PROJECT REPORT

UNSUPERVISED CREDIT CARD FRAUD DETECTION

Submitted by: Siddhank

Reg.no-12310242

Date: February 12, 2026

Table of Contents

1. Introduction

The Context of Financial Fraud

Financial fraud is an ever-evolving challenge in the modern digital economy. As transactions shift from physical cash to digital gateways, the avenue for malicious actors to exploit system vulnerabilities has widened significantly. Credit card fraud, in particular, represents a multi-billion dollar problem for financial institutions worldwide. The sheer volume of transactions processed every second makes manual review impossible, necessitating the use of automated systems.

In the pre-digital era, fraud was often physical-stolen cards or forged signatures. Today, it is increasingly digital, involving phishing, skimming, and sophisticated man-in-the-middle attacks. As the methods of attack become more complex, the methods of defense must evolve. Static rule-based systems, once the industry standard, are no longer sufficient. They are too rigid to catch novel fraud patterns and too slow to adapt to new threats. This has led to the widespread adoption of machine learning in financial security.

The Limitations of Traditional Approaches

Historically, banks relied on rule-based systems (e.g., 'flag any transaction over $10,000'). However, these rules are brittle and easily circumvented by sophisticated fraudsters. While supervised machine learning has replaced many of these rules, it faces a significant hurdle: the lack of labeled data. Fraudulent transactions are rare events, often accounting for less than 0.1% of all traffic. Obtaining a perfectly labeled dataset is expensive, time-consuming, and reactive-we only know a transaction was fraud after the money is lost. Furthermore, fraud patterns change rapidly. A supervised model trained on last year's 'identity theft' patterns might completely miss today's 'synthetic ID' attacks.

Moreover, labeling data is not just expensive; it is often inaccurate. A customer might not report a small fraudulent charge, or they might incorrectly flag a legitimate charge as fraud (friendly fraud). These label noises can severely degrade the performance of supervised classifiers.

The Unsupervised Solution

This project explores a proactive alternative: Unsupervised Learning. Instead of teaching the model exactly what fraud looks like using labeled examples, we teach the model to identify 'anomalies'-data points that differ significantly from the established norm. By assuming that fraud is both rare and statistically different from legitimate behavior, we can leverage algorithms like the Isolation Forest to detect suspicious activity without needing a predefined list of fraud types. This approach allows for the detection of 'zero-day' fraud attacks that have never been seen before.

The beauty of this approach lies in its generalization capability. It does not look for specific signatures of known fraud (like a specific merchant ID or IP address); rather, it looks for structural deviations in the transaction data itself. If a user typically spends $50 at local grocery stores and suddenly spends $5000 on electronics in a different country, the unsupervised model flags this not because it knows it is 'fraud', but because it is an 'anomaly' for that user.

Report Structure

In this report, we detail the end-to-end development of a fraud detection system. We cover the data extraction process, the rationale behind our feature engineering choices (specifically Robust Scaling), the mathematical intuition of the Isolation Forest algorithm, and a critical evaluation of our results using metrics suited for imbalanced datasets.

2. Problem Statement and Motivation

The 'Needle in a Haystack' Paradox

The core problem we address is the extreme class imbalance found in financial datasets. In our specific dataset of 284,807 transactions, only 492 are fraudulent. This represents a fraud rate of just 0.172%. In such a scenario, standard accuracy is a deceptive metric. A naive model could simply predict 'Normal' for every single transaction and achieve 99.8% accuracy. However, this model would be useless because it fails to catch a single fraud case. The challenge, therefore, is not to maximize accuracy, but to maximize the detection of the minority class without overwhelming the system with false alarms.

This 'Needle in a Haystack' problem is compounded by the high volume of data. Processing hundreds of thousands of transactions requires algorithms that are computationally efficient. We cannot afford to run complex, O(n^2) or O(n^3) algorithms on every transaction. We need a solution that scales linearly with the data volume.

The Cost Asymmetry

Our motivation stems from the asymmetry of costs in fraud detection. A 'False Negative' (missing a fraud) results in direct financial loss to the bank and the customer, potentially leading to chargebacks and reputational damage. A 'False Positive' (flagging a legitimate transaction) causes customer friction-a declined card at a dinner-but no direct loss of funds. Therefore, our system must prioritize Recall (catching the fraud) while keeping Precision (reliability) at a manageable level. We aim to build a safety net that catches the anomalies other systems might miss.

Financial institutions often employ a tiered approach. Automated systems like ours act as the first filter. Transactions flagged as 'high risk' are then either blocked automatically or sent to a human analyst for review. By improving the precision of our automated model, we reduce the workload on human analysts, allowing them to focus on the most complex cases.

