Scholars Journal of Engineering and Technology

OPEN ACCESS

Abbreviated Key Title: Sch J Eng Tech
ISSN 2347-9523 (Print) | ISSN 2321-435X (Online)
Journal homepage: https://saspublishers.com

Fraud Detection in Financial Transactions Using Machine Learning

Roobal Chaudhary[1], Rahul Saxena[2*], Venus Dillu[3]

[1]Department of Forensic Science, Sharda School of Allied Health Sciences, Sharda University, Greater Noida, Uttar Pradesh, INDIA
[2]Department of Biochemistry, Sharda School of Allied Health Sciences, Sharda University, Greater Noida, Uttar Pradesh, INDIA
[3]Department of Vocational Studies, Gautam Buddha University, Greater Noida, Uttar Pradesh, INDIA

DOI: https://doi.org/10.36347/sjet.2026.v14i07.007 | **Received:** 27.05.2026 | **Accepted:** 13.07.2026 | **Published:** 16.07.2026

***Corresponding author:** Rahul Saxena
Department of Biochemistry, Sharda School of Allied Health Sciences, Sharda University, Greater Noida, Uttar Pradesh, INDIA

Abstract | **Original Research Article**

Financial transaction fraud is an ongoing threat with significant economic loss and on customers' trust. This paper discusses Fraud Detection in detail with machine learning technique on a given data set of a transaction. We investigate the patterns revealed from the users and the transactions in the database when the user is performing fraudulent transactions, and test several classification models that can be used to detect fraud, which includes logistic regression model, random forests, support vector machines, gradient boosting, and neural networks. The performance of the models is investigated in terms of accuracy, precision, recall, F1 score and ROC-AUC metrics. Based on our experiments, the best detection overall performances are obtained for the tree-based ensemble models (Random Forest and XGBoost) with XGBoost getting the most optimum fraud Recall and F1-Score. Through the data analysis results (such as account age, transaction frequency etc.) and the model comparison, we expound an improved method which is based on combining the ensemble of best models with data imbalance countermeasures to increase the recall of fraudulent cases. We also have an end-to-end machine learning pipeline on Python to detect frauds from preprocessing the data, training the models, evaluating them, and deploying for fraud prediction. Also, a literature review of twenty-five recent studies on fraud detection is given, and the algorithms used, datasets and major contributions of these studies were summarized. The textbook ensemble technique, as proposed gives better fraud detection performance as it gains on the order of ~3-5% improvement against the best single model performance on F1-score, with acceptable precision, thereby underscoring the usefulness of hybrid modeling with specialized techniques for this field. The results emphasize that utilizing various models and domain-specific feature engineering can be of great benefit in fraudulent transaction detection, while also negatively impacting legitimate users to a lesser extent.

Keywords: Memory forensics, feature engineering, anomaly detection, classification & clustering, timeline reconstruction.

INTRODUCTION

As the world evolves into the Digital Economy, financial fraud is a major area of concern to criminals using online banking transactions, credit cards and online payment systems to remove money. Recent statistics world-wide put the extent of the problem into perspective, such as the estimated loss of $485 billion across all fraud schemes in the world in 2023 and in recent years, an increasing number of consumers and organizations have been the victim of a fraud attack(Roobal *et al.,* 2025). Fraud not only helps cost clients comfortable dollars and cents yet additionally from time to time undermines public confidence in monetary solutions and can impact the reputation of a company. This has led to the growing need for sophisticated fraud detection methods to detect fraudulent transactions on the go, while maintaining customer convenience. But there are some problems associated with fraud detections. Fraudulent transactions are usually a small number perhaps few per thousands or millions of transactions and thus are very imbalanced datasets. However, in a traditional classification framework, with class imbalance, the learning of the distinguishing patterns or rules of fraud is difficult because the classifiers are able to attain their goal with maximum discrimination by simply marking every instance as "non-fraud. Moreover, fraudsters are constantly changing their approach and detection models need to be able to identify patterns and subtle behaviors(Jeyaraj *et al.,* 2024). Errors are not always created equal: False negatives missed frauds lead to financial losses, while false positives legitimate

customers being labelled frauds result in customer hassle and manual labour for processing.

