USTH
VIETNAM FRANCE UNIVERSITY

Machine Learning and Data Mining II

Team Project Report

FRAUDULENT TRANSACTION DETECTION

One Class Classification [EN]

Authors:

Quoc Anh Le	ID - 23BI14025
Thai Nhat Anh Nguyen	ID - 23BI14024
Duy Anh Pham	ID - 23BI14023
Quang Vinh Pham	ID - 23BI14455

Group 10 | B2 DS

2025/2026 – 4th Semester

Abstract

Anomaly detection plays a critical role in identifying rare and irregular patterns within large datasets, particularly in domains such as fraud detection where labeled anomalies are scarce. One-Class Classification (OCC) techniques have emerged as effective solutions for these scenarios, focusing on learning patterns of normal data to detect outliers. In this report, we investigate the application of OCC to credit card fraud detection using the highly imbalanced IEEE-CIS Fraud Detection dataset, where only 3.5% of transactions are fraudulent. We evaluate both classical OCC methods—One-Class SVM, Isolation Forest, and Elliptic Envelope—and deep learning-based models such as Autoencoders and Deep SVDD. Our study includes a robust data preprocessing and feature engineering pipeline, along with temporal and behavioral analysis to uncover fraud patterns. Additionally, we propose a GAN-based data augmentation strategy to mitigate the lack of fraudulent samples. Experimental results, measured using ROC AUC and Average Precision, show that deep OCC models slightly outperform classical ones. However, the performance gap remains narrow, emphasizing the difficulty of fraud detection in real-world, noisy, and imbalanced settings. We conclude with suggestions for future work involving hybrid or semi-supervised approaches to further improve anomaly detection capabilities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 OCC Concept and Motivation

One-Class Classification (OCC) is a machine learning approach focusing on modeling the "normal" class when little or no labeled anomaly data is available. OCC methods learn the distribution or boundary of normal data; anything significantly deviating from this learned concept is flagged as an outlier or anomaly. This is especially useful for highly imbalanced datasets, or in situations where anomalies (e.g., fraud, failures) are rare or hard to label.

1.2 Dataset: IEEE-CIS Fraud Detection

1.2.1 General Information

- **Source:** The dataset used is the IEEE-CIS Fraud Detection dataset, released by IEEE and Vesta Corporation for a Kaggle competition. It simulates real-world online transaction records and aims to help develop robust fraud detection systems.
- **Purpose:** The main goal is to experiment with different OCC models on highly imbalanced dataset in order to find the one that has the best performance. The challenge is made more realistic by the severe class imbalance: only about 3.5% of all transactions are fraudulent.
- **Why use this dataset:** It is labeled, large-scale, features both transactional and device/identity information, and contains many anonymized but realistic variables. The severe class imbalance makes it a prime use-case for anomaly detection and one-class classification.

1.2.2 Dataset Structure

- `train_transaction.csv`: Transactional features (amount, product, card info, addresses, etc.), with `isFraud` as the label.
- `train_identity.csv`: Device and identity features (device type, browser, OS, email domain, etc.).

1.2.3 Metadata

The IEEE-CIS Fraud Detection dataset consists of two main files:

Column(s)	Description
TransactionID	Unique identifier for each transaction
isFraud	Target variable; 1 for fraudulent transactions, 0 for legitimate
TransactionDT	Time delta from a reference point; used for time-based features (hour, day, etc.)
TransactionAmt	Transaction amount; important for detecting unusual activity
ProductCD	Anonymized product category
card1–card6	Card identifiers, type, and issuer (some partially anonymized)
addr1, addr2	Billing region and country codes
dist1, dist2	Distance features, possibly related to user location or IP
P_emaildomain, R_emaildomain	Email domains for purchaser and recipient
C1–C14	Count features, e.g., frequency of product or user behavior
D1–D15	Time-based deltas (e.g., days since account creation or previous transaction)
M1–M9	Binary match flags (such as address, device match)
V1–V339	Engineered anonymized features with strong predictive signals

Table 1.1: Columns in `train_transaction` (Transaction-level features)

Column(s)	Description
TransactionID	Join key for the transaction table
id_01–id_11	Numeric identity features (possibly age, time lags)
id_12–id_38	Binary/categorical identity verification flags
DeviceType	Type of device (mobile or desktop)
DeviceInfo	Raw information about the device (browser, OS, model)

Table 1.2: Columns in `train_identity` (Identity and device metadata)

The two tables are joined using `TransactionID`.

1.2.4 Key characteristics:

- **Extreme class imbalance**: Only about 3.5% of transactions are fraudulent.
- **Rich, high-dimensional feature space**: Hundreds of anonymized features, many derived from device, session, and behavioral metadata.
- **Realistic noise and missing values**: Many features have missing data or have been deliberately anonymized to reflect real-world transaction data challenges.
- Too many Null values varying in different proportions in columns, requiring custom ways of dealing with each case.

A thorough understanding of the metadata, as well as careful feature engineering and missing value handling, is critical for successful fraud detection and the effective training of one-class classification models.

1.3 Models Used

For experimentation, we use both **machine learning** and **deep learning** based OCC models:

- **Classical/ML:**
  - OCC: One-Class SVM
  - OCC: Elliptic Envelope
  - Semi-OCC: Isolation Forest
  - Non-OCC: LOF (Adapted for OCC)
  - Non-OCC: kNN (Adapted for OCC)
- **Deep Learning (Adapted for OCC):**
  - Deep SVDD
  - Autoencoder
  - GAN (Generative Adversarial Network)

Chapter 2

Technical Details

2.1 OCC Methodology

One-Class Classification (OCC) is a machine learning paradigm designed to learn from data that primarily or exclusively consists of the "normal" (majority) class. Rather than discriminating between two or more classes, OCC learns the structure or boundary of the normal class and flags deviations as anomalies. Here are the key steps for training and testing:

- **Training:** The OCC model is trained using only samples from the normal class, learning a representation of "normality."
- **Testing:** When new data is encountered, points that deviate substantially from the learned normal region are labeled as outliers or anomalies.

OCC is especially useful for highly imbalanced scenarios—such as fraud detection, rare disease identification, or system fault detection—where anomalous cases are scarce, unknown, or too expensive to label.

2.2 Core Principles

- **Unsupervised learning:** OCC uses only normal data for training, unlike conventional supervised learning that requires both normal and anomalous samples.
- **Boundary or density modeling:** OCC methods can be geometric (learning a boundary or envelope) or probabilistic (modeling the data density).

2.3 Main Algorithms

2.3.1 One-Class SVM

Overview: One-Class SVM extends support vector machines to unsupervised anomaly detection by learning a decision boundary that encompasses the majority of "normal" data, treating points outside as anomalies.

How it works:

- Projects data into a high-dimensional feature space using a kernel function (e.g., RBF).
- Attempts to separate all data points from the origin with maximum margin.
- Outliers fall outside this region.

Mathematical Formulation:

$$\begin{aligned} \min_{w,\rho,\xi_i} \quad & \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \\ \text{subject to} \quad & (w \cdot \phi(x_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0 \end{aligned}$$

where w is the normal vector, ρ the bias, ξ_i slack variables, ν controls trade-off between errors and support vectors, and $\phi(x)$ is the kernel mapping.

Prediction: Test point x is "normal" if $(w \cdot \phi(x)) \geq \rho$, else "anomaly".

2.3.2 Elliptic Envelope

Overview: Fits a multivariate Gaussian (ellipse) to the data; points that deviate too far are outliers.

How it works:

- Estimates mean (μ) and covariance (Σ).
- Calculates Mahalanobis distance for each point.

$$d_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

Points with large $d_M(x)$ are flagged as anomalies.

2.3.3 Isolation Forest

Overview: Isolation Forest is an ensemble anomaly detector that partitions the data randomly, based on the principle that anomalies are easier to isolate than normal points.

How it works:

- Builds random trees by recursive splits.
- Points that require fewer splits (shorter path) to isolate are likely anomalies.

Scoring:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where $h(x)$ is the path length for x, $E(h(x))$ is the average path length, and $c(n)$ normalizes for sample size.

Interpretation: $s(x,n) \approx 1$ indicates anomaly, $s(x,n) \ll 0.5$ indicates normal.

Chapter 3

Implementation on Fraud Detection Dataset

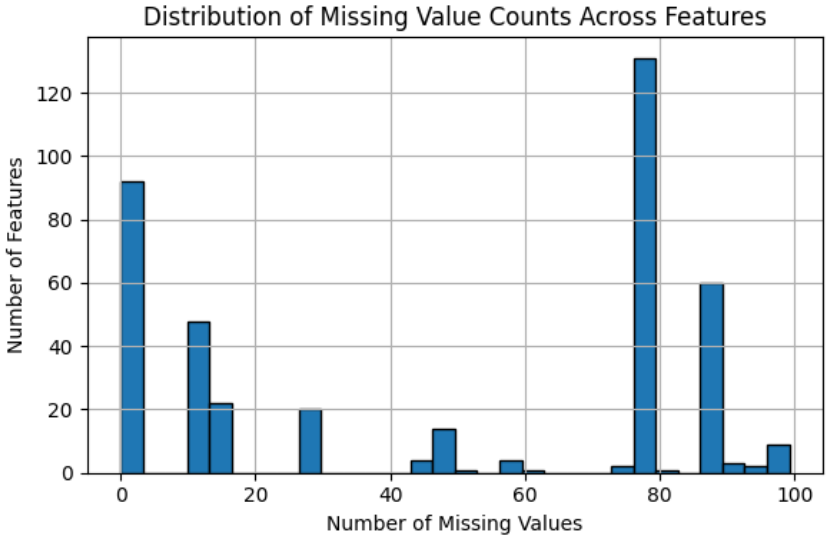
3.1 Data Preprocessing

The dataset consists of two main components:

- **Identity Table (ID)**: 144,233 rows $\times$ 41 columns
- **Transaction Table**: 590,540 rows $\times$ 394 columns

Both tables share a common `TransactionID` column, allowing us to merge them using a **left join**, with the transaction table serving as the primary reference. After joining, the final dataset contains **590,540 rows and 434 columns**.

