Row No.	Anomaly Found	median(LoginAttempts)	average(TransactionAmount)	average(TransactionDuration)	mode(CustomerAge)
1	Potential Fraud	4	583.511	139.307	24.000
2	Regular Transactions	1	284.001	118.709	26.000

Figure 13 - Anomaly detection using DBSCAN clustering: aggregation results

Row No.	Silhouette_Score
1	0.470

Figure 14 - Anomaly detection using DBSCAN clustering: silhouette score



### Advantages

- Detects clusters of arbitrary shapes
- Robust to outliers
- No requirement to specify the number of clusters

### Use Cases

- Banking and finance: DBSCAN has been used to detect unusual transaction behaviors and financial fraud by identifying transaction clusters and separating outliers
- Healthcare: Applied to identify anomalous patient records in electronic health systems that may signal misdiagnosis or system errors
- Mobility/transportation: Used for analyzing GPS and ride-sharing data to detect outlier travel routes or unauthorized fleet usage

### Anomaly Detection Using Isolation Forest (PyOD)

Isolation Forest is an unsupervised anomaly detection algorithm that operates by isolating data points rather than modeling normal behavior. The fundamental assumption behind the method is that anomalies are few and significantly different from majority of data, making them easier to isolate through random partitioning. Unlike traditional distance- or density-based techniques, Isolation Forest builds an ensemble of binary trees, known as isolation trees, where each tree is constructed by randomly selecting a feature and then randomly choosing a split value within the range of that feature. As this recursive process continues, data points are progressively partitioned until they are isolated. Anomalous points, which reside in sparse regions of the data space, typically require fewer splits to be isolated and thus tend to have shorter path lengths in the tree structure.

This approach provides a natural and intuitive anomaly score based on the average path length required to isolate a point across all trees in the ensemble. The shorter the average path, the more likely the point is an anomaly. Isolation Forest is computationally efficient, with linear time complexity in both the number of data points and the number of features. It also requires significantly less memory compared to other methods, making it suitable for high-dimensional and large-scale datasets. Furthermore, the interpretability of the algorithm lies in its simplicity. Since anomalies are defined in terms of ease of isolation, the model avoids assumptions about the data distribution and adapts well to different types of datasets. This makes Isolation Forest a robust and versatile technique for identifying rare or irregular patterns in complex data.

### Mathematical Foundation

Isolation Forest uses random partitioning to isolate observations:

- Given a dataset  $D$ , randomly select a feature  $f$  and a split value  $s \in [f_{\min}, f_{\max}]$
- Recursively split the data
- Path length  $h(x)$  for observation  $x$  is the number of splits to isolate it
- Average path length  $c(n)$  is computed as:
- $c(n) = 2H(n-1) - 2(n-1)/n$ ,
- where  $H(i) = \ln(i) + 0.5772$  (Euler's constant)
- Anomaly score:  $s(x, n) = 2^{-(E(h(x))/c(n))}$  where  $E(h(x))$  is the average of  $h(x)$  from a collection of isolation trees

## Implementation in Altair AI Studio

An Isolation Forest model was applied for anomaly detection using the PyOD library, executed via the execute Python operator in Altair AI Studio. This operator allowed the integration of custom Python scripts directly into the workflow, making it possible to implement advanced anomaly detection without switching platforms. Isolation Forest is an unsupervised learning algorithm that isolates anomalies by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Since anomalies are fewer and more different, they tend to be isolated in fewer steps than normal data points. This property makes Isolation Forest particularly effective in identifying irregular patterns in high-dimensional datasets like financial transactions. The model assigned an anomaly score to each record, which was then used to label transactions as either normal or suspicious. This helped uncover nuanced deviations that could indicate potential fraud, behavioral shifts, or system misuse, all while maintaining a streamlined and code-efficient pipeline (Figure 15).

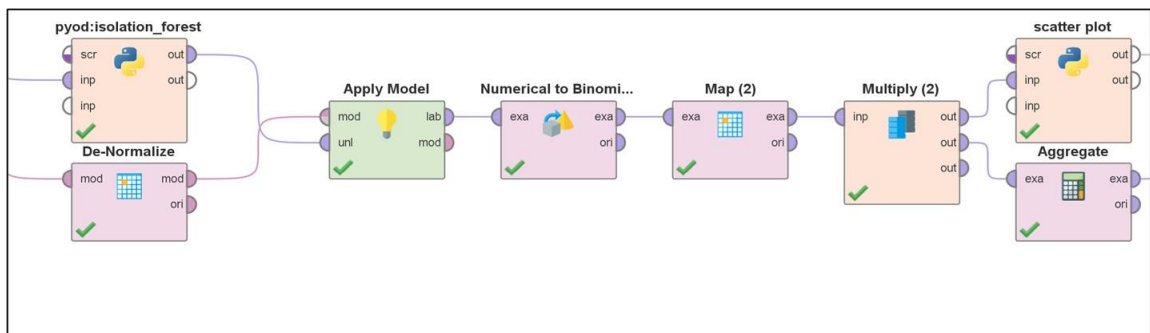


Figure 15 - Anomaly detection using Isolation Forest (PyOD): Altair AI Studio workflow

