Fraud detection for credit card transactions

Mathis Dumont, Pierre Chuzeville
ENSAE, Polytechnic Institute of Paris

January 18, 2026

ABSTRACT

This report investigates methodologies for fraud detection in highly imbalanced credit card transaction datasets. We explore the theoretical implications of undersampling bias, evaluate classical supervised classifiers (Logistic Regression, XGBoost), and implement an unsupervised anomaly detection framework using autoencoders. Finally, we propose a cost-sensitive thresholding strategy to align model performance with economic utility.

1. INTRODUCTION

A. Dataset description

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly imbalanced, the positive class (frauds) account for **0.173%** of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, it cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'.

The feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. The feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

B. Objectives

The primary challenge lies in managing the extreme class imbalance. Traditional accuracy metrics are insufficient in this context, as a naive model predicting the majority class would yield 99.8% accuracy while failing to detect any fraud.

2. STRATEGIES AND CHALLENGES

Our objective is to perform binary classification on an imbalanced dataset. To address the class imbalance, we initially focus on undersampling. This method involves rebalancing the training data to ensure an equal number of positive and negative samples (basically by down-sizing the majority class, in our case, the negative one).

Classical supervised learning relies on the fundamental assumption that the training and testing data are drawn from the same probability distribution. However, by undersampling this assumption is violated and sampling selection bias will be created [3].

A. Undersampling selection bias

In this section we will explain how sampling selection bias occurs when undersampling a skewed training set.
Let's consider the binary classification set up $f : \mathcal{X} \mapsto \mathcal{Y}$ with $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} = \{0, 1\}$ and assume that the negative class is overly represented in $\mathcal{X}$ compared to the positive class. Let $(X, Y) \subset (\mathcal{X}, \mathcal{Y})$ denote the balanced sample obtained by undersampling.
We define s as a random variable selection which takes value 1 if the point is inside (X, Y) and 0 otherwise. We assume that this selection variable s is independent of the input x given y (that is a legitimate assumption, in practice we operate the undersampling by only caring about the y label, not the other features), so $\mathbb{P}(s|x, y) = \mathbb{P}(s|y)$. This implies $\mathbb{P}(x|y, s) = \mathbb{P}(x|y)$, that means, by randomly removing data the within-class distribution does not change. With undersampling we change the prior distribution $\mathbb{P}(y|x, s) \neq \mathbb{P}(y|x)$.
From Bayes' rule we can write, $p_s := \mathbb{P}(y = 1|x, s = 1)$:

$$p_s = \frac{\mathbb{P}(s = 1|y = 1)\mathbb{P}(y = 1|x)}{\mathbb{P}(s = 1|y = 1)\mathbb{P}(y = 1|x) + \mathbb{P}(s = 1|y = 0)\mathbb{P}(y = 0|x)}.$$

As $\mathbb{P}(s = 1|y = 1) = 1$ (we select every positive instances), finally we obtain:

$$p_s = \frac{\mathbb{P}(y = 1|x)}{\mathbb{P}(y = 1|x) + \mathbb{P}(s = 1|y = 0)\mathbb{P}(y = 0|x)}. \quad (1)$$

Let us denote $\beta := \mathbb{P}(s = 1|y = 0)$, the probability of selecting a negative instances during the sampling. In practice, to obtain a perfectly balanced training set, we set this probability to match the undersampling ratio $\hat{\beta} := \frac{N^+}{N^-}$ where N^+, N^- denote the number of positive, resp negative, individuals in our dataset. If we write $p := \mathbb{P}(y = 1|x)$, it gives :

$$p_s = \frac{p}{p + \beta(1 - p)}. \quad (2)$$

By reversing this equation we can express p according to p_s:

$$p = \frac{\beta p_s}{\beta p_s - p_s + 1} \quad (3)$$

Equation (3) explains how the distribution of the balanced sample relates to the distribution in the original unbalanced sample. In the next section, we will propose a way to recover a meaningful result for our classification problem after the undersampling of our training set.

