Smart Spending Watchdogs: Using Machine Learning to Detect Anomalies in Financial Transactions

Introduction – Why Outliers Matter More Than You Think

Every second, financial systems around the world process countless transactions — from lunch purchases and rent payments to international transfers and investments. Hidden among them may be a handful of outliers: unusual transactions that deviate from expected behavior. Some are innocent; others, potentially fraudulent.

In a world increasingly dependent on digital finance, the ability to identify anomalies in transaction data is more than just a technical challenge — it's a matter of economic security. Financial fraud can cripple individuals and institutions alike. As global financial flows expand, so does the need for automated, intelligent systems that can separate the signal from the noise.

This blog dives into how machine learning can detect anomalies in financial transactions and, in doing so, supports **SDG 16** (Peace, Justice, and Strong Institutions) by enhancing transparency, accountability, and integrity in financial systems.

Background – What's the Problem?

Manual review of transactions is not scalable. Traditional fraud detection relies heavily on rules (e.g., flag transactions over a certain amount), but fraud patterns constantly evolve. This leads to too many false positives or, worse, undetected fraud.

With millions of transactions happening in real time, we need systems that learn normal behavior and detect subtle irregularities — even those that haven't been seen before. This is where machine learning steps in.

Purpose – What Did We Aim to Do?

This project aimed to explore the use of both statistical and machine learning methods to identify anomalies in transaction data. We used unsupervised learning techniques to automate the detection of abnormal transaction patterns. Our goal was to create a system that could flag potentially fraudulent or erroneous data points without the need for labeled examples.

Methodology – The Process in Detail

Dataset Overview

We worked with a dataset containing anonymized transaction details, including:

- Transaction Amount
- Account Type
- Day of the Week
- Customer Age

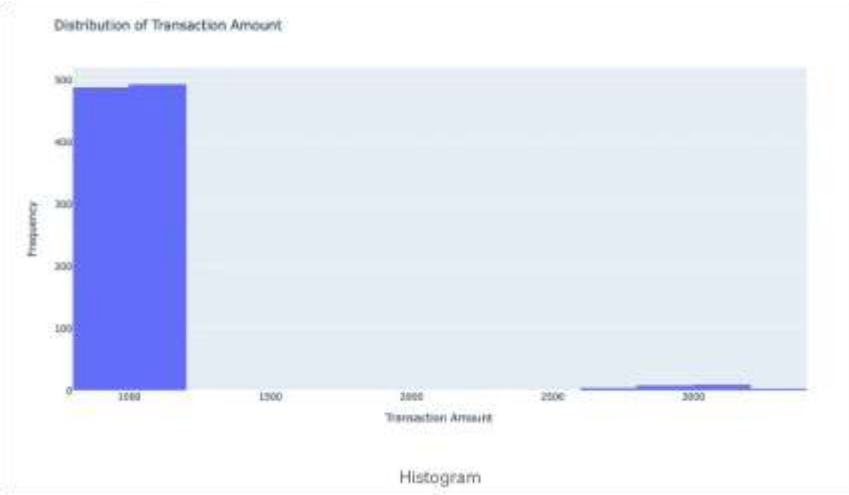
The data was preprocessed to remove nulls, encode categorical variables, and normalize numerical features.

Exploratory Data Analysis (EDA)