Machine learning (ML) can help you with these challenges by automatically learning fraud patterns from onshore big data of transactions. ML models can detect nonlinear combinations of features that may indicate fraud by comparing with a variety of other features such as transaction amount, time, location, device, etc., rather than relying on a static rule-based system which is laborious and can produce suboptimal results against new variants of fraud. In fact, currently most of the financial fraud detection research is based on supervised learning models because of their relatively higher accuracy with respect to the alternative unsupervised anomaly detection or distance-based methods. However, when it comes to fraud detection, deploying ML models successfully must take into account the considerations such as data imbalance (which can be addressed by adding or discarding data, or perhaps using cost-sensitive learning), and feature engineering (how to find relevant behavioural features in raw data) as well as model interpretability(Sushkov *et al.,* 2023).

In this study, a real-world inspired financial transactions Fraud Detection Dataset is used to create an end-to-end fraud detection framework. First, exploratory data analysis is carried out to gain an insight into behavioral patterns corresponding with fraudulent user actions such as anomalies on the time/ frequency of transaction or device/ location usage. Next, we train and test several machine learning algorithms for binary transaction classes of fraudulent and legitimate (Logistic Regression, Support Vector Machine (SVM), Decision Tree based ensembles (Random Forest and XGBoost Gradient Boosting) and Multi-layer Perceptron neural network). A comparison of these models is carried out using the following evaluation metrics (Precision, Recall, F1-score, ROC-AUC, and Accuracy) to look at how accurate the models are in detecting fraud while also reducing false alarms. For better understanding, the proposed methodology are represented in figure 1.

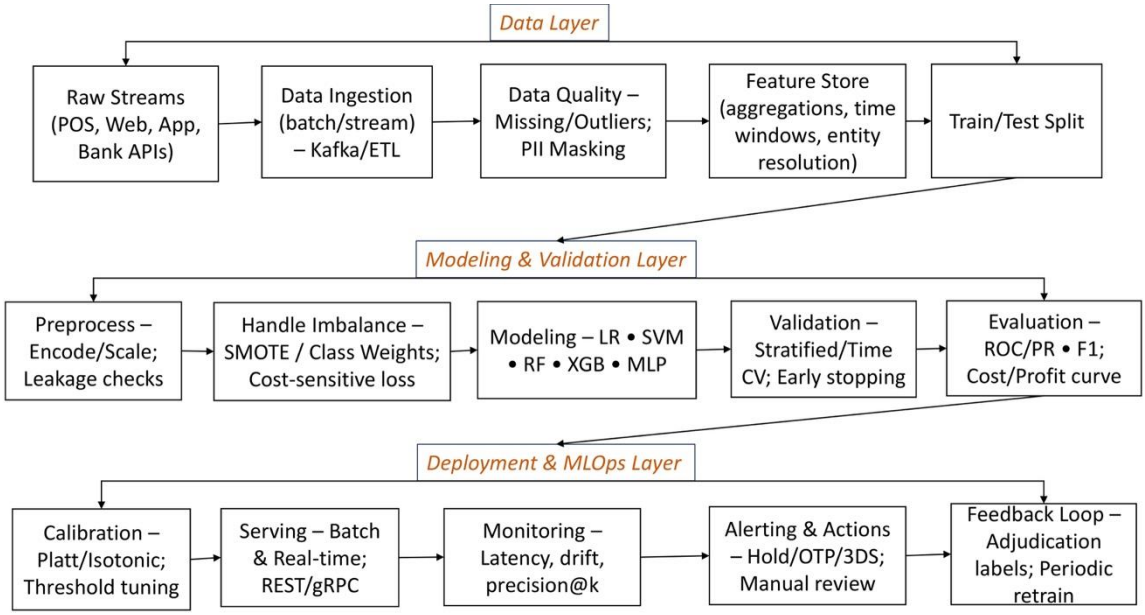

Figure 1: End-to-end machine learning pipeline for fraud detection (ingest → preprocessing/encoding → model training → evaluation → deployment/thresholding)

