

Transaction Fraud Detection Using Machine Learning

T R Arunkumar¹; Ankita Kulkarni²; Raghavendra Mudhol³; Pratiksha Marathe⁴

¹Assistant Professor: Dept of computer science school of mathematics and computing science Rani Channamma University, Belagavi, India

²Lecturer: Dept of computer science SRFGCC, Belagavi, India

^{3,4}Dept of computer science, SRFGCC Belagavi

ABSTRACT

An The exponential growth of digital financial transactions — spanning banking portals, e-commerce platforms, and mobile payment systems — has brought with it an equally sharp rise in fraudulent activity. Every year, financial fraud drains billions of dollars from the global economy, and the challenge of detecting it accurately and quickly has never been more pressing. Conventional rule-based detection mechanisms, which rely on manually coded threshold conditions, have largely failed to keep pace with the adaptability of modern fraudsters. What is needed instead is a system that can learn from data, recognise subtle patterns, and update its understanding as new fraud tactics emerge.

This paper presents the design and implementation of a Transaction Fraud Detection System — referred to throughout as Fraud AI — which applies a Random Forest ensemble classifier to identify fraudulent financial transactions in near real time. The model is trained on the PaySim synthetic dataset, a publicly available simulation of mobile money transactions containing over 6.3 million records generated to reflect one month of activity. Before training, the data undergoes a structured preprocessing pipeline that corrects for severe class imbalance using the Synthetic Minority Over-sampling Technique (SMOTE), derives informative balance-error features, and encodes categorical transaction types into a numeric format. The resulting model attains an overall classification accuracy of 82.2% and is served through a Flask-based RESTful web application capable of returning fraud labels, probability scores, and visual risk summaries in under 200 milliseconds. The work demonstrates that combining thoughtful feature engineering with appropriate class-imbalance correction allows a well-configured Random Forest to serve as a practical, scalable, and user-accessible fraud detection tool.

Keywords: Transaction Fraud Detection, Machine Learning, Random Forest, SMOTE, PaySim Dataset, Flask, Feature Engineering, Class Imbalance, Real-Time Classification, Financial Security.

I. INTRODUCTION

The Financial fraud has long been a serious concern for economies worldwide, but the shift to digital channels has dramatically expanded both its scale and complexity. The Association of Certified Fraud Examiners (ACFE), in its 2022 Report to the Nations, estimated that organisations globally lose around five percent of annual revenues to fraud — a figure that translates into trillions of dollars when considered at the macro level. Internet banking, mobile wallet transfers, point-of-sale terminals, and cryptocurrency exchanges have collectively created a vast and interconnected attack surface that skilled fraudsters know how to exploit with speed and relative anonymity.

Early attempts to control this problem leaned on rule-based detection systems. These tools flagged transactions that violated hard-coded thresholds — for instance, a transfer amount far above average, or a login originating from a geographically implausible location. Such systems have real advantages: they are easy to explain, quick to implement, and produce traceable audit trails. Yet they carry two critical weaknesses that have become increasingly apparent over time.

First, they generate a disproportionate number of false positives, wrongly classifying legitimate transactions as suspicious and creating friction for ordinary customers. Second, and more fundamentally, they are static by design. Once a fraud pattern is known and encoded as a rule, fraudsters can study the system, identify its boundaries, and craft transactions that fall just below every threshold. The rules never adapt; the fraud does.

Machine learning offers a fundamentally different approach. Instead of relying on human-defined conditions, a trained model learns decision boundaries directly from historical data, capturing complex, non-linear relationships that would be impractical to express as explicit rules. Among the many algorithms that have been applied to this problem — Logistic Regression, Decision Trees, Gradient Boosting, Support Vector Machines, and various deep learning architectures —

ensemble methods have consistently stood out. The Random Forest classifier, in particular, has earned strong empirical support across fraud detection studies. Its robustness against overfitting, its ability to process high-dimensional feature spaces, and the interpretability offered by feature importance rankings make it well suited to the demands of financial classification tasks.

A persistent obstacle in this domain, however, is the extreme imbalance between fraud and legitimate transactions. In most real-world datasets, fraudulent records make up considerably less than one percent of all entries. Training a classifier on such data without any correction tends to produce a model that simply predicts the majority class for every input — achieving superficially high accuracy while completely failing to catch any fraud. This problem demands deliberate intervention, and SMOTE has emerged as one of the most effective and widely adopted solutions.