B. Bias correction

Although a classification model is more easily learned from a balanced training set, it is ultimately deployed to predict transactions in a real-world environment where the

distribution remains highly skewed. Training on a 50/50 subsample introduces a systematic bias: the model learns the posterior probability $p_s = \mathbb{P}(y = 1|x, s = 1)$ instead of the true probability $p = \mathbb{P}(y = 1|x)$. To compensate for this, we perform probability calibration. In our study, with $\hat{\beta} \approx 0.173\%$, any raw score produced by the model would significantly overestimate the fraud risk. By applying the inverse transformation derived in Equation (3), we can map the "downsampled" scores back to the original probability space. This adjustment is crucial for operational decision-making, ensuring that the predicted risk scores reflect the true rarity of fraud and allowing for a meaningful comparison with business cost thresholds.

3. PERFORMANCE METRICS AND MODEL EVALUATION

In the context of extreme class imbalance, traditional accuracy becomes a deceptive metric. A "zero-fraud" classifier, which naively predicts the majority class for every transaction, would achieve an accuracy of 99.8% while being economically useless. Consequently, model evaluation must shift toward metrics that decouple performance on the minority class from the majority.

A. Precision, Recall and the F_β-score

We rely on the confusion matrix to derive two fundamental metrics: **Precision**, which measures the reliability of a fraud alert, and **Recall**, which measures the proportion of total frauds successfully captured by the model.

In credit card fraud detection, the operational priority is typically tilted toward Recall; missing a single high-value fraudulent transaction is often more costly than the administrative friction caused by a few false positives. To quantify this trade-off, we utilize the generalized F_β-score:

$$F_\beta\text{-score} = \frac{(1 + \beta^2) \cdot \text{precision} \times \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}. \tag{4}$$

As noted by Van Rijsbergen [4], the F_β metric measures effectiveness by attaching β times as much importance to recall as to precision. (Note that the β parameter in the F-score is a weight for recall importance, distinct from the sampling ratio defined in Section 2.) In our study, we will explore F_1 (balanced) and F_2 (recall-oriented) scores to align the model with a practical security standpoint.

B. Business-Driven Evaluation

While statistical metrics provide a standardized benchmark, the final validation of a fraud detection system must be governed by the asymmetric costs of classification errors. The last section of this report will demonstrate how to determine the optimal decision threshold by minimizing a concrete financial cost function, effectively bridging the gap between machine learning performance and economic utility.

4. SUPERVISED LEARNING BENCHMARKING

We evaluate three standard classifiers: Logistic Regression (LR), Random Forest (RF), and XGBoost. To facilitate learning, we train these models on a manually balanced subsample ($N^+ = N^-$), while evaluating performance on the original imbalanced test set to reflect real-world conditions.

To address the resulting distribution shift, instead of analytically recalibrating the probabilities (as discussed in Section 2) or relying on a default 0.5 threshold, we adopted a data-driven threshold optimization. By analyzing the Precision-Recall curve, we identified the optimal threshold τ that maximizes the F1-score for each model. This approach effectively aligns the decision boundary with the model's output distribution without requiring explicit probability transformation.

A. Comparative Analysis

The results in Table 1 illustrate the classic trade-offs associated with undersampling. The Logistic Regression model achieves a high Recall (∼92%), identifying 90 out of 98 frauds, but suffers from extremely low Precision (∼4%), generating over 2,200 false positives. This occurs because undersampling removes the "texture of normality," causing the linear model to over-classify atypical but legitimate transactions as fraud.

While Random Forest improves the F1-score to 0.34, XGBoost emerges as the superior supervised classifier ($F_1 = 0.90$). By capturing non-linear relationships and utilizing gradient boosting, XGBoost drastically reduces false alarms (only 2 FPs) while maintaining robust detection.

Table 1: Supervised Models Comparison (Full Test Set)

Model	TN	FP	FN	TP	F1
Log. Reg.	55,475	1,388	8	90	0.11
Rand. Forest	56,861	3	24	74	0.85
XGBoost(optim)	56,857	6	17	81	0.88

B. Precision-Recall Dynamics

The Precision-Recall curve for XGBoost (Figure 1) reveals three distinct operational zones. In the *High-Confidence Zone* (Recall < 60%), precision remains near 1.0, catching "obvious" frauds with zero false alarms. The *Optimal Zone* (Recall 60%–85%) maintains precision above 0.80, offering the best balance for production deployment. Finally, the *Saturation Zone* (Recall > 85%) shows a sharp precision drop; at this stage, fraudulent patterns become statistically indistinguishable from legitimate ones, making the marginal cost of full detection (in terms of false positives) economically prohibitive.

To address these limitations and detect novel fraud types, we next explore an unsupervised framework using Autoencoders.