However, this merging introduces a significant number of missing values due to the nature of the identity table (many entries in the transaction table do not have corresponding identity information).



To mitigate the sparsity, columns with more than **80% missing values** were dropped, resulting in a refined dataset of shape **(590,540, 358)**. The remaining columns were then grouped into clusters based on their semantic and functional similarity for preprocessing. The table below summarizes the cleaning strategy for each column cluster.

Column Cluster	**Description**	**Initial Null Level**	**Data Type**	**Ways to Clean**
P_emaildomain, R_emaildomain	Email domain of Purchases (buyer) and Receiver (seller)	16%, 76.75%	Categorical	Delete R_emaildomain and set 2 main labels of P_domain as "email" and "yahoo", leave the rest as "others"
TransactionID	ID of the Transaction (only for joining tables)	–	Categorical	Delete
isFraud	Index indicating Transaction as Fraud (1) or Nonfraud (0)	0%	Categorical	Keep as label
TransactionDT	Time Difference of Transaction from a known anchor	0%	Numerical (Discrete)	Divide TransactionDT by 3600 to get hours difference
TransactionAmt	Amount of the transaction	0%	Numerical (Continuous)	Keep (may need log transformation for model building)
ProductCD	Product Category	0%	Categorical	Keep
Card1 - Card6	Card information (IDs, bank ID, type, brand)	0% - 1.5%	Categorical	Low proportion of Null $\rightarrow$ impute by Median
Addr1, Addr2	Billing Region, Country Code	11.1%	Categorical	**Addr1**: Take Top 8 regions, others to "others" **Addr2**: Take Top 1 country (500k), others to "others"
Dist1	Distance between billing and cardholder address	59.6%	Numerical (Continuous)	Group by Addr1 (billing region), fill with group mean
C1 - C14	Counting features (frequency of products or behavior)	0%	Numerical (Discrete)	Keep, may be reduced by Heatmap

Column Cluster	**Description**	**Initial Null Level**	**Data Type**	**Ways to Clean**
D1 - D15	***D1***: Days since last login (or first observed activity) ***D2***: Days since card was issued or account was created ***D3***: Days since last known address update ***D4***: Days since last transaction with same card/account ***D5***: Days since last transaction using the same email or browser ***D10***: Days since first transaction on current session ***D11***: Days since device was first associated with the account ***D15***: Days since last online interaction with the current device	0.21% 47.5% 44.5% 28.6% 52.46% 12.87% 47.3% 15%	Numerical (Discrete)	***D1*** Fill with Median (low na) ***D2*** Relies on Card1 (CardID), take group mean ***D3*** Relies on addr1, addr2, take group mean ***D4*** Relies on Card1 (CardID), take group mean ***D5*** Relies on P_emaildomain, take group mean ***D10*** Relies on Card1 (CardID), take group mean ***D11*** Drop due to information mismatch and high null ***D15*** Drop due to no Device Information column
M1 - M9	Information Matches (Show different matches of a Transaction: matching amount, matching time, id,.....)	45.9%	Categorical (mostly binary)	Mark Found - Not-Found/ T-F Verification done during a transaction, can be aggregated to 1 column. Replace all Null with 'Missing', and create a new Column name 'VerificationSet' to count number of Finished Verification based on Each ID return T or 'Found' or 'New'
id_12 - id_38	Verification flags during transaction	75.5%	Categorical (binary/trinary)	Replace Null with "Missing", create 'VerificationSet' column counting T/Found/New
id_01 – id_11	Numeric identities of transaction/purchaser	76.1%	Numerical (Continuous)	Drop: too much null, lack of information

Column Cluster	**Description**	**Initial Null Level**	**Data Type**	**Ways to Clean**
DeviceType	Type of device used in transaction	76.1%	Categorical	Replace Null with "missing", keep "mobile" and "desktop"
DeviceInfo	Detailed information of device	80%	Text	Drop: high null and too many distinct values

3.2 Exploratory Data Analysis (EDA)

3.2.1 Key Insights from Exploratory Data Analysis

Target Imbalance & General Correlations:

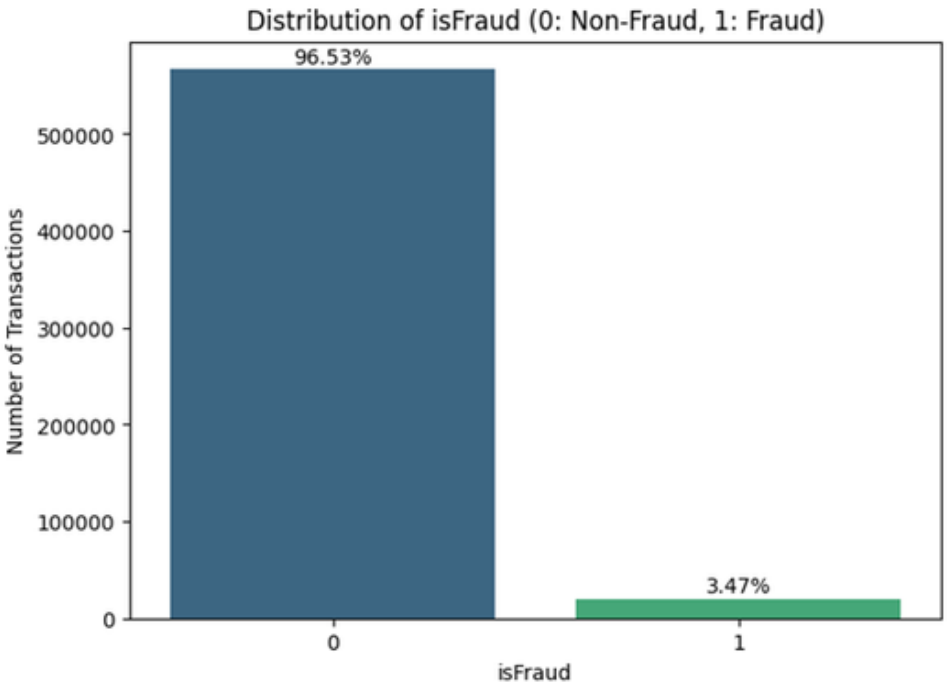


Figure 3.1: Distrbution of `isFraud`

The dataset exhibits a pronounced class imbalance, with only 3.47% of transactions labeled as fraudulent. This extreme skew justifies the use of a one-class classification (OCC) strategy that focuses on modeling the dominant "normal" class.

Furthermore, the majority of features demonstrate very low linear correlation with the target variable `isFraud`, indicating that fraudulent behavior is not easily captured through simple linear relationships. Instead, the patterns are likely to be complex, non-linear, and multi-dimensional. These characteristics align well with the strengths of OCC models, which are adept at detecting subtle deviations from the underlying distribution of legitimate transactions.

Core Numerical Features: `TransactionDT` and `TransactionAmt`

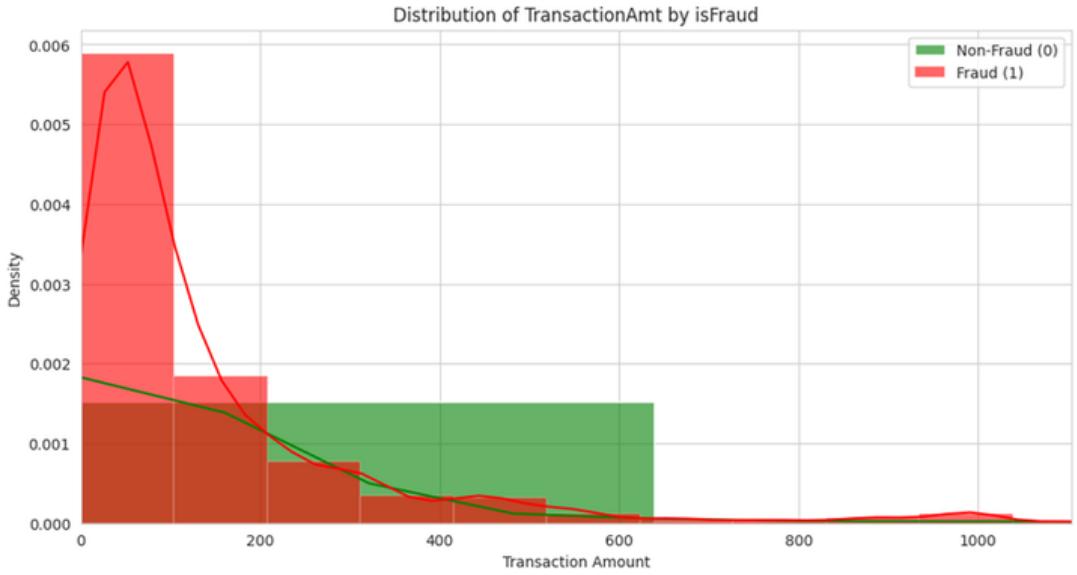


Figure 3.2: Distribution of Transaction Amount

`TransactionAmt` (Transaction Amount) reveals meaningful distinctions between fraudulent and legitimate transactions. Fraudulent transactions tend to exhibit higher density at lower amounts, while also displaying a slightly higher median and a broader spread. This dual pattern suggests diverse fraudulent strategies, ranging from small-scale test transactions to more substantial, high-risk activities.