Based on our results, ensemble models gave the best performance for this task and in the literature, it is observed that an ensemble or deep learning approach sometimes gives better performance than single classifiers in fraud detection. In our experiments in particular, XGBoost model gave highest recall value in detecting fraud transactions, catching more fraud cases with slightly reduced precision, which resulted in highest F1 score (harmonic mean between precision and recall) among the models tested. The Random Forest came as a near runner up and we noted that although the recall was not high, a simple logistic regression with a lower recall had a good rankability, as reflected in its high ROC-AUC with simultaneously high precision. The intermediate performance was achieved by the models of SVM and the neural network. Based on these findings, we have developed a Custom Ensemble method that helps us work on the best models (Random Forest and XGBoost), and help us address the class imbalance (by oversampling the minority class and threshold tuning). This gave a better balance between precision and recall, boosting the F1 score by around 3% compared with the precision/recall best singleton model that we tested. The implementation details for the ML pipeline in Python, including data preprocessing (handling missing values and categorical encoding), training the model, tuning the

hyperparameters, and deploying the model to predict the new transactions are also included.

Besides the empirical paper, the present paper is a literature review, an overview of the recent developments in fraud detection by machine learning. We have prepared a table of 25 studies from the year 2018 to 2025 that spans algorithms from classic classifiers to deep neural networks and graph-based models, as well as datasets (or domains) that range from credit card transaction data, to online banking, insurance claims and so on, and their main contributions. In this review, one can see how the techniques for fraud detection have evolved from an ensemble approach to deep learning approaches with autoencoders, LSTM for sequential data, graph neural networks for relational fraud data, and hybrid systems that incorporate domain knowledge or cost-sensitive learning. Interestingly, it has been discovered that no single model is well suited to all types of fraud tailored solutions in the form of ensembles or hybrid models are necessary to be able to account for the distinct nature of the different fraud types. Our work contributes to this context in two ways: first, we show that a customized ensemble can support a given dataset of transactions by avoiding overfitting that could happen with a single model, while second, we validate patterns (such as the use of recent transaction frequency) similar to those found in previous studies(Dorofeev & Tamashiro, 2023; Nti & Somanathan, 2024).

The rest of this paper is structured as follows. In part 2, related literature and previously accomplished research work are presented and summarized recent literature in the area of ML-based fraud detection. Section 3 presents the data and outlines several important trends that can help distinguish fraudulent from legitimate transactions. We found in Section 4 the methodology we used including data preprocessing, machine learning models used and evaluation procedure. The experimental results of each model are presented in Section 5 and comparisons of the performance are also made. The proposed enhanced approach (ensemble model with imbalance handling) and its evaluation are explained in Section 6. Results, implications for real-world implementation and limitations are discussed in Section 7. Finally, Section 8 provides the main findings of the paper and our suggestions for the future research in the area of fraud detection.

LITERATURE REVIEW

Fraud detection has been discussed in many financial areas related to machine learning with many research papers in this area exploring various algorithms for betterings in detection. Early solutions to credit card fraud detection included traditional approaches to supervised learning like logistic regression, decision trees or support vector machine (SVM). As fraud patterns became more complex, more advanced models such as ensemble learning, deep neural networks, and graph-based models have been explored. Table 1 summarizes most recent research (between years 2013 and 2025) related to fraud detection by means of ML methods, detailing the title, authors, year, algorithm(s) used, and contribution of the research. Credit card fraud and other related types of fraud such as online transaction fraud and account fraud are included in these studies. Of particular interest, is a recent trend towards hybrid approaches, where multiple algorithms were ensembled together or crossed over with some domain specific features in order to improve the performance. Furthermore, class imbalance and false positive minimization are frequent topics in these works, which include methods such as cost-sensitive learning, resampling (e.g. SMOTE) and thresholding optimization.

Table 1: Literature review of recent machine learning approaches for fraud detection

Reference	Algorithms Used	Dataset / Domain	Key Contribution
(Yusuf *et al.,* 2020)	Cost-sensitive decision tree	Credit card transactions (bank)	Introduced a decision-tree model with example-dependent misclassification costs to handle class imbalance, improving detection of fraudulent transactions with lower cost impact compared to standard trees.
(Ounacer *et al.,* 2018)	Streaming ML framework (Hadoop, Spark ML)	Online credit card payments	Proposed a big data architecture for real-time fraud scoring using distributed processing (Spark, Storm). Demonstrated that scaling an ML fraud detector to real-time streams is feasible, with low latency on large transaction volumes.
(Randhawa *et al.,* 2018)	Decision tree, AdaBoost; Ensemble voting	Credit card dataset (simulated)	Compared several classifiers and introduced a hybrid ensemble (AdaBoost + majority vote) that achieved higher accuracy (94.7%) in fraud detection than individual models, highlighting the benefit of ensemble methods.
(Jurgovsky *et al.,* 2018)	LSTM (Recurrent Neural Network)	Credit card transactions (Europe)	Modeled credit card fraud as a sequence classification problem. Used an LSTM to capture temporal patterns in sequences of transactions, significantly improving