5. AUTOENCODER FOR FRAUD DETECTION

A. Why Autoencoders? Brief explanation

Autoencoders excel at anomaly detection in an unsupervised manner, meaning we don't need labeled "fraud" examples to train them.

Firstly, autoencoder models learn "normality": we train an autoencoder only on normal (non-fraudulent) transactions, the model actually learns to compress these normal transactions into a small summary (**latent space**) and then reconstruct them perfectly.

The main idea of their ability to detect frauds is that autoencoders fail to reconstruct anomalies. When a fraudulent transaction comes in, it looks different from the nor-

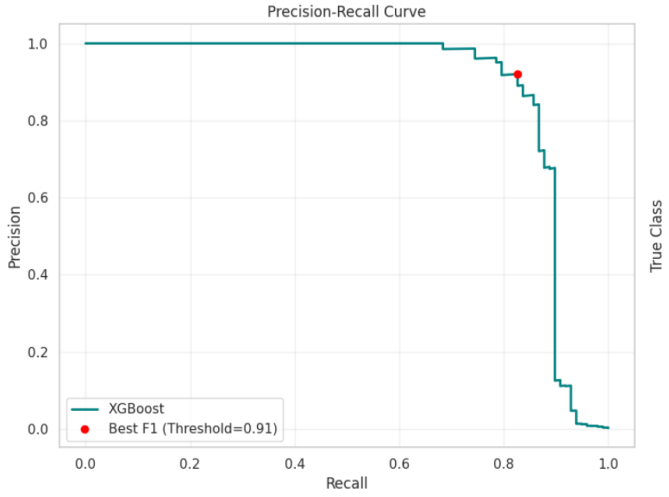

Figure 1: Precision-Recall curve for XGBoost

mal data the model has seen, so the model tries to compress and reconstruct it using the rules it learned for normal data, but it fails.
The error is the signal, because the model fails to reconstruct the fraud, the difference between the input and the output (the **reconstruction error**) is huge.
Autoencoders demonstrate robust performance in unsupervised anomaly detection, making them ideal for identifying zero-day fraud attacks. Unlike supervised models (such as XGBoost) that rely on labeled data, autoencoders are particularly effective at detecting anomalies indicative of new fraud methodologies. The architecture of autoencoders allows them to generalize better to unseen distributions, thereby improving detection rates for unknown fraud types.

B. Baseline Autoencoder structure and first results

We first present the baseline structure for our autoencoder implementation. Then we will discuss improving it, switching activation functions and adding regularization (dropouts).

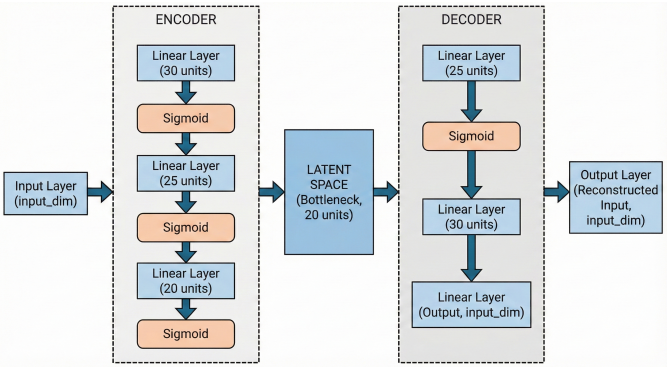

Figure 2: Baseline autoencoder structure

For this autoencoder we set : input_dim = size of the training set's normal transaction. After normalization, we trained this autoencoder using the following hyperparameters:
- num_epochs = 50
- learning_rate = 0.001
- criterion = MSE
- optimizer = Adam
- patience = 5 : we added a threshold to stop the learning process if it doesn't improve for 5 successive epochs.

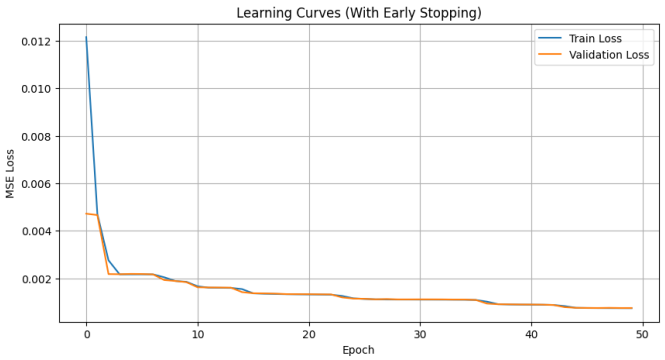

Figure 3: Learning curve of our autoencoder

It seems the learning works well (50 epochs did not trigger the early stopping). The most positive signal is how closely the Validation Loss tracks the Train Loss, it probably means that there is no overfitting.
The sharp drop in the first 2-3 epochs indicates the model quickly found the most obvious patterns in the data. This suggests the learning rate is likely appropriate. While this graph looks great for training, in the specific context of fraud Detection, we must verify one potential issue:

If the reconstruction error is extremely low for everything, the model might be acting as an Identity Function, simply copying the input to the output perfectly without learning meaningful features. Then the model won't be able to detect fraud, because the reconstruction error will be low for both fraud and non-fraud.

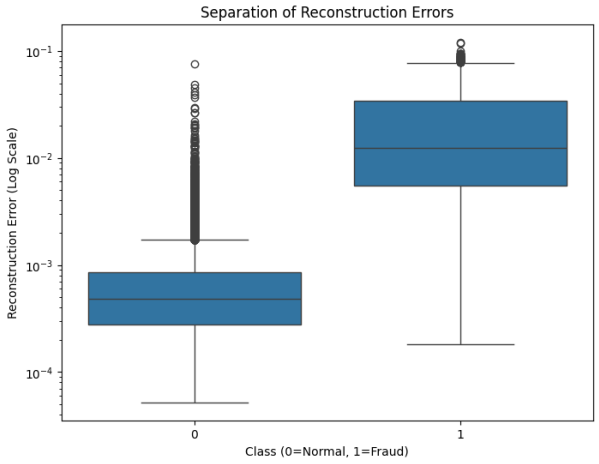

Figure 4: Reconstruction error by class for baseline AE

The distribution of reconstruction errors (Figure 4) validates the anomaly detection hypothesis, showing a two-order-of-magnitude separation between classes.
This means the model clearly "understands" normal data and is "surprised" by the fraud data.

Table (2) shows that even if the F1-score is not very high, the baseline autoencoder managed to detect 187 over 246 frauds in the test set, with only 130 false positives.
The next step of our work was to look at a way to improve this autoencoder, especially we are interested in improving the speed of our learning (which, in a practical stand point could be even more important that perfectly accurate detection). To achieve this, we implemented regularization

Table 2: Confusion Matrix Baseline AE

	Predicted Class	
Actual Class	**Normal (0)**	**Fraud (1)**
Normal (0)	28,302 (TN)	130 (FP)
Fraud (1)	59 (FN)	187 (TP)

F1-Score: 0.66

(by adding dropouts) and switched activation functions to ReLu (only for the internal 'hidden' layers). Indeed sigmoid function requires calculating an exponential function (e^{-x}), which is expensive to compute for the processor while ReLU is a simple threshold operation: $\max(0, x)$. It is mathematically much simpler and faster to compute, speeding up training times.

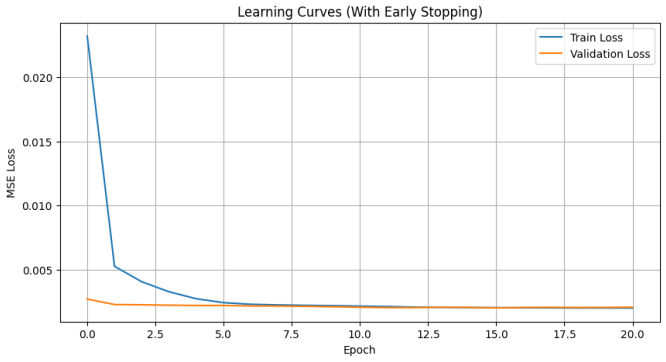

Figure 5: Learning curve with ReLu activation and Dropouts

This time the early stopping triggers after 13 epochs, as expected. This autoencoder regularized is much faster than the baseline autoencoder. Let us compare how it performs.