We were able to analyze the data as to which was marked as regular transaction and potential fraud (Figure 16), and a scatter plot was created using customer age and transaction amount to visualize the anomaly scores produced by the Isolation Forest model (Figure 17). The plot revealed a clear distinction between normal transactions and those flagged as anomalous by the model. Most data points formed dense regions, reflecting common transactional behavior, whereas potential outliers appeared as isolated points away from the core clusters. This visualization not only provided an intuitive grasp of how the model identified anomalies but also offered a valuable cross-check to assess the alignment of model outputs with observable behavioral deviations. It reinforced the effectiveness of Isolation Forest in detecting subtle yet significant irregularities within the transaction dataset.



Figure 16 - Anomaly detection using Isolation Forest (PyOD): anomaly found after clustering

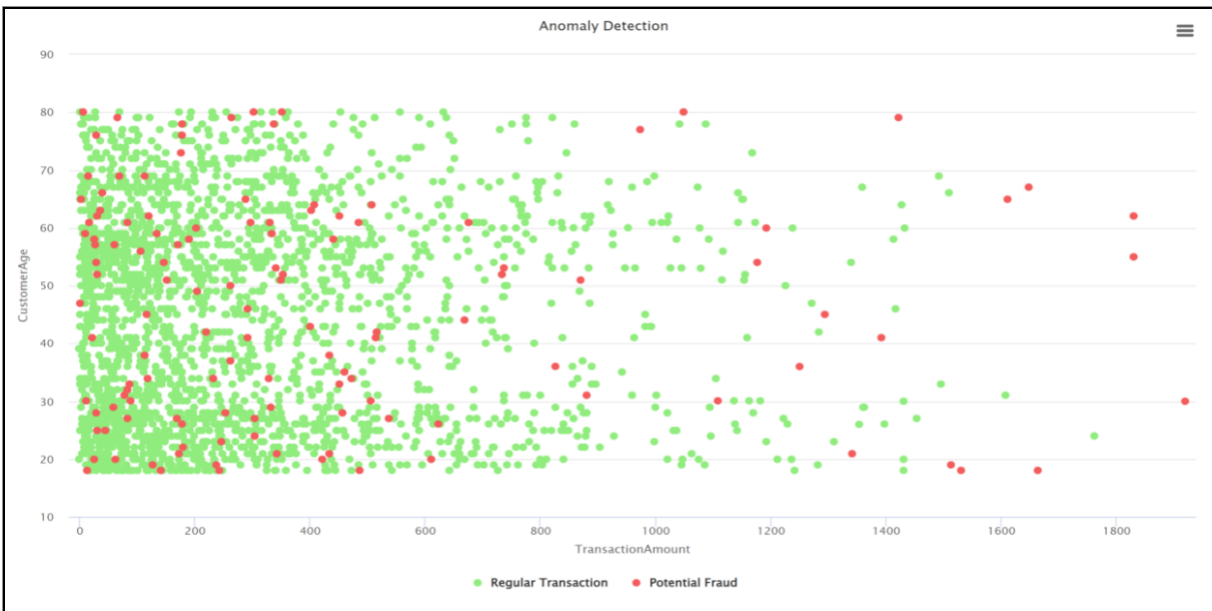


Figure 17 - Anomaly detection using Isolation Forest (PyOD): scatter plot visualization in AI Studio and Tooltip in Python plotly

A similar aggregation process to DBSCAN clustering was conducted on the output of the Isolation Forest model (Figure 18), to interpret its anomaly predictions. The grouped data revealed that transactions flagged as anomalies tended to share characteristics such as increased access attempts, variations in transaction timing, and unusual spending patterns. This method of analysis proved valuable in understanding the logic behind the model's isolation decisions, especially since Isolation Forest identifies outliers based on how easily they can be separated from the rest of the dataset. Overall, this aggregation enriched the interpretability of the model and provided actionable insights into potential fraud risks (Figure 19).