Concept Drift and Adaptability

Fraudsters are adaptive adversaries. Once a bank implements a rule to block a specific type of attack, criminals shift their tactics. This phenomenon, known as 'Concept Drift', renders static models obsolete quickly. An unsupervised anomaly detection system is theoretically more resilient to this drift. Since it models 'normality' rather than 'fraud', any deviation-whether it's a new type of hack or an old one-stands out as an anomaly. This adaptability is the primary motivation for choosing an unsupervised approach over a traditional supervised classifier.

For instance, during the COVID-19 pandemic, consumer spending habits shifted dramatically from travel and dining to online shopping and home delivery. A rigid, rule-based system might have flagged all these new online transactions as suspicious. An adaptive unsupervised model, however, would continuously update its definition of 'normal' behavior, reducing false alarms during such global shifts.

3. Theoretical Background

Unsupervised Learning for Anomaly Detection

Unsupervised learning is a paradigm of machine learning where the algorithm is provided with input data that does not have labeled responses. The system tries to learn the patterns and the structure from the data itself. In the context of anomaly detection, the algorithm builds a profile of 'normal' behavior. Any new data point that falls outside this profile is flagged. The underlying hypothesis is simple but powerful: Anomalies are few and they are fundamentally different from the majority.

There are two main types of anomaly detection: 'Novelty Detection' (training on clean data and detecting new deviations) and 'Outlier Detection' (training on a mixed dataset containing anomalies). Since our training data inevitably contains some fraud (which we don't label), we operate in the 'Outlier Detection' domain. We assume the fraud is sparse enough that it won't distort the model's understanding of normality.

The Isolation Forest Algorithm

The Isolation Forest algorithm, introduced by Liu et al., takes a unique approach. Unlike other methods that try to construct a complex model of what 'normal' looks like (e.g., Gaussian Mixture Models or One-Class SVM), Isolation Forest explicitly isolates anomalies.

The intuition is based on decision trees. Imagine you have a dataset of points. If you randomly select a feature and randomly select a split value between the max and min of that feature, you can 'cut' the data.
(1) Anomalies are rare and different (outliers). They are susceptible to isolation. They will likely be separated from the rest of the data in very few cuts.
(2) Normal points are clustered together in dense regions. Isolating a specific normal point requires many, many random cuts.

This is analogous to peeling a potato versus peeling an onion. The anomalies are on the 'surface' of the data distribution (like the skin of a potato) and are easily peeled off. The normal points are deep inside the core (like the layers of an onion) and are harder to reach.

Mathematical Intuition

Mathematically, the algorithm builds an ensemble of random trees known as Isolation Trees. For each observation in the dataset, we calculate the path length from the root of the tree to the leaf node where the observation ends up.

- Short Path Length: The point was isolated quickly. This indicates a high anomaly score.
- Long Path Length: The point required many splits to be isolated. This indicates a normal point.

The final anomaly score is an average of path lengths across all trees in the forest. This approach is highly efficient because it has linear time complexity and handles high-dimensional data well, making it computationally superior to distance-based methods like K-Nearest Neighbors for large datasets. Because it relies on tree structures, it is also invariant to the scale of features, although extreme outliers can still impact the split selection.

4. Methodology: Data Preprocessing

Dataset Characteristics

We utilized the standard Credit Card Fraud Detection dataset. It contains transactions made by credit cards in September 2013 by European cardholders. The dataset presents transactions that occurred in two days.

- Total Transactions: 284,807
- Fraudulent Transactions: 492
- Dimensionality: 30 features (Time, Amount, and V1-V28).

The features V1 through V28 are the result of a Principal Component Analysis (PCA) transformation. This was done by the data provider to protect user confidentiality. However, 'Time' and 'Amount' were left in their original, raw format. 'Time' represents the seconds elapsed between each transaction and the first transaction in the dataset. 'Amount' is the transaction amount.

Feature Scaling: Why RobustScaler?

Scaling is a critical step in algorithms that rely on partitioning logic. The 'Amount' feature is particularly problematic. Transaction amounts can range from $0 (verification checks) to tens of thousands of dollars. Standard scaling techniques, like Min-Max Scaling (squashing data between 0 and 1) or Standard Scaling (centering around mean), are highly sensitive to outliers.