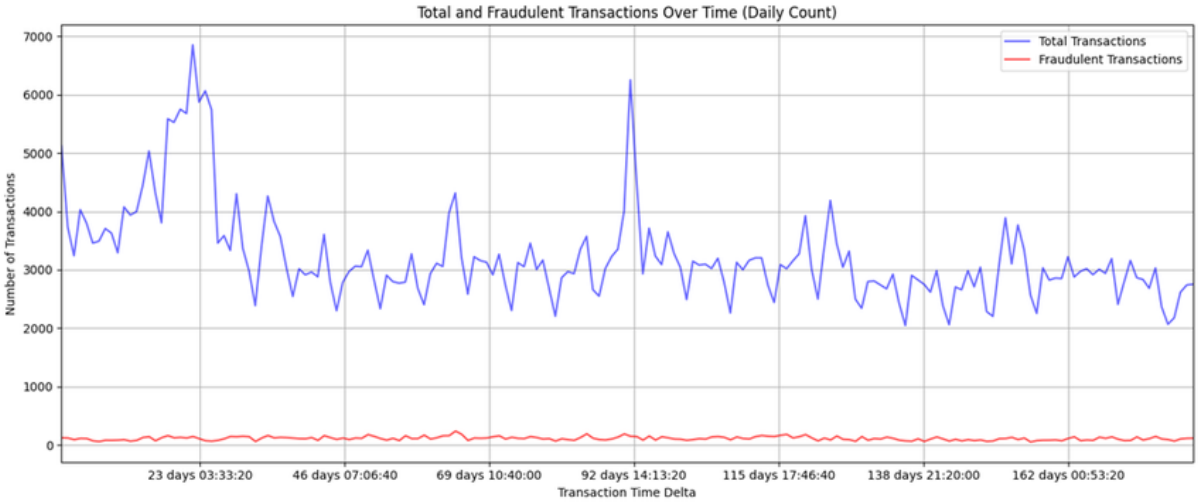


Figure 3.3: Total and Fraudulent Transactions Over Time (Daily Count)

`TransactionDT` (Transaction Timestamp) highlights temporal dynamics in fraudulent behavior. Although the overall volume of transactions remains relatively stable over time, the proportion of fraudulent transactions fluctuates significantly on a daily basis, with notable spikes exceeding 7%. This temporal volatility indicates that fraudulent activity is not uniformly distributed over time and reinforces the importance of incorporating time-based features in fraud detection models.

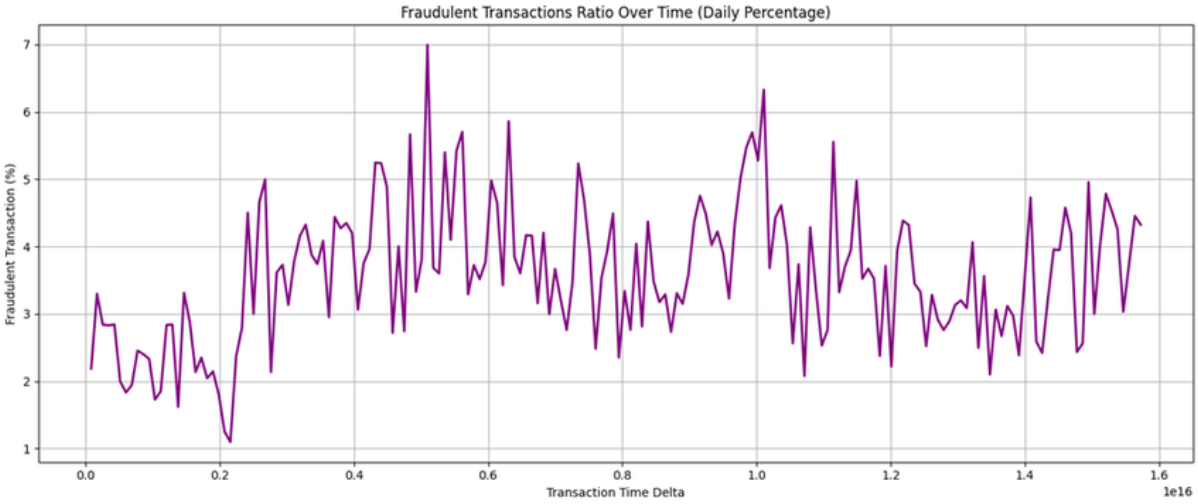


Figure 3.4: Fraudulent Transactions Ratio Over Time (Daily Percentage)

Core Categorical Features: `card4`, `card6`, and `P_emaildomain`

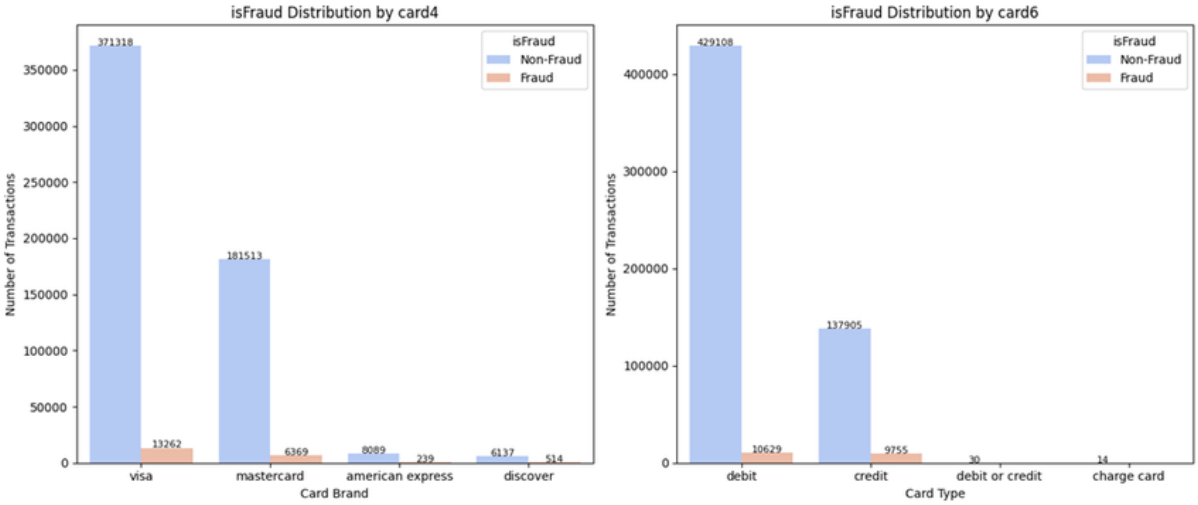


Figure 3.5: `isFraud` Distribution by `card4` and `card6`

Among the categorical variables, `card4` (card brand) is dominated by Visa and Mastercard, which naturally account for the largest number of fraudulent transactions due to their high transaction volume. However, brands like American Express and Discover, though less frequent, may exhibit disproportionately high fraud rates relative to their usage—warranting further investigation into their risk profiles.

The `card6` feature (card type) shows a significant skew toward debit cards, which appear far more frequently than credit cards. It is essential to analyze fraud incidence per card type, as certain types may be targeted more often due to lower security protocols or user behavior.

For `P_emaildomain` (purchaser email domain), `gmail.com` stands out with the highest fraud ratio (4.31%) among the top email domains. This may indicate a preference among fraudsters for widely used, easily accessible services that offer greater anonymity and ease of account creation.