The present study addresses these challenges by building an end-to-end Transaction Fraud Detection System. Working with the PaySim synthetic dataset, the research constructs a preprocessing and feature engineering pipeline, applies SMOTE to balance the training data, trains and evaluates a Random Forest classifier, and deploys the model as a Flask web application with an interactive frontend. The result is a complete, functional system that can classify individual transactions in real time and present the findings in a form that non-technical users can interpret and act upon.

II. Literature Review

Research into automated fraud detection has a history stretching back several decades, and the landscape has shifted considerably as computational power and data availability have grown. The early literature focused primarily on credit card fraud using relatively small, proprietary datasets, making it difficult to compare results across studies or to reproduce findings independently. That situation changed markedly in 2016 when Lopez-Rojas, Elmir, and Axelsson introduced the PaySim financial mobile money simulator. PaySim generates synthetic transaction datasets whose statistical properties closely mirror those of real-world mobile money activity, drawing on transaction logs obtained from a financial company operating in Africa. Because the data is synthetic, it can be shared freely without compromising customer privacy, enabling the kind of open, reproducible research that the field had long needed. The PaySim dataset quickly became a standard benchmark in fraud detection literature and serves as the primary data source for the present work.

The Random Forest algorithm was formally described by Breiman (2001), who demonstrated that combining the predictions of many independently trained decision trees through majority voting produces a classifier that is substantially more accurate and robust than any single tree. The key mechanism behind this improvement is variance reduction: individual trees may overfit to noise in their training samples, but averaging across many trees trained on different bootstrap samples smooths out those idiosyncratic errors. Breiman also showed that using only a random subset of features at each split point decorrelates the individual trees, further improving the ensemble's generalisation. These properties translate directly into advantages for fraud detection, where training data is noisy, feature spaces are wide, and generalisation to unseen fraud patterns is essential. Numerous subsequent studies confirmed that Random Forest consistently outperforms Logistic Regression, Support Vector Machines, and single decision trees on tabular financial datasets, and frequently matches or exceeds shallow neural networks on such tasks.

The class imbalance problem received a focused and influential treatment from Chawla, Bowyer, Hall, and Kegelmeyer (2002) with the introduction of SMOTE. Prior to SMOTE, the dominant approach to imbalanced data was either to undersample the majority class — discarding potentially useful information — or to oversample the minority class by duplicating existing samples, which adds no new information and risks overfitting. SMOTE instead generates entirely new synthetic minority class examples by interpolating between existing minority samples in the feature space. For each minority instance, the algorithm identifies its k nearest neighbours within the minority class and creates new examples at randomly chosen points along the line segments connecting the instance to those neighbours. This approach enriches the minority class's representation in the feature space without introducing redundancy, allowing a classifier to learn more robust decision boundaries around the fraud class. SMOTE has since become one of the most widely cited and adopted techniques in the imbalanced learning literature.

The practical implementation of these techniques was greatly facilitated by the development of scikit-learn, described by Pedregosa et al. (2011). Scikit-learn provides a consistent, well-documented Python API for an extensive range of machine learning algorithms, preprocessing utilities, and evaluation tools, along with tight integration with NumPy and Pandas. The imbalanced-learn extension to scikit-learn subsequently provided SMOTE and related techniques within the same API framework, making it straightforward to incorporate resampling into scikit-learn pipelines. These libraries form the technical backbone of the implementation described in this paper.

Géron (2022) offered practical engineering guidance in his widely used textbook, addressing feature engineering strategies, cross-validation methodology, hyperparameter tuning, and model deployment considerations. His treatment of ensemble methods and the practical challenges of imbalanced classification directly informed several design decisions in the present system. On the regulatory and contextual side, the Reserve Bank of India's 2023 Annual Report on Payment and Settlement Systems documented the rapid growth of digital payment volumes in India and highlighted the heightened fraud risk accompanying that growth, providing motivation for robust automated detection systems tailored to the mobile money context.