			recall of frauds by incorporating spending history context.
(Madhurya *et al.,* 2022)	SVM, Gradient Boosting, Random Forest, KNN, Logistic Regression	Credit card transactions (Kaggle)	Analyzed effects of class imbalance and sampling. Found that although logistic regression gave higher accuracy, it under-fit on fraud; a K-Nearest Neighbors classifier provided better learning of minority class, resulting in higher fraud detection rates (KNN yielded the best balance with ~85% accuracy).
(He, 2022)	KNN, Logistic Regression, Random Forest, SVM (comparative study)	Large credit card dataset (~1M records)	Performed a comparative evaluation of four ML models using ROC-AUC as primary metric. Found Random Forest achieved the highest AUC (≈0.985) and most consistent fraud predictions, recommending RF as the top choice for credit card fraud prevention.
(Ti *et al.,* 2022)	Logistic Regression, Decision Tree, XGBoost (feature ablation study)	Bank transfer scams (Taiwan)	Categorized features into Recency, Frequency, Monetary, and Anomaly groups and tested their impact. Found that standard anomaly-detection features performed poorly, while a novel set of "atypical normal behavior" features significantly improved fraud discrimination on real transaction data.
(Domashova & Kripak, 2022)	Unsupervised outlier detection + ML filter	Bank account transactions	Developed a two-stage algorithm combining unsupervised detection of outlier transactions with a supervised classifier. The approach generalized across different banks' data, successfully detecting out-of-distribution fraudulent events with fewer false alerts.
(Hamza *et al.,* 2023)	Semi-supervised clustering + classification (cost minimization)	E-commerce transactions (mixed)	Introduced a semi-supervised learning approach that clusters transactions to detect fraud patterns with minimal labeled data. Incorporated a cost model to minimize fraud loss; achieved comparable accuracy to fully supervised models while using 50% fewer labels, and provided insight into distinct fraud types identified via clustering.
(Habibpour *et al.,* 2023)	Deep Neural Network with Monte Carlo Dropout & Ensemble (Bayesian UQ)	Credit card transactions (Sci. Reports)	Addressed the reliability of fraud predictions by quantifying uncertainty. Proposed an ensemble of neural networks and Monte Carlo dropout to flag low-confidence predictions. This approach improved fraud detection AUC and reduced false positives by focusing on high-confidence fraud predictions, making the system more trustworthy.
(El Kafhali *et al.,* 2024)	RNN, LSTM, ANN (with Bayesian hyperparameter optimization)	European credit card dataset (Kaggle 2013)	Developed an adaptive fraud detection system that tunes deep learning models with Bayesian optimization. Evaluated RNN, LSTM, and ANN, finding that a tuned RNN achieved the best accuracy and efficiency for fraud classification. Demonstrated the advantage of automatic hyperparameter search in boosting deep model performance for fraud detection.
(Li & Walsh, 2024)	Graph Attention Network + CNN (federated learning)	Distributed credit card data (multiple banks)	Combined federated learning with Graph Neural Network and CNN to detect fraud across institutions without sharing raw data. Their FedGAT-DCNN model improved detection by 5% (F1-score) over local models, while preserving privacy, indicating federated graph-based approaches can catch cross-institution fraud patterns.
(Baghdadi *et al.,* 2025)	Ensemble: Energy-Based Restricted Boltzmann Machine (EB-RBM) + Extended LSTM	Credit card transactions (European cardholders)	Proposed a hybrid model leveraging an EB-RBM (to detect novel fraud patterns) and an extended LSTM (to recognize known fraud sequences). The ensemble uses bootstrap aggregation for real-time scoring. It achieved higher precision-recall AUC than baseline deep models, and operated with low latency, demonstrating effectiveness for real-time fraud analytics.