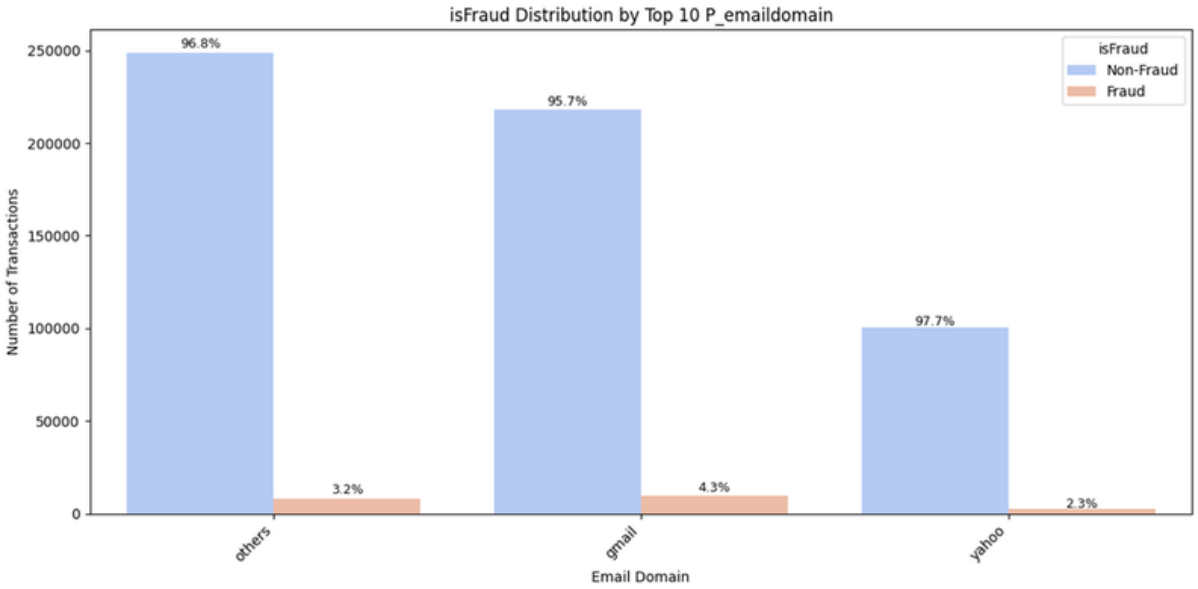


Figure 3.6: `isFraud` Distribution by `top10` $P_{E}maildomain$

Other Highly Discriminative Features

Several additional features demonstrate strong discriminatory power for fraud detection. Notably, `ProductCD == 'C'` is a highly indicative fraud marker, exhibiting an exceptionally high fraud rate of 11.81%, far above the dataset's average. This suggests that certain product categories may be inherently riskier or more frequently exploited.

The `Dist1` variable, which captures the distance between billing and cardholder addresses, also reveals meaningful patterns. Fraudulent transactions are associated with larger distances, particularly a prominent peak around 400 units, whereas legitimate transactions tend to cluster around minimal distance values—likely indicating local or routine behavior.

Count-based features such as `C3`, `C7`, and `C8` provide behavioral signals linked to fraud. Higher values in these fields, which may correspond to failed login attempts, declined transactions, or other interaction counts, are strongly associated with suspicious activity.

Time-based delta features (`D`-series), such as the time since prior user activity, often show inflated values for fraudulent transactions. This may reflect less frequent or atypical usage patterns, suggesting that temporal irregularity is a meaningful signal in detecting fraud.

Finally, the `Card1` identifier reveals that certain card IDs are repeatedly involved in multiple fraudulent events. Monitoring card-specific behavior over time can be critical for identifying recurring fraud and enhancing anomaly detection mechanisms.

3.2.2 Temporal Analysis

As the dataset contains the temporal features of Card Transactions along the time, itś a practical approach to look at how the Transaction spans over time may affect the Flag condition of the Card.

In this part, only certain columns will be used for Analysis

- **CardID**: Mark the Distinct Card identifier
- **TransactionDT**: Time Difference of a transaction to a set anchor.
- **isFraud**: whether the transaction is flagged as Fraud, will be used for overall card aggregation
- Value columns: contain the value of the time series, in this part 2 different components will be used **"TransactionAmt"** and **14 Engineering Encoded features**.

Firstly, a graph containing some Card transactions over time will be plotted to get an Overall grasp of the transaction behavior of certain Fraud-flagged cards

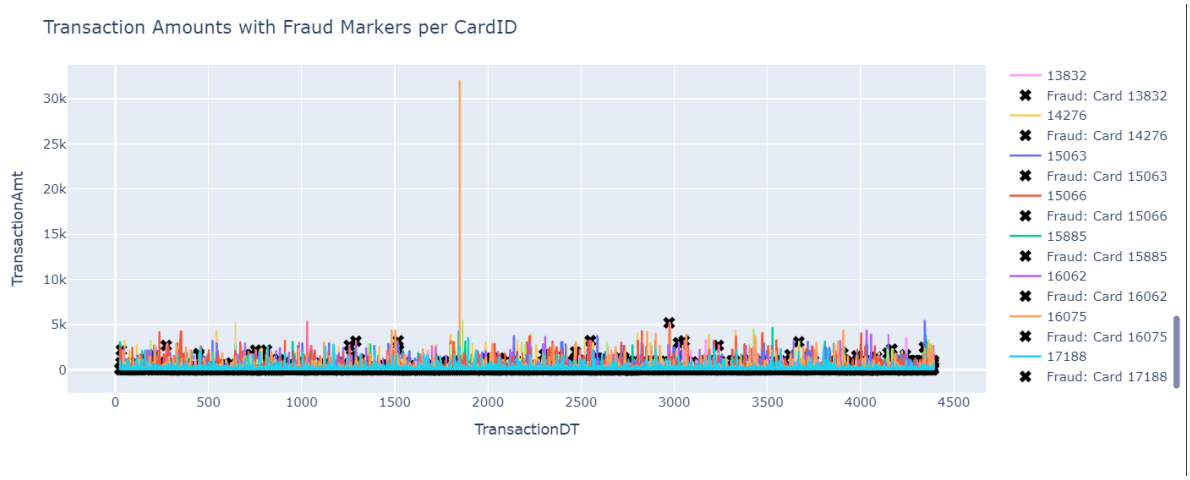


Figure 3.7: 31 Fraud Card Transactions

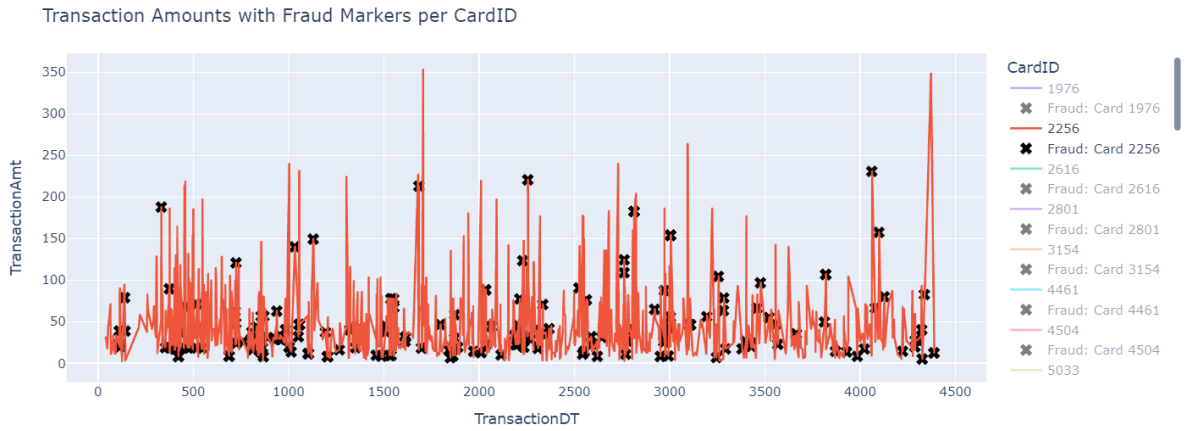


Figure 3.8: Example of 1 Fraud card

From this figure, an overall view of the behavior is caught as there is no actual relation between whether a Transaction is flagged with the Time difference and traded amount. However, a several more tests need to be conducted to fulfill this hypothesis.

The test is to use TimeSeries Clustering algorithms (**TimeSeriesKMeans**) to split the Card Transaction logs into cluster and use **Crosstab** to see how each Cluster interacts with the actual Label.

The steps will be conducted as below:

- Create a dataset that groups the initial dataset by CardID and set agg(max) to isFraud, this means that a Card with at least 1 fraud transaction will be flagged as Fraud. The Output dataset has 13543 cards (1735 Frauds and 11808 Non-Frauds).
- Expand the Label Diversity by calculating the Fraud Level of each cards

$$FraudLevel = \frac{\sum Fraud}{\sum All}$$

and a Fraud card will be divided by Low/ High Frauds with Low Fraud cards having FraudLevel $< 10\%$

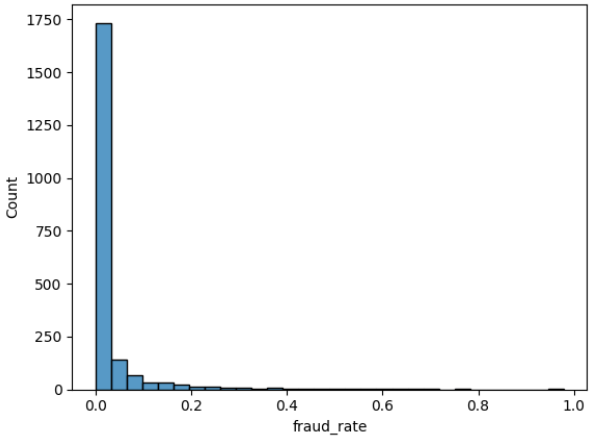


Figure 3.9: Fraud Ratio of Cards

- Select only Cards with more than 30 and less than 200 Transactions to ensure computational scalability and Label balance.