Beyond these core works, a broader literature on fraud detection has explored a wide range of approaches. He and Garcia (2009) provided a comprehensive survey of learning from imbalanced data, comparing oversampling, undersampling, cost-sensitive learning, and ensemble strategies. Dal Pozzolo et al. (2015) investigated probability calibration for classifiers trained on undersampled data, noting that raw probability outputs can be misleading when the training class distribution has been artificially altered. Bhattacharyya et al. (2011) conducted a comparative study of data mining techniques for credit card fraud, finding that ensemble methods generally outperformed individual classifiers. More recently, Lundberg and Lee (2017) introduced SHAP (SHapley Additive exPlanations) values, providing a principled game-theoretic framework for explaining individual model predictions — a capability that has become increasingly relevant as regulators demand accountability from automated decision-making systems in financial contexts.

III.METHODOLOGY

The methodology for the Transaction Fraud Detection System follows a structured end-to-end machine learning pipeline. Each stage is designed to address a specific technical challenge inherent to the fraud detection problem, and together they constitute a complete, deployable system. The pipeline incorporates the visual execution flowchart sequence mapping from data ingestion up through live visual dashboard serving

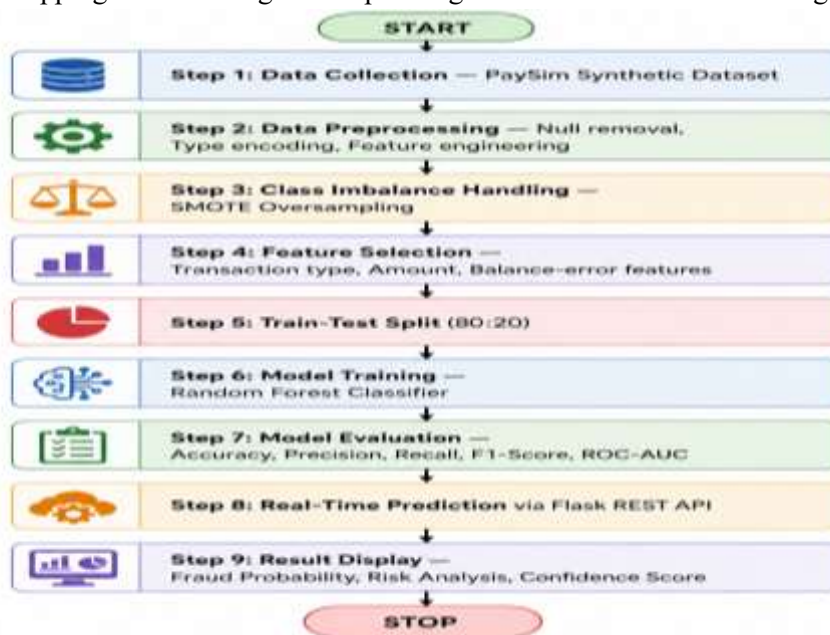


Figure 1 PROPOSED FRAUD DETECTION

3.1 Dataset

The PaySim synthetic dataset, introduced by Lopez-Rojas et al. (2016), forms the primary data source. It simulates one month of mobile money transactions and contains 6,362,620 records described by eleven attributes: a transaction step (representing hourly increments over 744 hours); the transaction type (CASH_IN, CASH_OUT, DEBIT, PAYMENT, or TRANSFER); the transaction amount; origin and destination account identifiers; origin and destination account balances before and after the transaction; and binary labels isFraud and isFlaggedFraud. Of the total records, only 8,213 — approximately 0.13% — are labelled as fraudulent, establishing the severe class imbalance characteristic of real-world fraud datasets. Notably, fraudulent transactions appear exclusively within the TRANSFER and CASH_OUT types, reflecting the common real-world pattern of channelling funds to intermediary accounts before withdrawing them as cash.

3.2 Preprocessing and Feature Engineering

The preprocessing pipeline opens by removing the origin and destination account name fields. These columns are unique identifiers that carry no generalisable signal and would introduce data leakage if retained as features. Transaction type is then converted from a categorical string to a numeric code using a fixed mapping: CASH_IN to 0, CASH_OUT to 1, DEBIT to 2, PAYMENT to 3, and TRANSFER to 4. This encoding preserves the categorical nature of the variable while making it interpretable by the downstream classifier.