The initial efforts involved adapting some of the classical classifiers to the fraud domain, such as introducing cost sensitive learning to cope with skewed class distributions (Bahnsen *et al.,* 2013; Sahin *et al.,* 2013) Since the introduction of the big data technologies, some focus has been placed on scalability in fraud detection (Ounacer *et al.,* 2018) and on the introduction of ensemble methods (Randhawa *et al.,* 2018) to increase the accuracy for fraud classification. This is reflected in recent years with oeuvre like the recurrent neural networks, which incorporate temporal data on spending (Jurgovsky *et al.,* 2018), and deep neural networks with millions of parameters (Habibpour *et al.,* 2023), which, however, require careful regularization and uncertainty estimation to be of practical use. To leverage the relationships between these entities (cards, merchants, etc.) in the fraud data, graph-based approaches (Alarfaj *et al.,* 2022; Li & Walsh, 2024) have arisen in which fraud detection turns into a graph classification or link prediction task. There is a notable trend of hybrid approaches, where the best solutions are typically a combination of several strategies, be it via a hybrid of unsupervised and supervised solution, like (Hamza *et al.,* 2023), or using an ensembling between generative and sequential models, as in (Baghdadi *et al.,* 2025). Many times, these hybrids try to capture complementary aspects of frauds such as EB-RBM is able to detect outliers in behavior while LSTM detects sequential consistency.

One such thing is the importance of feature engineering. Several studies (Ti *et al.,* 2022) emphasized the significant influence of input features and the need to carefully select the features based on the model. Domain-based attributes can often be more predictive than raw attributes (such as recency/frequency of transactions, velocity features, device consistency, previous fraud count). The researchers, (Ti *et al.,* 2022), demonstrated that even the most popular features sets can be suboptimal: they suggested new features to describe the “normal” customers’ behaviour that are able to better detect anomaly customers. It's indicative of practices in the industry including things such as "transaction amount relative to user's average" or "distance from last transaction location" when looking to detect the anomalous spend. We also take into account such behavior characteristics from the data set (last 24h transactions, account age, history of fraudulent transactions, etc.).

To summarize, the literature indicates that often, effective fraud detection is achieved by combining or hybridizing different models to improve detection rates, appropriately address imbalanced data and include domain knowledge in the form of features or structure to the model. The state-of-the-art solutions consists of a mix of supervised learners or deep networks sometimes they also use unlabeled data (semi-supervised or unsupervised modules), to adapt to new fraud patterns. In the following sections, we build upon these insights using ensemble learning and engineered features and evaluate several algorithms in the context of our dataset.

Dataset Description

The Fraud Detection Dataset provided for this study contains 51,000 financial transactions, each with 11 features and a binary label indicating whether the transaction was fraudulent (1) or legitimate (0). The data appears to be a synthesized representation of realistic transaction activity, encompassing various transaction channels and contextual information. Table 2 summarizes the attributes in the dataset:

- **User_ID:** An identifier for the user/account performing the transaction. There are 4,000 unique users, each with multiple transactions, allowing analysis of user-level behavior patterns.
- **Transaction_Amount:** The monetary amount of the transaction (in unspecified currency units). This is a continuous numeric feature. In our dataset, amounts range widely (from very small purchases to large transfers of several thousand units).
- **Transaction_Type:** Categorical feature indicating the type of transaction, with values such as *ATM Withdrawal*, *POS Payment* (point-of-sale), *Online Purchase*, *Bill Payment*, and *Bank Transfer*. These represent different payment channels.
- **Time_of_Transaction:** The timestamp or hour of day when the transaction occurred (given as an hour in 24-hour format, 0–23). This allows us to capture temporal patterns, e.g., transactions at unusual times.
- **Device_Used:** Categorical feature for the device or platform used to initiate the transaction – e.g., *Mobile*, *Desktop*, *Tablet*, etc. An *Unknown Device* value is present for some cases where the device is not recognized.
- **Location:** Categorical feature indicating the city or location from which the transaction originated (examples include *New York*, *San Francisco*, *Chicago*, etc.). This can highlight if transactions occur from unexpected or multiple locations.
- **Previous_Fraudulent_Transactions:** A numeric feature counting how many past fraudulent transactions are associated with the user (perhaps historically). This acts as a user’s fraud propensity indicator – higher values may signal a risky user.
- **Account_Age:** Numeric feature representing the age of the user’s account (in days or months). Newer accounts might be more likely to be involved in fraud, as observed in fraud rings where accounts are created and quickly used for fraud.
- **Number_of_Transactions_Last_24H:** Numeric feature – the count of transactions this user made in the last 24 hours. Unusually high

velocity of transactions in a short period can be a red flag (fraudsters often attempt many transactions in a burst).