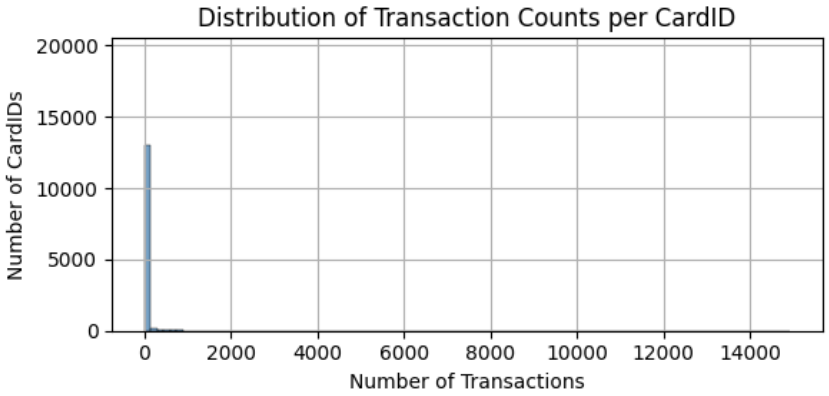


Figure 3.10: Transaction Number of Cards

The relabel dataset has 1491 cards (971 Non-Frauds, 300 Low Frauds, 220 High Frauds). Note that this is caused due to the cutting by Transaction Length, which may cause unexpected outcome.

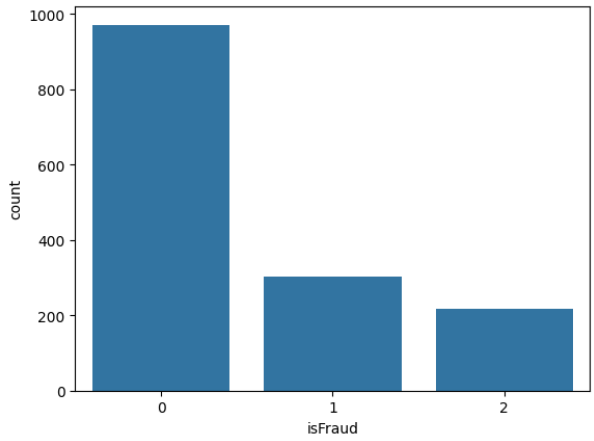


Figure 3.11: Relabeled Dataset

- Use **TimeSeriesResampler** to pad/ truncate all Card to 50 fixed transactions. This will ensure the size balance between inputs for the model.
- Preprocess the Value columns, in this part 2 kinds of values will be used: **TransactionAmt** and **Engineering Encoded Features**. TransactionAmt can be kept for integrity and the E-E features will be compressed into 1 column via 2 methods **Autoencoder and Weighted Sum by Variance Explained**.

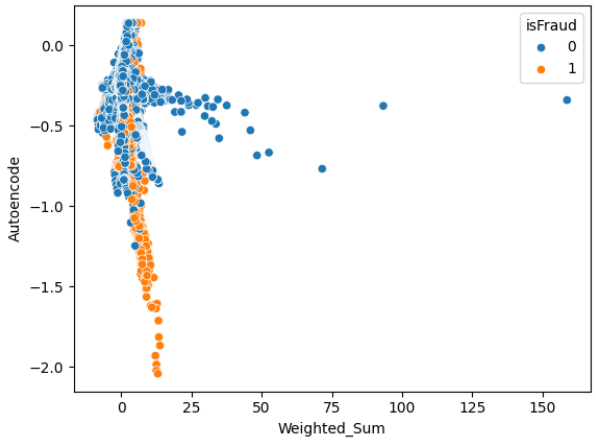


Figure 3.12: Label Distribution between Different Approaches

Card characteristics tend to stay more in the lower value of Weighted Sum approach and just having some noises in higher values, the Weighted Sum approach failed to differentiate between Fraud/ Non-Fraud. While in the Autoencoder axis, a clear relation can be seen as Non-Fraud cards stay in cluster in higher values where Fraud cards tend to distribution evenly along the axis but more focus in the lower values.

- Fit the TS KMeans model with fixed 4 cluster and **SoftDTW** metric, draw a Crosstab to compare each cluster with the Labels.

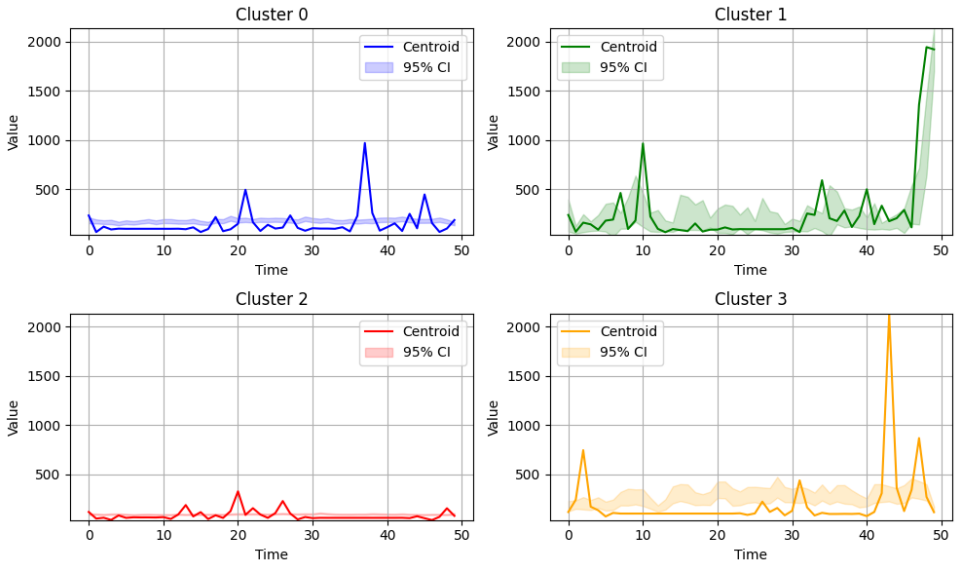


Figure 3.13: TransactionAmt Clusters

isFraud	**0**	**1**	**2**
Clusters			
0	234	108	60
1	14	2	1
2	658	152	137
3	65	40	20

Note that the Solid line is the **DTW barycenters** (centroids), not the Mean value of the time series lying between the 95% Confidence Interval. The DTW Barycenters mostly account for temporal misalignments by softly aligning similar parts of different series even if they occur at slightly different times, they don't explain the central and skewness of the data.

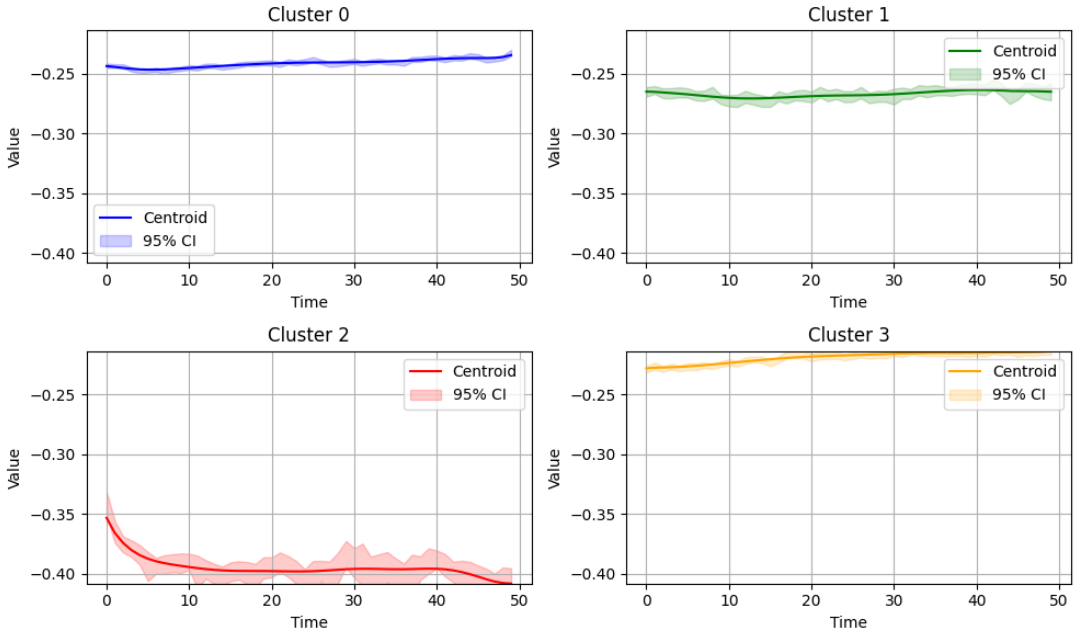


Figure 3.14: Autoencoder Clusters

isFraud	**0**	**1**	**2**
Clusters			
0	477	160	84
1	109	49	38
2	31	21	72
3	354	72	24

From this a clear trend is observed where most of the Cards will stay in Clusters with more stable values (0 and 2 for TransactionAmt or 0 and 3 for E-E Features), while very few cards are in stochastic, high variance clusters. Moreover, the Class distribution in each cluster somewhat replicates the actual label distribution, hence showing no novel pattern (Only in E-E features where **High Fraud** value tend to be superior in the Cluster 2 where values are much smaller). This helps explain that most of the Cards Transaction logs are stable and very similar to each other instead showing a clear difference between Non-Fraud/ Low-Fraud/ High-Fraud.