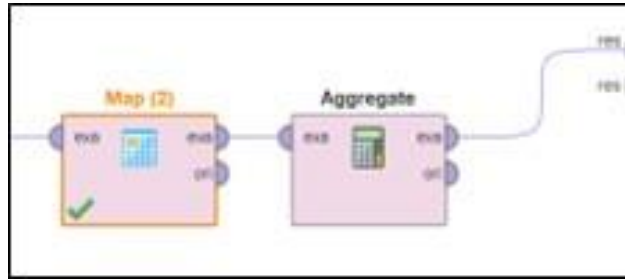


Figure 18 - Anomaly detection using Isolation Forest (PyOD): Altair AI Studio workflow for output

Row No.	Anomaly_found	average(TransactionAmount)	average(TransactionDuration)	median(LoginAttempts)	mode(CustomerAge)
1	Potential Fraud	328.785	137.016	3	18.000
2	Regular Transaction	295.947	118.726	1	26.000

Figure 19 - Anomaly detection using Isolation Forest (PyOD): aggregation results

### Advantages

- Handles high-dimensional data
- Effective for skewed distributions
- Efficient computation

### Use Cases

- Cybersecurity: Used to detect insider threats and anomalous user access behaviour within enterprise networks
- E-Commerce: Applied to identify fraudulent browsing behaviour and fake reviews based on session and clickstream data
- Manufacturing and IoT: Deployed in predictive maintenance to isolate abnormal sensor readings that indicate early signs of machine failure

## Architecture

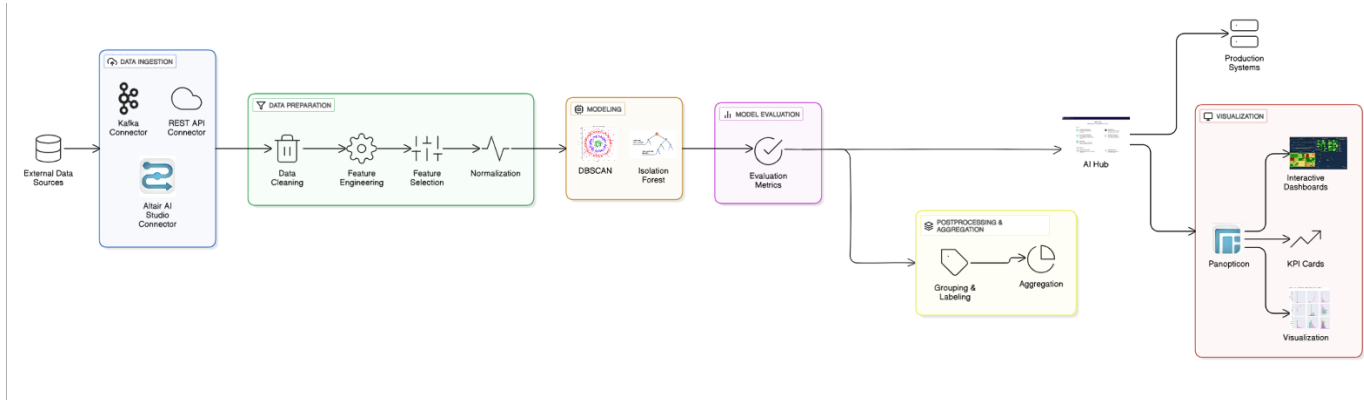


Figure 20 – Architecture Diagram

## Operationalization

The anomaly detection workflow was deployed using [Altair® AI Hub™](#) (Figure 21), enabling seamless integration of real-time processing and analytics. Altair AI Hub acts as the operational backbone for Altair AI Studio workflows by exposing models as scalable, secure REST endpoints that can be triggered in response to live transaction data. Its core benefits include:

- Real-time model deployment with secure REST API endpoints for streaming or batch inference
- Centralized model and workflow lifecycle management, including version control and rollback capabilities
- User and access governance through role-based permissions and audit trails
- CI/CD integration with DevOps pipelines for streamlined deployment