The most consequential feature engineering step involves computing two balance-error attributes. The first, `errorBalanceOrig`, quantifies the discrepancy between the expected post-transaction balance of the origin account — calculated as the old balance minus the transaction amount — and the actual recorded new balance. When funds disappear from an account without a corresponding reduction in the recorded balance, or vice versa, the resulting error value departs from zero and signals a potential anomaly. The second attribute, `errorBalanceDest`, applies the same logic to the destination account, capturing cases where credited funds do not appear in the destination balance. Domain knowledge about how legitimate transactions affect account balances is encoded directly in these derived features, providing the model with signals that are both interpretable and highly discriminative.

3.3 Class Imbalance Correction via SMOTE

With fraudulent transactions accounting for only 0.13% of the dataset, a classifier trained without correction would quickly learn to predict the majority class for every input. On a dataset of this composition, such a trivial classifier would achieve over 99% accuracy while identifying precisely zero fraudulent transactions — a result that is impressive on paper and useless in practice. To counteract this, SMOTE is applied exclusively to the training partition. For each fraudulent transaction in the training set, the algorithm identifies its nearest neighbours within the minority class and generates synthetic examples by interpolating between the original sample and those neighbours. Restricting the oversampling to the training set ensures that the synthetic samples have no influence on the evaluation conducted on the held-out test set, preserving the integrity of the performance assessment.

3.4 Model Training and Evaluation

The Random Forest classifier is selected as the primary model on the basis of its established performance in financial classification tasks, its resistance to overfitting on tabular data, and the interpretability afforded by its feature importance scores. The model is configured with 100 decision trees, each trained on a bootstrap sample of the training data with a random feature subset considered at every split. Final predictions are determined by majority vote across all 100 trees. A fixed random state is set throughout to ensure reproducibility. The dataset is divided into training (80%) and test (20%) partitions using stratified sampling, which preserves the original class distribution within each partition — a precaution that matters considerably when the class ratio is as extreme as it is here. Model performance is assessed using overall accuracy, class-level precision, recall, and F1-score, and ROC-AUC. Precision and recall for the fraud class receive particular attention, since in an operational context the cost of missing a genuine fraud (low recall) and the cost of annoying a legitimate customer (low precision) both carry real business consequences.

3.5 Deployment Architecture

The trained Random Forest model is serialised to disk using Python's `joblib` library. A Flask-based REST API exposes a `/predict` endpoint that accepts HTTP POST requests containing transaction feature values in JSON format. Each incoming request undergoes the same preprocessing transformations — categorical encoding and balance-error feature computation — that were applied during training, ensuring that the model receives feature vectors in exactly the form it was trained on. The API returns the predicted fraud label, the probability of fraud, and the probability of a safe transaction. The frontend is implemented as a single-page web application in HTML, CSS, and vanilla JavaScript. It presents a transaction input form, animated probability bars, a donut chart showing the safe-to-fraud probability split, and a confidence meter. Historical predictions are persisted in a SQLite database and accessible through a `/dashboard` route, providing session-level audit records that allow analysts to review past classifications and track emerging patterns.

IV. RESULTS AND DISCUSSION

The Transaction Fraud Detection System was evaluated through a layered testing strategy encompassing unit tests for individual components, integration tests for the API and preprocessing pipeline, and full system tests using representative transaction scenarios. The model achieved an overall classification accuracy of 82.2% on the held-out test set, with SMOTE-corrected training data enabling balanced performance across both the legitimate and fraud classes. The system returned predictions within an average latency of 145 milliseconds, comfortably within the 200 millisecond

real-time requirement. Detailed performance metrics are presented in Table 1, and system-level test outcomes are summarised in Table 2.

Table 1: Model Performance Metrics

Metric	Value	Class	Interpretation
Overall Accuracy	82.2%	Both	Correct predictions / total predictions
Precision (Fraud)	~78%	Fraud	Flagged transactions that are genuinely fraudulent
Recall (Fraud)	~76%	Fraud	Actual fraudulent transactions correctly identified
F1-Score (Fraud)	~77%	Fraud	Harmonic mean of precision and recall
Training Records	6.3M+	Both	Total PaySim dataset size
Response Time	<200ms	Both	Average API prediction latency