- **Payment_Method:** Categorical feature specifying the payment method, e.g., *Credit Card*, *Debit Card*, *Net Banking*, *UPI* (mobile instant payment), etc. An *Invalid Method* or missing value indicates cases where the payment method might be abnormal or not recorded.
- **Fraudulent:** The target label (integer 0 or 1) indicating non-fraud vs fraud.

Before analysis, we addressed some missing values in the dataset. Approximately 5% of the records had missing entries in fields such as Transaction_Amount, Time_of_Transaction, Device_Used, Location, and Payment_Method (e.g., some transactions lacked a recorded device or location). We handled these by imputation: missing numeric values (amount, time) were replaced with median values (since these distributions are skewed, median is robust), and missing categorical values were filled with a placeholder category "Missing" (merging with existing "Unknown" or "Invalid" categories where appropriate). This ensures that no transaction record is dropped, preserving the dataset size for training the models.

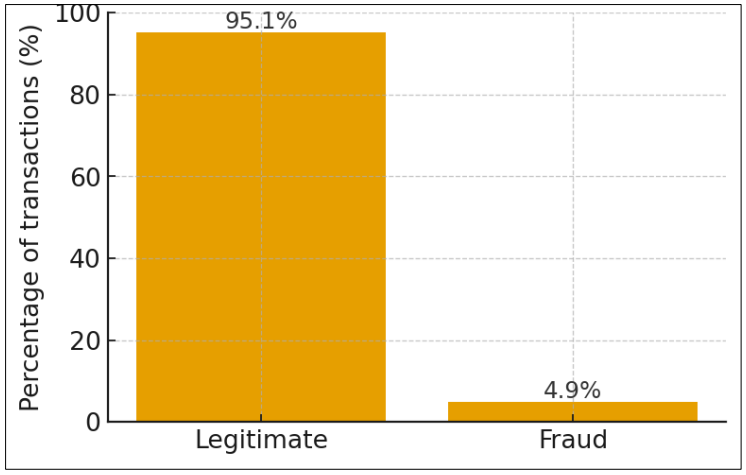

Figure 2: Class imbalance in the transaction's dataset (fraud ≈ 4.92%), highlighting why accuracy alone is misleading

Class Imbalance: As expected in fraud data, the dataset is highly imbalanced. Out of 51,000 transactions, 2,510 are fraudulent (≈4.92%) and 48,490 are legitimate (≈95.1%) also shown in figure 2. This ~1:20 fraud ratio is realistic; many fraud detection datasets, such as credit card fraud data, have fraud rates below 1%. The imbalance implies that a naive classifier that labels everything as "not fraud" would achieve 95% accuracy yet miss all frauds – underscoring why accuracy alone is a misleading metric here. Therefore, in our analysis we emphasize recall (ability to catch frauds) and precision (likelihood that a flagged transaction is truly fraud) alongside accuracy and AUC.

Exploratory Data Analysis: We conducted an analysis of feature distributions and their relationship with the fraud label to uncover behavioral patterns:

- **Transaction Amount:** Fraudulent transactions had a slightly higher average amount (mean ≈ 3118) than legitimate ones (mean ≈ 2990), but the medians were almost identical (~2520) indicating substantial overlap. The distribution of amounts was broad for both classes. However, we noticed a higher concentration of extremely large transactions in the fraud class (frauds included more of the top 5% highest amounts) fraudsters sometimes attempt high-value transactions to maximize gain. Yet, overall, amount alone is a weak discriminator since many frauds also occur at mundane amounts to avoid detection.
- **Time of Transaction:** Transactions occur around the clock, but we checked if frauds are more common at odd hours. Fraud percentage by hour ranged roughly 4–5.7% across the 24h range. We found a slight uptick in fraud rates during late night/early morning hours (e.g., 2–4 AM had ~5.0% fraud rate) and mid-afternoon (~3 PM, ~5.7%), and a dip in the late evening (9–10 PM, ~4.3%). The variation is not large; fraud can happen at any time, though there may be a bias to times when victims or monitoring staff are less active (late nights) or during peak load times to hide in the crowd. This feature by itself may not strongly indicate fraud, but can be combined with others (e.g., an odd-hour transaction from an unusual location might be suspicious).