For example, if the average transaction is $50, but there is one valid transaction of $25,000, the Standard Scaler will be skewed by this massive value, compressing the variance of the normal transactions. This effectively hides the patterns we want to find. When an algorithm sees a feature range of [0, 25000], and 99% of data is in [0, 100], the meaningful variations in that [0, 100] range become microscopic noise to the model.

Implementing RobustScaler

To counter this, we employed the `RobustScaler`. This scaler ignores the mean and instead removes the median. It scales the data according to the Interquartile Range (IQR)-the range between the 25th percentile and the 75th percentile.

Equation: x_scaled = (x - Q2) / (Q3 - Q1)

By using RobustScaler, we ensured that extreme outliers in the 'Amount' column did not bias our model. This allows the Isolation Forest to focus on the actual structural anomalies rather than just flagging every large purchase as fraud, which would lead to a poor user experience. This step was arguably the most critical preprocessing decision in the pipeline.

Handling Missing Values

Although our specific dataset was clean, a robust pipeline must handle missing values. In a production environment, missing values can occur due to system timeouts or data corruption. We implemented a check for null values. If found, rows with missing critical information would be dropped to preserve data integrity. Imputation (filling missing values) was avoided for this anomaly detection task because synthetic data points could inadvertently introduce false anomalies or mask real ones.

5. Methodology: The Isolation Forest

Hyperparameter Tuning

We instantiated the Isolation Forest model using the Scikit-Learn library. Several key hyperparameters dictate the model's behavior:

1. Contamination: This is the most critical parameter. It defines the expected proportion of outliers in the dataset. Since we are operating in a semi-supervised manner (we have the labels for validation), we calculated this explicitly as the ratio of fraud to total transactions (~0.0017). In a purely blind deployment, this would be an estimated threshold based on historical risk appetite.

2. n_estimators: The number of base estimators (trees) in the ensemble. We selected 100 trees. A higher number of trees provides a more stable anomaly score but increases computational cost. 100 is a standard starting point that balances performance and speed.

3. max_samples: The number of samples to draw from the dataset to train each base estimator. We used 'auto', which usually draws min(256, n_samples). Using a smaller sub-sample size is actually beneficial for anomaly detection as it reduces the masking effect where anomalies clump together.

Training Protocol

The model was trained on the feature set X (V1-V28 + Scaled Amount + Scaled Time). Crucially, we did *not* provide the labels (Y) during the training phase. The model viewed the data structure solely based on the feature distribution. The algorithm constructed the isolation trees by randomly selecting features and split values.

This randomness is a feature, not a bug. By aggregating the results of 100 random trees, the model achieves a robust consensus on which points are truly isolated (anomalies) and which are just noise. The ensemble nature of the algorithm makes it resistant to overfitting, a common plague in machine learning models.

Prediction and Thresholding

Once trained, the model assigns an anomaly score to each transaction. The Scikit-Learn implementation automatically thresholds this score based on the 'contamination' parameter we provided.

- Output -1: Anomaly (Fraud)
- Output 1: Inlier (Normal)

We post-processed these outputs, mapping -1 to 1 (Fraud) and 1 to 0 (Normal) to align with our ground-truth labels for evaluation.

It is worth noting that the 'contamination' parameter acts as a trade-off slider. Increasing it will flag more transactions as fraud (increasing Recall but decreasing Precision). Decreasing it will make the model more conservative (increasing Precision but potentially missing fraud). We stuck to the statistical prevalence of fraud for this parameter to maintain a balanced approach.

6. Implementation Details

Technical Stack

The project was implemented using Python 3.8, leveraging the data science ecosystem.

- Pandas: Used for high-performance data manipulation. DataFrames were used to manage transaction records, handle missing values (though none were present), and perform feature dropping.
- NumPy: Used for high-speed numerical operations and array handling.
- Scikit-Learn: Provided the implementation of Isolation Forest, RobustScaler, and PCA.
- Seaborn/Matplotlib: Used for visualizing the class imbalance and the confusion matrix.

Code Architecture

The codebase is structured into a logical pipeline:

1. Data Ingestion: Secure loading of the CSV file with error handling for file existence. We implemented try-except blocks to ensure the script fails gracefully if the data source is missing.
2. Exploratory Data Analysis (EDA): Initial inspection of data shapes and class distributions to confirm the imbalance. We visualized the imbalance using bar charts to confirm our hypothesis about the dataset's nature.
3. Preprocessing: Transformation of raw columns into model-ready features using RobustScaler. This step ensured that the 'Amount' column was on the same scale as the PCA features.
4. Model Training: Unsupervised fitting of the Isolation Forest. We fixed the random seed to 42 for reproducibility.
5. Evaluation: Comparing the unsupervised predictions against the hidden labels to generate metrics.
6. Visualization: Reducing the 30-dimensional data to 2 dimensions using PCA to visually confirm the separation of fraud cases. This plot serves as a powerful diagnostic tool.