3.2.3 Implications for One-Class Classification

This exploratory data analysis (EDA) provides a strong foundation for a one-class classification (OCC) model. The severe class imbalance necessitates an approach focusing on modeling the majority "normal" class. Key fraud indicators identified include:

- Distinct patterns in `TransactionAmt` and temporal spikes in `TransactionDT` fraud ratio.
- Specific categorical values (`ProductCD == 'C'`, `gmail.com` in `P_emaildomain`, potential `card4`/`card6` insights).
- Behavioral shifts (`Dist1 > 0`, higher `C`-feature counts, larger `D`-feature deltas).
- Compromised `Card1` IDs.

Strategy:

In the preprocessing stage, it is important to manage high-cardinality categorical variables through methods such as grouping or target encoding, while numerical features should be properly scaled—particularly for algorithms that rely on distance measures.

Model selection should consider established OCC techniques such as Isolation Forest, One-Class SVM, and Local Outlier Factor (LOF), each of which offers unique advantages in identifying anomalies in imbalanced datasets.

Given the severe class imbalance inherent in fraud detection, evaluation metrics must emphasize the minority class. Accordingly, metrics like Precision, Recall, F1-score, and the Area Under the Precision-Recall Curve (AUC-PR) are recommended for assessing model performance.

3.3 Feature Engineering

3.3.1 Categorical Feature Handling

Reducing Cardinality. Columns such as `card2` contain many infrequent values. To enhance model stability and reduce overfitting, rare values (occurring less than 1,000 times) in `card2` were grouped into a new category (`"others"`).

Dropping Non-informative/High-cardinality Columns. Certain features such as `card1` and `card5` were dropped due to their extremely high cardinality, limited interpretability, or potential for data leakage. This step is especially important for OCC models, which may overfit on near-unique identifiers.

3.3.2 Encoding and Scaling

Categorical Variables. Categorical columns were encoded according to their type:

- **OrdinalEncoder**: Used for ordinal features (e.g., `addr2`)
- **OneHotEncoder**: Used for nominal features (e.g., `card2`, `ProductCD`, `P_emaildomain`, etc.)

Numerical Variables. All numerical features were standardized to zero mean and unit variance using `StandardScaler`, ensuring comparability and preventing scale dominance in distance-based models.

3.3.3 Dimensionality Reduction

Incremental PCA. Due to the high number of features, `IncrementalPCA` was applied to reduce dimensionality:

- 15 components for OCSVM and Elliptic Envelope
- 10 components for LOF and k-NN

This approach speeds up computation and reduces the risk of overfitting.

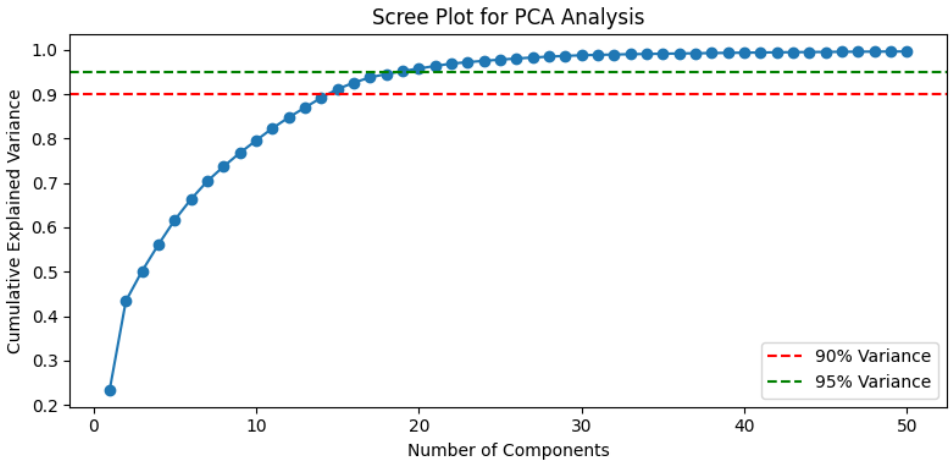


Figure 3.15: Cumulative explained variance ratio of PCA components.

3.3.4 Validation Strategy

Although the original dataset provides a dedicated test set (for supervised binary classification), it was not used for OCC model selection or evaluation. Instead, the available training data was split into new training and validation subsets:

- **Training:** Only normal samples were used to fit the models.
- **Validation:** The validation set, containing both normal and fraud transactions, was used to assess performance.

Evaluation metrics focused on ROC AUC and Average Precision, providing a robust measure of model capability in the face of class imbalance.

3.3.5 Choice of Metrics

- **ROC-AUC (Area Under the Receiver Operating Characteristic Curve):**
  ROC-AUC evaluates the trade-off between true positive rate (sensitivity) and false positive rate across all classification thresholds. An AUC close to 1.0 indicates excellent discrimination between normal and fraudulent transactions, while 0.5 suggests random guessing. ROC AUC is robust to class imbalance and widely used in anomaly detection.
- **Average Precision (AP, Area Under the Precision-Recall Curve):**
  Average Precision summarizes the trade-off between precision (positive predictive value) and recall (sensitivity) across different thresholds. It is particularly informative for highly imbalanced datasets, where the minority class (fraud) is of primary interest. AP reflects the expected precision averaged over all levels of recall, and is sensitive to both false positives and false negatives.
- **Why Not Use Accuracy or Precision Alone?**
  With imbalanced data, a classifier that predicts all samples as non-fraud can achieve high accuracy but zero practical value. Similarly, focusing on precision alone may result in missing many actual frauds. By considering ROC AUC and Average Precision, we ensure the evaluation captures both the model's ability to detect rare events and its tendency to avoid false alarms.

3.4 Machine Learning OCC Models

3.4.1 Choice of Models

To tackle the highly imbalanced nature of the IEEE-CIS fraud detection problem, several state-of-the-art One-Class Classification (OCC) algorithms were implemented. Each model offers a distinct theoretical approach to anomaly detection:

- **One-Class SVM (OCSVM):** Learns the boundary of normal data in a high-dimensional space and detects anomalies as points lying outside this boundary.
- **Elliptic Envelope:** Fits a Gaussian distribution to the data and identifies anomalies based on Mahalanobis distance.
- **Isolation Forest:** Detects anomalies by isolating observations via random partitioning, effective for large and complex datasets.
- **Local Outlier Factor (LOF):** Scores each sample by comparing local density to that of its neighbors, flagging those with significantly lower density as outliers.
- **k-Nearest Neighbors (kNN) Anomaly Scoring:** Assigns an anomaly score based on the mean distance to the k nearest neighbors; higher distances suggest outliers.

3.4.2 Hyperparameter Selection (Grid Search)

Hyperparameters for each model were tuned via grid search on the validation set. After grid search, the final hyperparameters selected for each model are summarized in the table below.

Model	Final Parameters
One-Class SVM	nu=0.03, gamma=0.005, kernel=rbf
Elliptic Envelope	contamination=0.01, support_fraction=0.8
Isolation Forest	n_estimators=250, contamination=0.01, max_samples=auto, random_state=42
Local Outlier Factor	n_neighbors=20, contamination=0.01, novelty=True
k-Nearest Neighbors	n_neighbors=20, threshold_percentile=90

Table 3.4: Final Parameters Selected for Each OCC Model (after Grid Search)

3.4.3 Training Protocol

All models were trained exclusively on non-fraud (normal) transactions to learn the typical distribution of legitimate data.

- **Training Set:** Only samples labeled as `isFraud = 0` were used for model fitting, aligning with the One-Class Classification paradigm.
- **Batching and Subsampling:**
  - For computational efficiency, models processed data in mini-batches of `50,000` samples during prediction.
  - For training, random subsampling of normal data was performed:
    - `OCSVM`: Up to `40,000` samples.
    - `LOF`: Up to `50,000` samples.
    - `Isolation Forest`: Used up to `20,000` samples (`max_samples=20000`).
- **Reproducibility:** All subsampling operations used `random_state = 42` to ensure consistent results across runs.

3.5 Deep Learning OCC Models

3.5.1 Model Architectures

Two deep learning models were investigated for one-class anomaly detection:

- **Autoencoder:** A symmetric feed-forward neural network trained to reconstruct the input data, using normal (non-fraud) transactions only. Anomalies are flagged by high reconstruction error.
- **Deep SVDD:** A deep neural network that learns to map normal data close to the center of a hypersphere in feature space. Transactions with large distances from the center are considered anomalies.

3.5.2 Training Protocol

- Both models were trained exclusively on normal data, following the OCC paradigm.
- Data was split: 90% for training, 10% for validation (from normal class).
- Full data (including fraud) was used for evaluation.
- Training lasted 50 epochs with batch size 256 (Autoencoder) or custom loop (Deep SVDD).

3.5.3 Scoring and Thresholds

- **Autoencoder:** Anomaly score is the mean squared error (MSE) between input and reconstruction. Threshold: 95th percentile of train errors.
- **Deep SVDD:** Anomaly score is the L2 distance to the learned hypersphere center. Threshold: 95th percentile of train distances.