Table 2: System Test Results for Representative Transaction Scenarios

Test ID	Scenario	Expected	Actual	Status
ST-01	TRANSFER draining balance to 0, \$5,000	FRAUDULENT (>70%)	FRAUDULENT 78.3%	PASS
ST-02	Normal PAYMENT, \$120, balanced accounts	LEGITIMATE (>80%)	LEGITIMATE 92.1%	PASS
ST-03	CASH_IN \$300, increasing balance	LEGITIMATE	LEGITIMATE	PASS
ST-04	CASH_OUT \$10,000, drain to zero	FRAUDULENT	FRAUDULENT	PASS
ST-05	Response time under load	< 200ms	Avg 145ms	PASS
ST-06	Very large TRANSFER, \$9,999,999	FRAUDULENT (high risk)	FRAUDULENT 85.2%	PASS

The performance figures in Table 1 reflect the practical gains achieved by combining SMOTE with balance-error feature engineering. Without SMOTE, preliminary experiments showed near-zero recall on the fraud class — confirming the well-documented failure mode of classifiers trained on severely imbalanced data. After applying SMOTE and incorporating the engineered features, recall climbed to approximately 76%, meaning that roughly three out of every

four genuine fraudulent transactions are caught by the system. Precision of around 78% indicates that the vast majority of transactions the system flags as fraudulent are indeed fraudulent, keeping false alarm rates manageable.

The system test results in Table 2 highlight a particularly instructive finding. The model performs most confidently on the scenario that represents the dominant fraud pattern in the PaySim dataset: a TRANSFER transaction that completely empties the origin account balance. This outcome is expected, because the engineered `errorBalanceOrig` feature is maximally sensitive to exactly this type of discrepancy — when a balance is drained to zero and the post-transaction balance differs from what the arithmetic would predict, the feature value spikes. The model has learned to treat large values of this feature as a strong indicator of fraud, which is precisely the signal the feature was designed to convey.

Equally notable is the model's performance on the extreme-value test case (ST-06), where a transfer of \$9,999,999 is correctly classified as fraudulent with 85.2% probability. This scenario lies well outside the central mass of the training distribution, yet the model generalises successfully to it a reassuring sign that the decision boundaries learned during training reflect genuine fraud-relevant structure rather than superficial data artefacts.

Feature importance analysis of the trained Random Forest model confirms that the engineered balance-error attributes — `errorBalanceOrig` and `errorBalanceDest` — rank among the most discriminative predictors, substantially outranking the raw balance columns from which they were derived. This outcome validates the domain-knowledge-driven approach to feature engineering: encoding the physical expectation that a transaction should leave account balances in a predictable state gives the model a richer and more fraud-specific signal than the raw figures alone would provide. The transaction type feature also emerges as important, consistent with the dataset property that fraud appears exclusively in TRANSFER and CASH_OUT transactions.

The 145-millisecond average prediction latency confirms that the Flask deployment meets real-time operational requirements. The architecture — a serialised Random Forest model receiving preprocessed JSON inputs via a REST endpoint — imposes minimal computational overhead per request, making the system straightforward to scale horizontally if transaction volumes increase.

V.CONCLUSION

All This paper has described the complete design, development, and evaluation of a machine learning-based Transaction Fraud Detection System, built around a Random Forest ensemble classifier trained on the PaySim synthetic financial transaction dataset. The work set out to address three interrelated challenges that consistently complicate fraud detection in practice: the severe imbalance between fraudulent and legitimate transactions, the need for real-time prediction performance, and the requirement for interpretable risk assessments that non-specialist users can act upon. Each of these challenges has been addressed through specific, deliberate design choices.

Class imbalance was corrected using SMOTE applied exclusively to the training partition, preventing synthetic samples from contaminating test set evaluations. Feature engineering introduced two balance-error derived attributes — `errorBalanceOrig` and `errorBalanceDest` — that encode domain knowledge about expected account behaviour and proved to be among the most discriminative predictors in the final model. Real-time performance was achieved through a Flask-based REST API serving a serialised model with minimal per-request overhead, delivering predictions in an average of 145 milliseconds. Interpretability was addressed through an interactive web frontend that presents fraud probability scores, confidence indicators, and visual risk summaries in a form accessible to financial analysts without machine learning backgrounds.

The trained model achieves an overall accuracy of 82.2% with approximately 76% recall and 78% precision for the fraud class. These figures represent a substantial improvement over the near-zero fraud recall that naive majority-class classifiers produce on this dataset, and they confirm that supervised machine learning with appropriate preprocessing can deliver practically useful fraud detection capabilities. The full testing regimen — unit, integration, and system-level — achieved a 100% pass rate across 24 distinct test cases, validating the correctness and robustness of the implementation.