Reproducibility

To ensure the results are reproducible, we set a fixed `random_state=42` across all stochastic components (Train-Test Split and Isolation Forest). This ensures that the random subsampling of trees is identical every time the code is run, allowing for consistent debugging and performance tuning. The code was designed to be modular, allowing for easy swapping of the algorithm (e.g., to Local Outlier Factor) without rewriting the entire pipeline.

We also implemented a dummy data generator. In the event that the source CSV file is missing, the script automatically generates a synthetic dataset with similar statistical properties. This ensures that the pipeline can be tested and demonstrated even without access to the confidential source data.

7. Results and Evaluation

Evaluation Strategy

Evaluating an unsupervised model on a highly imbalanced dataset requires nuance. Standard 'Accuracy' is misleading. If we predicted every transaction as legitimate, we would achieve 99.8% accuracy but catch 0 fraud. Thus, we focused on the Confusion Matrix and the F1-Score.

Confusion Matrix Analysis

The Confusion Matrix provides a breakdown of our predictions:

- True Positives (TP): Fraud transactions correctly caught. This is our primary success metric.
- False Negatives (FN): Fraud transactions we missed. We want to minimize this number to zero.
- False Positives (FP): Normal transactions we incorrectly flagged as fraud.
- True Negatives (TN): Normal transactions correctly identified as normal.

Our model successfully identified a significant portion of the fraudulent cluster. The PCA visualization confirmed that the detected anomalies (Red points) were indeed located in the sparse, outlying regions of the feature space, validating the Isolation Forest's core hypothesis. Visually, we could see a dense 'blob' of blue points (normal) and a scattered halo of red points (anomalies) that our model correctly circumscribed.

Performance Metrics

- Precision: The probability that a flagged transaction is actually fraud. In unsupervised anomaly detection, this is typically lower than supervised methods because the model detects *all* anomalies, not just fraud (e.g., a customer making a very large but valid purchase).
- Recall: The percentage of total fraud caught. Our model prioritized Recall, ensuring that we cast a wide net.
- F1-Score: The harmonic mean of Precision and Recall.

While supervised models like XGBoost might achieve higher precision, the Isolation Forest demonstrated a respectable ability to filter out the noise without any labeled training data, proving its value as a 'first line of defense' system. Our F1-score indicates a balanced performance, suggesting that while we do generate some false alarms, the cost of these alarms is justified by the number of fraud cases we successfully intercept.

8. Conclusion and Future Scope

Project Conclusion

This project successfully demonstrated the viability of Unsupervised Learning for financial fraud detection. By leveraging the Isolation Forest algorithm, we constructed a pipeline capable of detecting fraudulent transactions without relying on historical labels. The key takeaways include:

1. Robust Preprocessing is Vital: The use of `RobustScaler` was essential. Without it, the variance in transaction amounts would have masked the subtle patterns of fraud.
2. Unsupervised Learning Works: Even without being told what fraud looks like, the model found it by looking for what 'normal' did not look like.
3. Trade-offs are Real: We accept a higher number of False Positives (lower precision) to ensure we do not miss actual fraud (higher recall).

This system is not a silver bullet but a powerful tool in a layered security architecture. It excels at finding the 'unknown unknowns'-the novel fraud attacks that rule-based systems miss.

Future Scope

There is significant room for expansion:

1. Deep Learning Autoencoders: A neural network trained to reconstruct normal transactions. High reconstruction error implies fraud. This often captures complex non-linear relationships better than trees.
2. Ensemble Voting: Combining Isolation Forest with Local Outlier Factor (LOF) and One-Class SVM. A transaction is only flagged if 2 out of 3 models agree. This would drastically reduce False Positives.
3. Real-Time Streaming: Porting the logic to Apache Spark or Kafka to flag transactions in milliseconds. The current batch processing approach introduces a delay that fraudsters can exploit.
4. Feature Enrichment: Incorporating geolocation and device fingerprinting to improve the signal-to-noise ratio. Knowing that a transaction originated from a device never used by the customer before is a powerful signal.

References

1. Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation Forest. IEEE International Conference on Data Mining.
2. Scikit-Learn Documentation. (n.d.). 2.7. Novelty and Outlier Detection.
3. Kaggle Dataset: Credit Card Fraud Detection. ULB Machine Learning Group.
4. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys.
5. Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial intelligence review.