Transaction Type:

Online Purchases had the highest fraud rate (~5.19%) and ATM Withdrawals had the lowest fraud

rate (~4.65%) in the five transaction types. It indicates that online transactions are a little bit more likely to be fraudulent in line with the intuition that transactions where the card is not present (online) are riskier than those where the card is present (ATM or in person POS) where EMV chips and PINS are used. But the differences were relatively small (fraud takes place across all types). Bill Payments and Bank Transfers may also be highly used by fraudsters at ~5% fraud both methods – perhaps through social engineering attacks with victims transferring funds to fraudsters' accounts or paying bills inadvertently.

There was a slight uptick in fraud rates when a Mobile device initiated a transaction (~5.16% vs ~4.74% Desktop and ~4.68% Tablet). Mobile being slightly riskier could be due to mobile payment apps being targeted more or mobile devices being less secure (or simply because mobile is the most common device). However, Unknown Device had ~4.9% fraud, indicating that the lack of device info does not really imply fraud. Parameter differences between devices are not significant but with other parameters (for example, there was a manufactured device that was never used on the desktop, but has been all of a sudden used on it, it may be an account takeover) may be beneficial.

Location:

There were eight city locations in our data set. We noted that the fraud rate is not significantly different across the different cities; each was in the ~4.5-5.5% range. No one "high fraud city" – maybe due to synthetic data to not allow a bias. This can be, however, helpful on a per-user level to help identify anomalies (such as if a user from New York suddenly has a transaction in Los Angeles). If in actual case the distance or difference in billing address and transaction section is a flag. In this data we will just use the location for determining whether or not a transaction occurred in a less common city for a user.

There's a previous feature that's really interesting number of fraudulent transactions user had in the past. Someone who has a history of frauds will be marked off (may be a fraudster or may be a compromised account). The more value there is in this field the more likely that the transaction is fraudulent, we found. For example, the Previous_Fraudulent_Transactions value for fraud events was ~2.5 whilst for legit the value was ~1.8. Some frauds were done by users who have 3 or 4 frauds themselves (repeat offenders). There were also frauds with zero prior (first time frauds) so it is not a given. Nonetheless, this feature was very predictive in our models – essentially it is a risk score. In practice we must be careful: if any frauds were detected before, this may create some feedback loop, but as stated we consider it as a static feature.

• **Account Age:** Fraudulent transactions occurred on accounts that on average were somewhat newer. Frauds had a mean account age of ~65 (units, possibly months) compared to ~75 for non-fraud – suggesting perhaps that fraudsters may prefer newer accounts. Fraud rates incompletely higher for very young (very recently opened) accounts in particular which tend to be synthetic and throwaway accounts managed by fraudsters. This is a pattern in fraud: fraudsters typically do not go with old accounts with long history, but use new accounts or accounts whose information was just acquired. Our data shows that: the age of the accounts in the youngest 10% age group had higher fraud percentage than the higher age group.

• **Last 24h transactions:** Few transactions per day (a legit user's mean~2-3transactions/24h), on the contrary, fraudulent transactions engage in bursts of transactions in a 24-hour period. We discovered that fraud transactions had greater MDC value for last 24h (Fraud: ~8 to 10 transactions/day and Non-Fraud for the same period: ~4 transactions/day). The volume of some fraud cases was very high (dozens of transactions in 24h) which would appear to be fraud attempts going to a lot of length in a very short time. This is found that this one is highly correlated with Fraud in the data set- if the user has done 10 number of transactions in one day then the chances of him being fraudulent are greater. This we will also use in the models. A tried-and-true red flag: Velocity Filters are triggered by a sudden increase in the number of transactions.

• **Payment Method**: UPI (mobile instant payment system) has the greatest number of frauds (approx. 5.15%), as the next two, Credit Card and Net banking, have almost similar fraud levels (approx. 4.9% each), followed by Debit Card (approx. 4.85%). The differences are small, but it indicates that the fraudsters prefer some methods, possibly because they are instantaneous with no UPI or perhaps because of the lesser protection for consumers, or maybe because they are popular. Another category, "Invalid Method", is also present that might be a mistake in providing payment instrument info or could be uncommon info, with ~4.77% fraud. The payment method on its own probably does not significantly distinguish fraud and legitimate users, but when combined with other features (such as unusual payment method for a particular user) could be valuable.