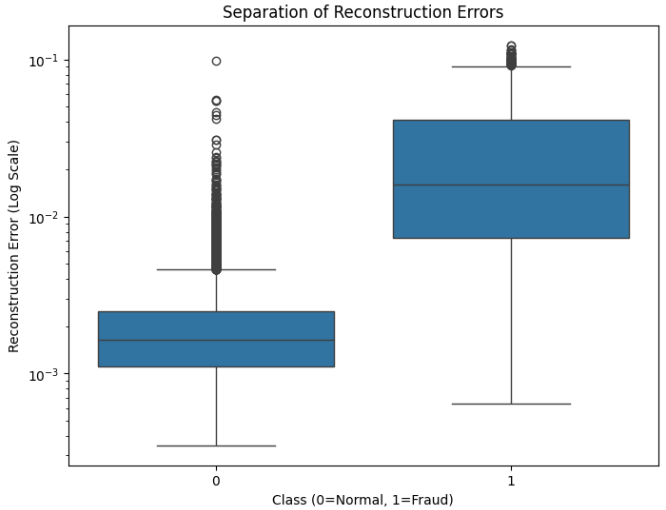

Figure 6: Reconstruction error by class (ReLu + Dropout)

From Figure 6 we observe that:
For normal class (0): The median is roughly around 10^{-3} to 2.5×10^{-3}.
For fraud class (1): The median error is significantly higher (around 1.5×10^{-2}), and the bulk of the distribution sits higher than the normal class. This confirms that the model finds fraudulent patterns "unfamiliar" and thus cannot reconstruct them accurately. However, we note that the gap is not as big as our first baseline autoencoder. It means, by learning faster we lose a bit of power.
Now let us consider the confusion matrix.

Table 3: Confusion Matrix ReLU + Dropout

	Predicted Class	
Actual Class	**Normal (0)**	**Fraud (1)**
Normal (0)	28,371 (TN)	61 (FP)
Fraud (1)	97 (FN)	149 (TP)

F1-Score: 0.65

As we can see the classification, in terms of F1-score is almost the same as the Baseline Autoencoder, this Regularized Autoencoder detected less frauds but had very few false positives (61). As it's much faster to learn (around 5 times faster than the baseline AE), we will keep discussing about this one (Autoencoder ReLu + Dropout) for the following sections. Now we discuss the lift curve of this model (Figure 7).

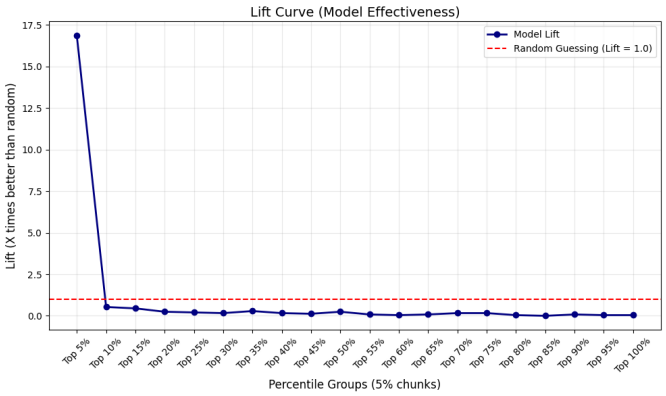

Figure 7: Lift curve ReLu + Dropout AE

The "Top 5%" Peak ($\sim 16.8\times$) is an interesting result. This means that if we look at the 5% of transactions with the highest reconstruction error (the most "anomalous"), the model finds fraud nearly 17 times more often than if we just picked transactions at random.

The curve drops drastically after the Top 5% and stays near the "Random Guessing" line (1.0). It tells us that the model has successfully concentrated almost all the fraud into the very top tier of reconstruction errors.
In concrete fraud detection, time is money, this chart tells an analyst: "Don't waste time looking at the whole dataset. Just look at the top 5% of alerts, because that's where $17\times$ the fraud is hiding."

6. IDEAS TO GO FURTHER INTO PRACTICAL FRAUD DETECTION

A. Adjusting the threshold

One way to improve the recall is to adjust the threshold (see section 3.), at this point we only treat the F1-score as our way to evaluate model but it's supposed that we consider at the same level recall and precision. It could be interesting to put more importance to the recall, for ex-

ample by considering F2-score. Let us show the confusion matrix from our last autoencoder model optimizing this F2-score :

Table 4: Confusion Matrix AE, maximizing F2

	Predicted Class	
Actual Class	**Normal (0)**	**Fraud (1)**
Normal (0)	28,207 (TN)	225 (FP)
Fraud (1)	71 (FN)	175 (TP)

F2-Score: 0.63

As we can see, the model detected more frauds (compared to 3). In the same time, it classifies more normal transactions as frauds.
At this point, it is a matter of business choice, we will discuss this part in the next part.