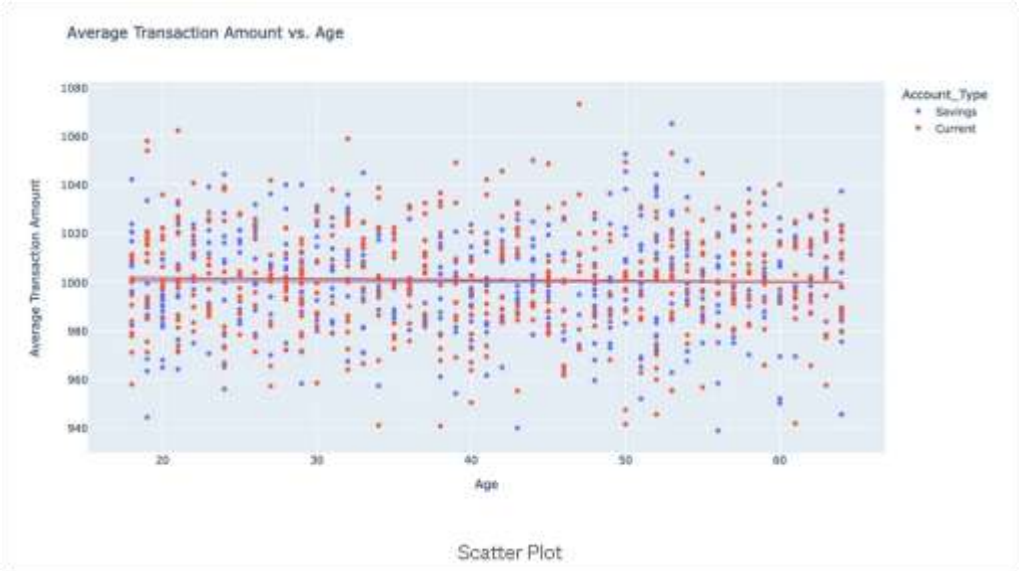
Before modeling, we explored the data to identify patterns:

- **Histograms** revealed skewed transaction amount distributions.
- **Boxplots** showed higher variance in business accounts.
- **Heatmaps** exposed low correlation between customer age and transaction volume — suggesting age wasn't a strong indicator on its own.

Visualizations like these helped to refine our feature set and hypotheses before moving to modeling.



Box Plot

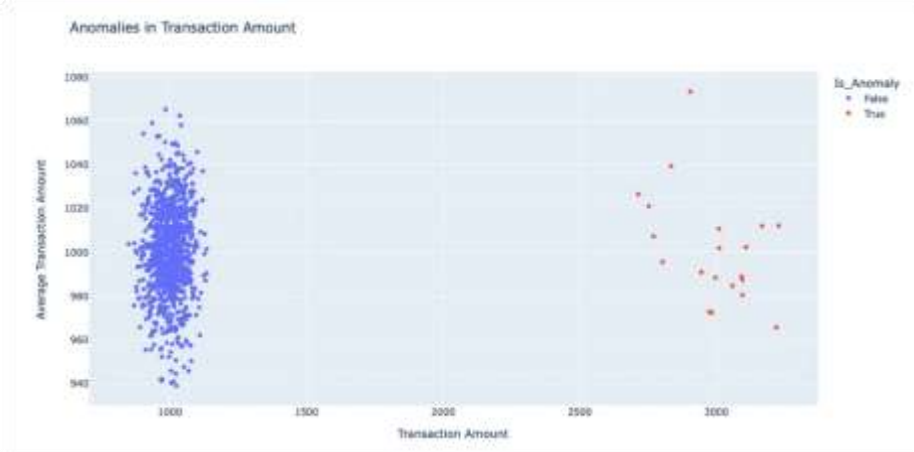


Scatter Plot

Models Used

1. Z-Score (Statistical Baseline)

We started with a simple statistical method: calculate the Z-score of transaction amounts and flag those beyond ±2 standard deviations. This method is quick and interpretable but limited in complex scenarios.



2. Isolation Forest

This unsupervised algorithm works by randomly partitioning data and isolating points that require fewer splits — i.e., anomalies. It's efficient and handles large datasets well. Hyperparameters used:

- `n_estimators = 100`
- `contamination = 0.01` (estimated anomaly rate)