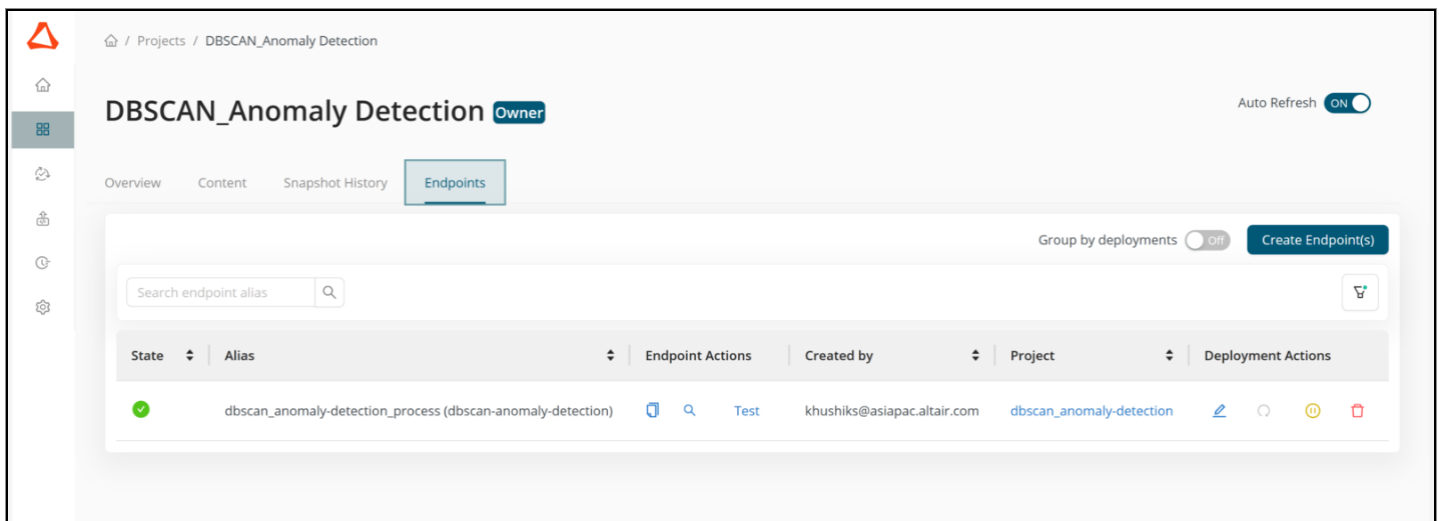


Figure 21 - Architecture: deployment of DBSCAN process on Altair AI Hub

## Challenges

During the design of the anomaly detection pipeline using Altair AI Studio, a few challenges were encountered, particularly in the areas of modeling and post-processing

A primary challenge involved the trial-and-error tuning of DBSCAN parameters, such as epsilon and minimum samples. Given DBSCAN's sensitivity to data scale and density, identifying optimal values required extensive experimentation. Similarly, configuring the Isolation Forest model, particularly determining the right contamination threshold, was complex due to the lack of supervision and variability in transaction patterns.

Another major hurdle was the absence of labeled data, which made model evaluation difficult. Without a clear ground truth, performance had to be assessed through visual validation and domain intuition, which introduced subjectivity into the process.

Additionally, the post-processing and aggregation stage, where detected anomalies were grouped and labelled required custom rule creation based on behavioral patterns and data observations, which was not always generalizable across datasets.

## Conclusion

This project demonstrated the practical application of machine learning models within a low-code environment like Altair AI Studio to address real-world challenges in financial anomaly detection. By leveraging the strengths of both density-based (DBSCAN) and isolation-based (Isolation Forest via PyOD) algorithms, the approach effectively identifies patterns that deviate from normal transaction behavior, many of which are indicative of fraudulent activity. The seamless integration of preprocessing, feature engineering, clustering, and model interpretation in a single pipeline underscores the potential of low-code platforms to accelerate the development of intelligent systems. As digital transactions continue to grow in volume and complexity, AI-powered and scalable solutions are crucial in fortifying the security and integrity of financial operations.

## References

- [1] Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*.
- [2] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*.
- [3] Sun, L., Leung, H., Zhang, H., & Zheng, V. (2016). *Detecting Anomalous User Behaviour Using an Extended Isolation Forest Algorithm: An Enterprise Case Study*. arXiv:1609.06676.
- [4] Sakurada, M., & Yairi, T. (2014). Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*.
- [5] Zhang, K., Song, H., He, J., & Kou, Y. (2021). Fault Detection for Predictive Maintenance Using Isolation Forest in IoT. *IEEE Access*, 9, 89664–89672.
- [6] Hassani, H., Silva, E. S., & Unger, S. (2019). Healthcare analytics in anomaly detection: A case study using DBSCAN. *Health Information Science and Systems*, 7(1), 1–8.
- [7] Zhao, W., Yu, C., & Jin, R. (2020). Detecting Mobility Anomalies in GPS Data using DBSCAN. *Journal of Intelligent Transportation Systems*, 24(3), 248–258.
- [8] Altair Engineering Inc. (2025). *Altair AI Hub™ Product Documentation*. Retrieved from <https://www.altair.com/ai-hub/>