3.6 Results and Discussion

3.6.1 Model Performance Summary

Model	Accuracy	Precision	Recall	F1	ROC AUC	Avg Precision
Isoforest	0.959504	0.285336	0.111018	0.159845	0.704483	0.115213
OCSVM	0.942988	0.179958	0.180779	0.180368	0.643679	0.112419
Elliptic	0.957541	0.164062	0.054602	0.081935	0.675256	0.083476
KNN	0.887478	0.110888	0.319564	0.164645	0.655329	0.121781
LOF	0.957032	0.102472	0.030710	0.047258	0.643076	0.059736
Deep SVDD	0.9257	0.1512	0.2472	0.1876	0.6769	0.1082
Autoencoder	0.9253	0.1462	0.2380	0.1812	0.6563	0.1079

Table 3.5: Performance of Classical and Deep Learning OCC Models

3.6.2 Result Interpretation

Table 3.5 highlights the ROC AUC and Average Precision (AP) performance of various classical and deep learning one-class classification (OCC) models in detecting fraud.

- **Deep Learning Models Outperform Classical Models on Discrimination Metrics:** Deep SVDD achieved the highest ROC AUC of 0.677 and AP of 0.108, followed closely by the Autoencoder with ROC AUC of 0.656 and AP of 0.108. These results suggest that deep models are better at ranking instances by their likelihood of being anomalous, despite not being trained on fraudulent samples.
- **Classical Models Showed Limited Discriminative Power:** Among classical OCC methods, Isolation Forest had the strongest ROC AUC (0.704) and AP (0.115), slightly outperforming even the deep models. However, this came with a trade-off in recall. kNN also performed well with ROC AUC of 0.655 and AP of 0.121, suggesting that simpler distance-based methods can still be competitive. Other methods like OCSVM (0.643/0.112) and Elliptic Envelope (0.675/0.083) showed moderate scores, while LOF lagged behind (0.643/0.060).

- **General Observation:** Despite substantial feature engineering and model tuning, no model crossed the 0.70 ROC AUC threshold consistently. Even the best-performing methods yielded low AP values (0.12), indicating difficulty in identifying rare fraudulent cases with high confidence. This highlights the limitations of unsupervised or OCC methods when dealing with highly imbalanced, overlapping, and unlabeled fraud detection tasks.

3.6.3 Visualization

Confusion Matrix of Classic Detection Models

Each confusion matrix includes:

- True Negatives (TN): Normal correctly predicted as Normal
- False Positives (FP): Normal incorrectly predicted as Fraud
- False Negatives (FN): Fraud incorrectly predicted as Normal
- True Positives (TP): Fraud correctly predicted as Fraud

K-Nearest Neighbors (KNN) and Local Outlier Factor (LOF)

KNN:

- High True Negative rate shows it performs well on normal instances.
- Decent True Positive count ($TP = 6514$), indicating it detects some fraud cases.
- Moderate False Positive ($FP = 52230$) and False Negative ($FN = 13870$), suggesting a trade-off.

LOF:

- Very low False Positive rate ($FP = 5483$), meaning it rarely mistakes normals for frauds.
- However, it misses many actual frauds ($FN = 19758$) with a low True Positive rate.
- Conservative model with high precision but low recall for fraud detection.

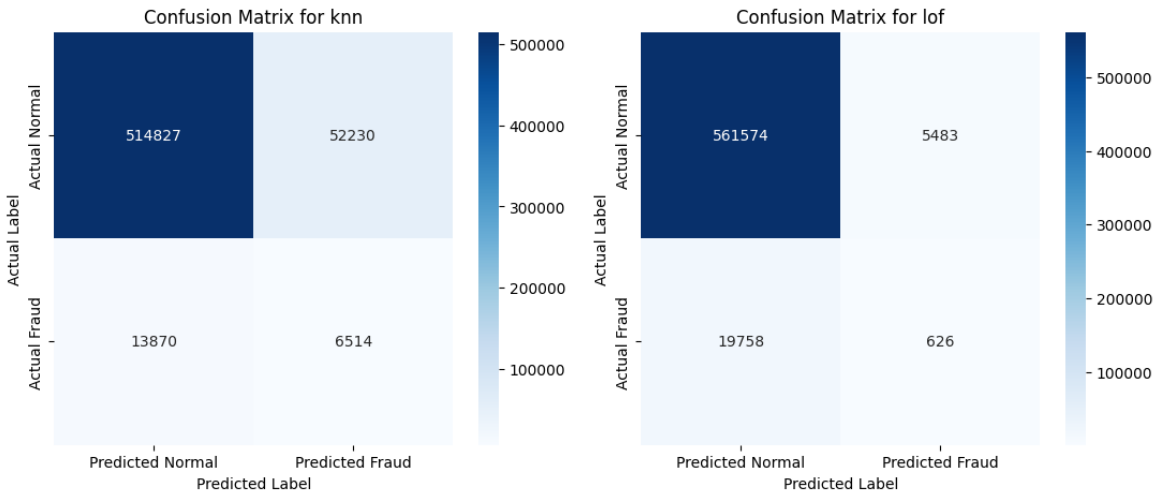


Figure 3.16: Confusion Matrix for **KNN (K-Nearest Neighbors)** and **LOF (Local Outlier Factor)**

Isolation Forest and Elliptic Envelope

Isolation Forest:

- Maintains high True Negative count similar to LOF.
- Better True Positive count ($TP = 2263$) than LOF, meaning better fraud detection.
- Balanced between False Positive and False Negative.

Elliptic Envelope:

- Similar performance to Isolation Forest in TN and FP.
- Fewer frauds correctly identified ($TP = 1113$), indicating worse recall than Isolation Forest.
- Prioritizes minimizing False Positives over catching fraud.

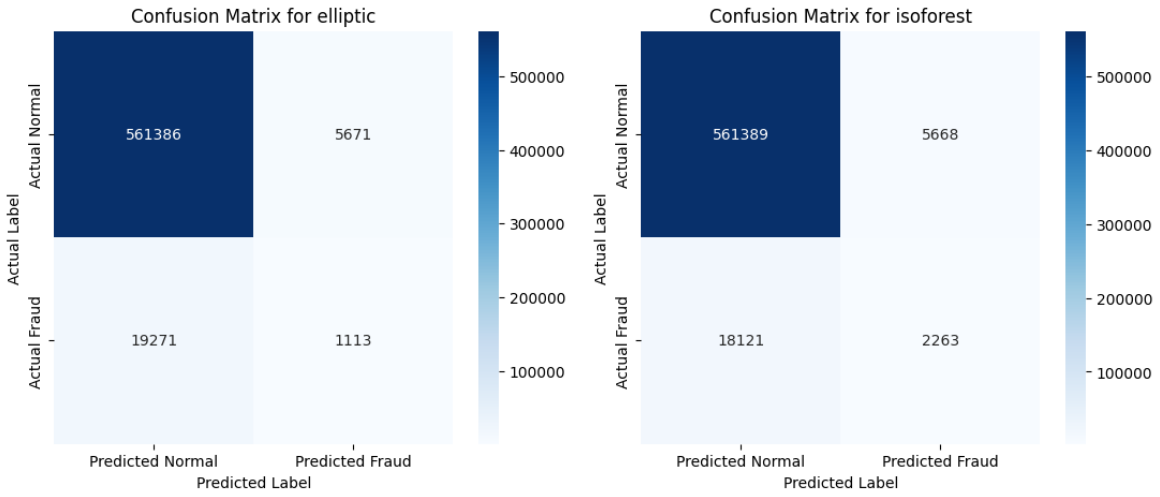


Figure 3.17: Confusion Matrix for **elliptic** and **isoforest**

One-Class SVM

- More aggressive model: detects more frauds ($TP = 3685$) than Elliptic and LOF.
- However, suffers from higher False Positive rate ($FP = 16792$).
- Suitable when detecting fraud is more important than avoiding false alarms.