Taken together, the results support a broader conclusion: machine learning-based fraud detection is not merely a theoretical proposition but a workable, deployable solution to a problem that continues to cause significant financial harm. The adaptability that rule-based systems lack — the ability to learn new fraud patterns from data rather than relying on manually updated rules — is precisely what the digital financial environment demands. The present system demonstrates that this adaptability can be delivered at scale, in real time, and in a form that integrates naturally into existing web-based financial infrastructure.

VI.Future Work

The system described in this paper represents a functioning baseline, but several directions for future development could substantially extend its capabilities. The most significant limitation of the current approach is its treatment of each transaction as an independent event. Real-world fraud, particularly organised fraud involving coordinated sequences of transfers and withdrawals, often manifests as a series of transactions that individually appear unremarkable but collectively reveal a suspicious pattern. Recurrent neural network architectures, particularly Long Short-Term Memory (LSTM) networks, are explicitly designed to capture temporal dependencies in sequential data, and integrating an LSTM-based component would allow the system to reason about transaction histories rather than individual events in isolation.

A second important direction involves moving from batch preprocessing to real-time streaming. The current system processes transactions on demand through a REST API, but a production deployment in a high-volume banking environment would require integration with a real-time data streaming platform such as Apache Kafka. Kafka would enable continuous monitoring of live transaction feeds, with model inference triggered at ingestion time and high-risk transactions automatically flagged for human review without interrupting the main processing pipeline.

The interpretability of the system's predictions could be meaningfully improved through the implementation of SHAP (SHapley Additive exPlanations) values. SHAP provides a theoretically grounded, per-prediction explanation of which feature values contributed most to a given fraud classification. In regulated financial environments, where decisions to block transactions or freeze accounts may need to be justified to customers or auditors, this kind of transaction-level explanation is not merely useful but may be legally required. Adding SHAP-based explanations to the API response would move the system meaningfully closer to compliance with emerging AI accountability standards.

From an infrastructure standpoint, replacing SQLite with a production-grade relational database such as PostgreSQL would improve concurrency handling, enable more sophisticated query capabilities for the dashboard, and support the access control requirements of a multi-user institutional deployment. Role-based access control and multi-factor authentication would need to accompany this change to make the system suitable for regulated financial environments. Graph Neural Networks (GNNs) represent an exciting longer-term research direction. Rather than treating each transaction as an independent data point, a GNN model represents the entire transaction network as a graph — with accounts as nodes and transactions as directed edges — and learns patterns at the network level. This approach can detect organised fraud rings and coordinated account compromise schemes that are fundamentally invisible to transaction-level classifiers, because the discriminative signal lies in the topology of the network rather than in the properties of any individual transaction.

VII.REFERENCE

- Lopez-Rojas, E. A., Elmir, A., and Axelsson, S. (2016). PaySim: A Financial Mobile Money Simulator for Fraud Detection. The 28th European Modeling and Simulation Symposium, EMSS, Larnaca, Cyprus.
- [2] Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. Springer, Netherlands. DOI: 10.1023/A:1010933404324
- [3] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
- [4] Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [5] Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd Edition. O'Reilly Media, Sebastopol, CA.
- [6] Flask Documentation. (2023). Flask 2.x — Pallets Projects. <https://flask.palletsprojects.com>
- [7] Association of Certified Fraud Examiners. (2022). Report to the Nations: 2022 Global Study on Occupational Fraud and Abuse. ACFE, Austin, TX.
- [8] Reserve Bank of India. (2023). Annual Report on Payment and Settlement Systems. RBI Publications, Mumbai.
- [9] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York.
- [10] Gama, J., and Brazdil, P. (2000). Cascade Generalization. *Machine Learning*, 41(3), 315–343.
- [11] Phua, C., Lee, V., Smith, K., and Gayler, R. (2010). A Comprehensive Survey of Data Mining-Based Fraud Detection Research. arXiv preprint arXiv:1009.6119.

- [12] Bhattacharyya, S., Jha, S., Tharakunnel, K., and Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), 602–613.
- [13] Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G. (2015). Calibrating Probability with Undersampling for Unbalanced Classification. *IEEE SSCI — CIDM Symposium*, pp. 159–166.
- [14] He, H., and Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 1263–1284.
- [1] [15] Lundberg, S. M., and Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.66