B. Adjusting threshold according to concrete cost prediction

In a practical financial environment, the performance of an anomaly detection model is not merely a matter of statistical accuracy but of economic utility, where the trade-off between sensitivity and specificity represents a direct conflict between the cost of fraud and the cost of operational friction. The financial impact of classification errors in banking is inherently asymmetric, comprising the cost of false negatives (C_{FN}), representing missed fraudulent transactions, and the cost of false positives (C_{FP}), representing false alarms. To operationalize this evaluation, we adopt a cost-minimization framework; although the features in our dataset have been obfuscated via Principal Component Analysis (PCA) preventing the direct use of transaction amounts as cost variables, we model the system using representative average cost estimates.
To highlight the strategic implications of threshold selection, we assign the cost weights $C_{FN} = 500$ and $C_{FP} = 10$, creating a significant gap that reflects the strategic priority of financial institutions to favor recall over precision. These parametric values are intended to be adjustable.
The objective is to identify an optimal threshold τ that minimizes the Total Expected Cost, defined as:

$$\text{Total Expected Cost}(\tau) = FN(\tau) \cdot C_{FN} + FP(\tau) \cdot C_{FP} \tag{5}$$

.

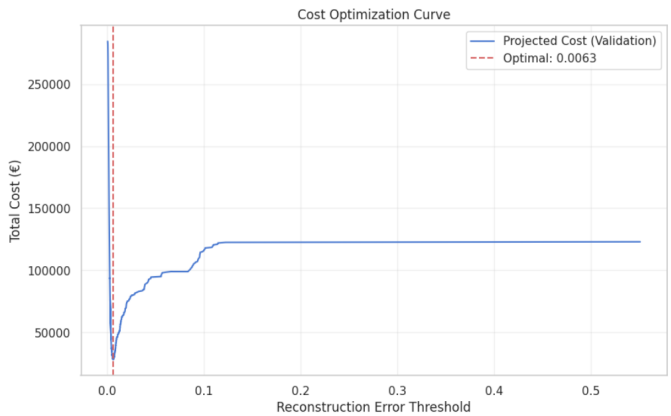

Figure 8: Cost-sensitive strategy, the optimal threshold

As illustrated in Figure 8, the calculated optimal threshold is $\tau \approx 0.0063$.
This threshold is notably low, reflecting the heavy emphasis placed on the detection of fraudulent activities. To examine the new classification and error distribution, the corresponding confusion matrix is presented below.

Table 5: Confusion matrix, minimizing cost method

	Predicted Class	
Actual Class	**Normal (0)**	**Fraud (1)**
Normal (0)	27,986 (TN)	446 (FP)
Fraud (1)	46 (FN)	200 (TP)

As we can see from Table (5) by optimizing the cost, the model detects 200 over the 246 tested frauds, but employees (or other adapted models) will have to deal with 446 transactions classified as frauds which are not: that is the optimal cost to pay.
In our situation we also compute the lowest projected loss : €28,180.

7. CONCLUSION

This study demonstrates that detecting credit card fraud requires a multi-faceted approach, balancing statistical rigor with economic constraints. While classical supervised models like XGBoost achieve high precision by leveraging historical patterns, unsupervised frameworks such as Autoencoders provide a robust alternative for detecting novel or evolving fraud methodologies.

Our findings highlight that the "best" model is not necessarily the one with the highest accuracy, but the one that best aligns with an institution's risk appetite. By shifting the evaluation from standard metrics to a cost-minimization framework, we transitioned from a purely mathematical problem to a decision-support tool.

Ultimately, the choice of the decision threshold τ remains a strategic lever. A lower threshold, as suggested by our cost-optimization analysis, prioritizes security and recall, accepting higher operational friction (false positives) to mitigate the catastrophic financial and reputational risks associated with missed fraud. Future work could integrate transaction amounts directly into the loss function to further refine the economic utility of these models.

REFERENCES

[1] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. *Calibrating Probability with Undersampling for Unbalanced Classification.* In Symposium on Computational Intelligence and Data Mining (CIDM) IEEE, 2015.

[2] Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. *Learned lessons in credit card fraud detection from a practitioner perspective.* Expert Systems with Applications, 41(10):4915–4928, 2014.

[3] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning.* The MIT Press, 2009.

[4] Van Rijsbergen, C. J. *Information Retrieval* (2nd ed.). Butterworth-Heinemann, 1979