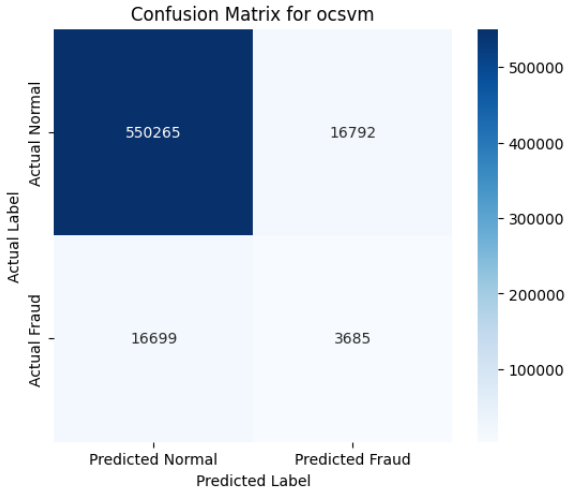


Figure 3.18: Confusion Matrix for **OCSVM (One-Class Support Vector Machine)**

Deep Learning Models

Target score of this problem are demonstrated via ROC curves for Deep Learning Models

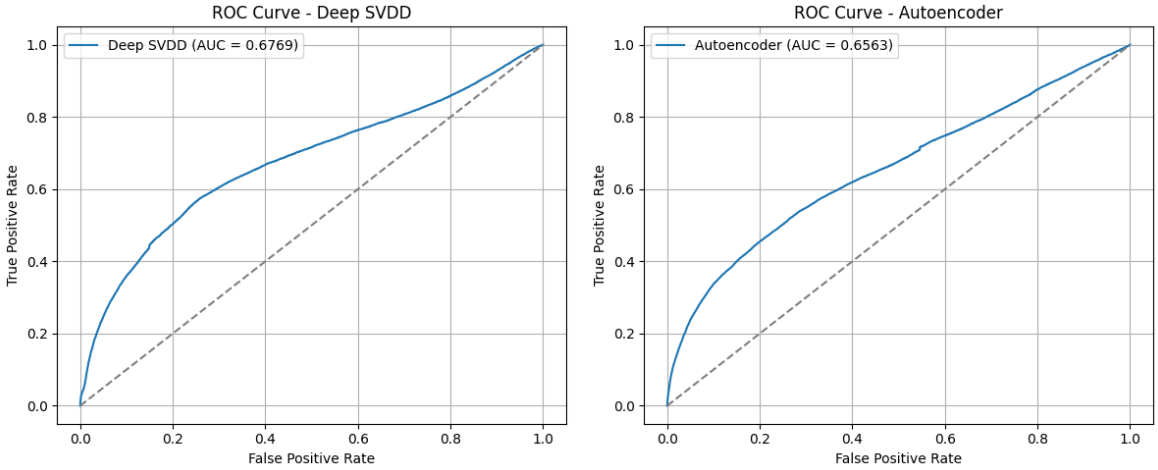


Figure 3.19: ROC Curves for Deep SVDD (left) and Autoencoder (right) on the validation set.

3.7 Extension: GAN-based Enhancement

To further tackle the severe class imbalance in the IEEE-CIS Fraud Detection dataset and enhance model robustness, we implemented a Generative Adversarial Network (GAN) to synthesize realistic fraudulent transaction data. This extension is motivated by the observation that fraud samples represent only $\sim 3.5\%$ of the dataset, significantly limiting the learning capacity of supervised and one-class models when attempting to generalize anomaly boundaries.

3.7.1 Architecture Design

The GAN was designed with a simple feedforward architecture:

- **Generator**: Takes a latent vector of dimension 32 and produces synthetic fraud samples across 21 features (including transaction-related and engineered PCA features).
- **Discriminator**: A binary classifier that attempts to distinguish between real fraud transactions and those synthesized by the Generator.

We selected 21 features for the input space, consisting of:

- **Strong EDA-driven features**: TransactionAmt, dist1, C3, C7, C8, D1, D10.
- **PCA-compressed versions of anonymized variables**: 14 dimensions (0 through 13), representing reduced V-columns.

Data was standardized using StandardScaler, and only real fraud samples were used to train the GAN, following a one-class generation scheme.

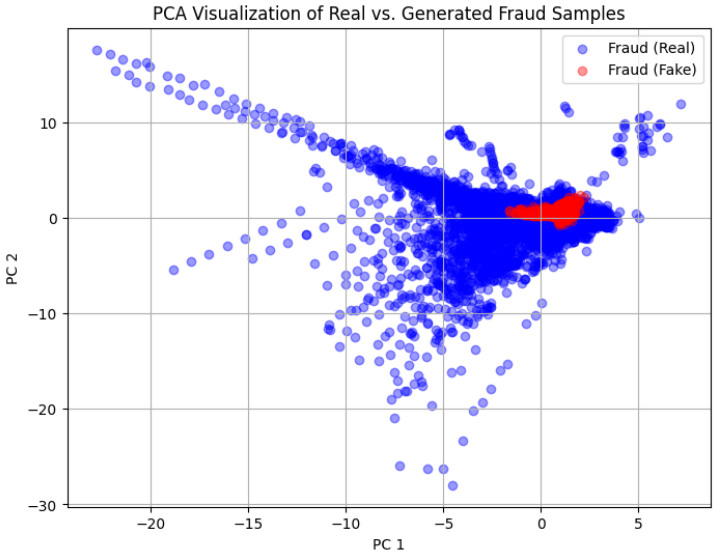
3.7.2 Training Protocol

The GAN was trained for 50 initial epochs, followed by a second training phase of an additional 50 epochs to refine the generator's performance. The training objective was to minimize the adversarial loss for both the Generator and Discriminator using binary cross-entropy. During training, we observed stable convergence with no sign of mode collapse, and the Generator increasingly produced realistic outputs.

3.7.3 Evaluation of Generated Samples

We assessed the quality of generated fraud samples using three complementary methods:

- **Discriminator Score**: The discriminator assigned a mean confidence of 0.31 to fake samples versus 0.67 to real fraud data, indicating moderate realism. Notably, some fake samples achieved confidence scores up to 0.97, confirming that the generator could convincingly imitate real fraud patterns.
- **PCA Visualization**: A 2D PCA projection revealed that synthetic samples cluster around the densest regions of real fraud samples, overlapping significantly in the central distribution. However, fake data lacked coverage in the periphery, suggesting limited diversity and poor outlier modeling.
- **External Classifier Evaluation**: A logistic regression classifier was trained to distinguish real vs. fake fraud. The model achieved 85.4% F1-score on fake data and 99.3% on real fraud. This indicates that while some fake data can be easily detected, a meaningful proportion (20%) is indistinguishable from real samples—sufficient for augmenting the minority class in downstream models.



3.7.4 Implications and Future Use

The GAN module proves effective in:

- Enhancing data diversity for the rare fraud class.
- Producing high-quality synthetic samples suitable for classifier augmentation or anomaly boundary reinforcement.
- Complementing One-Class Classification models by generating informative hard-negative samples

Chapter 4

Conclusion

In this report, we investigated the effectiveness of One-Class Classification (OCC) methods in detecting fraud from the IEEE-CIS dataset. The highly imbalanced nature of the data, with only a small portion of fraudulent transactions, necessitated a focus on modeling normal behavior and identifying anomalies as deviations from this norm.

Our experiments showed that deep learning models such as Autoencoders and Deep SVDD provide moderate improvements in ROC AUC and Average Precision over classical OCC models like Isolation Forest and One-Class SVM. However, all models demonstrated limitations in distinguishing fraudulent cases due to the substantial feature overlap between normal and anomalous transactions.

The integration of GANs to generate synthetic fraud samples offered a promising augmentation strategy, producing reasonably realistic samples that can potentially support hybrid or semi-supervised learning approaches.

Despite these efforts, model performance remained modest. Key challenges include the lack of supervised fraud labels during training, high-dimensional feature noise, and non-stationarity in real-world transaction data.

Future work should explore semi-supervised learning, continual learning, or domain-specific constraints to better capture the complexity of fraud behavior. Moreover, incorporating explainability and ethical considerations will be crucial for real-world deployment of such systems in financial domains.

References

[1] YouTube. *[Video] Anomaly Detection in Cybersecurity (YouTube ID: 7HroYexkvXs)*. Retrieved from https://youtu.be/7HroYexkvXs, Accessed: June 11, 2025.

[2] YouTube. *[Video] One-Class SVM Explained (YouTube ID: 3YN4487laXo)*. Retrieved from https://youtu.be/3YN4487laXo, Accessed: June 11, 2025.

[3] YouTube. *[Video] Isolation Forest - Machine Learning Algorithm (YouTube ID: 5YlhdWzltM4)*. Retrieved from https://youtu.be/5YlhdWzltM4, Accessed: June 11, 2025.

[4] YouTube. *[Video] Autoencoders for Anomaly Detection (YouTube ID: CfHnguQBm04)*. Retrieved from https://youtu.be/CfHnguQBm04, Accessed: June 11, 2025.

[5] YouTube. *[Video] LSTM for Anomaly Detection (YouTube ID: ocgHIMLvU54)*. Retrieved from https://youtu.be/ocgHIMLvU54, Accessed: June 11, 2025.

[6] YouTube. *[Video] One-Class Classification Practical Guide (YouTube ID: qb62ifIxDbY)*. Retrieved from https://youtu.be/qb62ifIxDbY, Accessed: June 11, 2025.

[7] Brownlee, Jason. *One-Class Classification Algorithms for Imbalanced Datasets*. Machine Learning Mastery, 2020. Retrieved from https://machinelearningmastery.com/one-class-classification-algorithms/, Accessed: June 11, 2025.

[8] Khan, Shehroz S., and Michael G. Madden. "One-Class Classification: Taxonomy of Study and Review of Techniques." *ResearchGate* (2013). Retrieved from https://www.researchgate.net/publication/259044274_One-Class_Classification_Taxonomy_of_Study_and_Review_of_Techniques, Accessed: June 11, 2025.

[9] Perera, Pramuditha, Poojan Oza, and Vishal M. Patel. "One-Class Classification: A Survey." *Applied Sciences*, vol. 13, no. 1, 2023, p. 259. Retrieved from https://www.mdpi.com/2076-3417/13/1/259, Accessed: June 11, 2025.

[10] Carnegie Mellon University. *Insider Threat Test Dataset*. CMU KiltHub, 2016. Retrieved from https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247/1, Accessed: June 11, 2025.