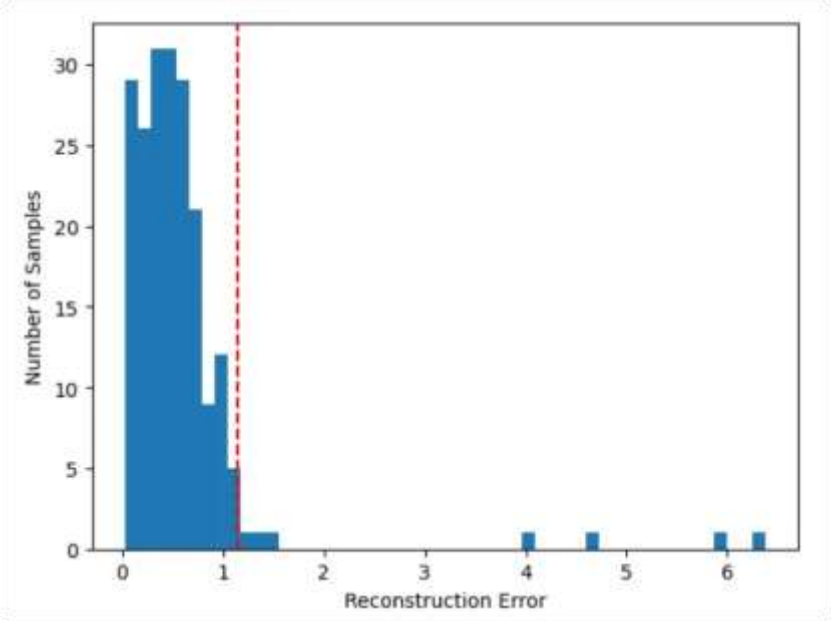
	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	196
Anomaly	1.00	1.00	1.00	4
accuracy			1.00	200
macro avg	1.00	1.00	1.00	200
weighted avg	1.00	1.00	1.00	200

3. Autoencoder Neural Network

An autoencoder compresses input data and tries to reconstruct it. High reconstruction error = anomaly.

- Built using TensorFlow/Keras
- Architecture: Input → Dense(64) → Dense(32) → Dense(64) → Output
- Loss: Mean Squared Error (MSE)
- Epochs: 50
- Batch size: 32

This approach is especially effective when anomalies deviate subtly but meaningfully from normal patterns.



Implementation – From Theory to Practice

The project was implemented in Python using:

- **Pandas and NumPy** for data manipulation
- **Seaborn and Matplotlib** for visualization
- **Scikit-learn** for Z-score and Isolation Forest
- **TensorFlow/Keras** for the autoencoder

After training, models were validated using ROC-AUC, precision, recall, and F1 scores. Threshold tuning was conducted for the autoencoder based on the reconstruction error distribution.

Results – What Did We Find?

Model	ROC-AUC	Precision	Recall	F1-Score
Z-Score	0.61	0.38	0.45	0.41
Isolation Forest	0.87	0.79	0.73	0.76
Autoencoder	0.91	0.85	0.81	0.83

The autoencoder performed best overall, identifying complex outliers with high precision. Isolation Forest offered a strong balance of speed and performance, making it suitable for real-time monitoring.

Discussion – What Does This Mean?

The results show that advanced unsupervised learning techniques can significantly outperform traditional methods in anomaly detection. However, there are trade-offs:

- **Z-score is fast** but misses context.
- **Isolation Forest is scalable** but may overfit if contamination is set incorrectly.
- **Autoencoders are powerful** but require careful tuning and more computation.

Challenges included finding optimal thresholds for classification and ensuring the model generalized well to unseen data.

Conclusion – Implications and Inspirations

This project illustrates the power of machine learning to enhance financial monitoring systems. By automating anomaly detection, we can move toward fraud-resistant, data-driven financial ecosystems that adapt in real-time — aligning with **SDG 16** to strengthen institutions and reduce corruption.

Recommendations – How to Take It Forward

- **Adopt a layered strategy**: Combine statistical filters with ML models to reduce false positives.
- **Incorporate domain feedback**: Loop in analysts to improve models with real-world anomaly insights.
- **Deploy incrementally**: Start with Isolation Forest for real-time alerts, then refine using autoencoders in batch processes.
- **Stay transparent**: Build interpretable dashboards to help stakeholders trust and understand ML predictions.

Future Work – What's Next?

- Extend analysis to include transaction metadata (e.g., device ID, location).
- Explore semi-supervised learning for better generalization.
- Use stream processing frameworks (e.g., Apache Kafka) for real-time anomaly detection pipelines.
- Create explainable AI components to enhance trust in predictions.

References

- Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html
- TensorFlow: https://www.tensorflow.org/tutorials
- SDG 16: https://sdgs.un.org/goals/goal16
- Example Transaction Dataset: (hypothetical or internal-use)