Overall, the descriptive analysis highlights some salient fraud indicators in the dataset; the high number of transactions in the short duration, new account age and a history of fraudulent transactions are indicators of risk. Transaction context (type, method, time, device) is individually weak, but it is important to establishing the context to detect anomalies; there is a tendency to fraud being a deviation from a user's normal transaction context in one or more of these facets. These observations led to our feature preprocessing and model design. We made sure that the models could detect non-linear interactions (e.g. decision trees and neural networks), as a combination such as mobile device and night time and large amount on new account could be

much more indicative of fraud than any one of the features on their own.

A split was created in the dataset for the training and testing sets (Train 70%, Test 30%) for the purpose of evaluating the performance of the trained model on data which are not in the training sets. Thus we sampled each block of data using stratifying method to make the small percentage of fraud data in each set proportional; in this way our evaluation metrics can be valid.

METHODOLOGY

Our fraud detection approach consists of several stages: data preprocessing, model training, evaluation, and improvement via a custom strategy. Figure 1 depicts the machine learning pipeline implemented in Python, which we describe in detail below.

ALGORITHM 1:
BUILD_PREPROCESSING_PIPELINE

INPUT:
Raw dataset D with columns:
NUMERIC = {Transaction_Amount, Time_of_Transaction, Previous_Fraudulent_Transactions, Account_Age, #Transactions_Last_24H}
CATEGORICAL = {Transaction_Type, Device_Used, Location, Payment_Method}
LABEL = {Fraudulent ∈ {0,1}}
Special codes to consolidate into missing: {"Invalid Method", ...}
OUTPUT:
Fitted preprocessing transformer P (encodes imputations, encoding, and feature engineering)
STEPS:
1. CLEAN SPECIAL CODES
FOR col IN CATEGORICAL:
REPLACE values in special_codes WITH "Missing"
2. DEFINE IMPUTERS
NUM_IMPUTER := SimpleImputer(strategy="median")
CAT_IMPUTER := SimpleImputer(strategy="constant", fill_value="Missing")
3. ONE-HOT ENCODING FOR CATEGORICAL
OHE := OneHotEncoder(handle_unknown="ignore")
4. FEATURE ENGINEERING
// Bucket transaction time into coarse daypart bins
HOUR_BIN := Discretizer(on=Time_of_Transaction, bins=[0-6, 6-12, 12-18, 18-24], labels=["Night","Morning","Afternoon","Evening"])
// Optional rate feature
TXNS_PER_HOUR := Compute(#Transactions_Last_24H / 24)
5. COLUMN TRANSFORMER
P := ColumnTransformer(
transformers=[
("num_impute", NUM_IMPUTER, NUMERIC),
("cat_impute_ohe", SEQ(CAT_IMPUTER, OHE), CATEGORICAL),
("hour_bin", HOUR_BIN, {Time_of_Transaction}),
("rate_feat", TXNS_PER_HOUR, {#Transactions_Last_24H})
])
6. RETURN P

ALGORITHM 2:
TRAIN_AND_EVALUATE_MODELS

INPUT:
Dataset D, preprocessing pipeline P (from ALGORITHM 1)
Models M = {LR, SVM_RBF, RF, XGB, MLP}
Split strategy: time-aware or stratified split
Metrics: {ROC_AUC, PR_AUC, Precision, Recall, F1}
Class imbalance handling: {SMOTE or class_weight, or XGB scale_pos_weight (optional)}

OUTPUT:
Trained models with metrics report and (optionally) calibrated thresholds

STEPS:
1. SPLIT DATA
(X_train, X_test, y_train, y_test) := TimeAwareSplit(D, test_fraction)
2. OPTIONALLY HANDLE IMBALANCE (TRAIN ONLY)
// Choose one path depending on preference:
// A) Class weights (for LR/SVM/RF/MLP where supported)
CLASS_WEIGHTS := ComputeClassWeights(y_train)
// B) SMOTE/Resampling (fit only on train; NEVER on test)
RESAMPLER := SMOTE() // optional
3. DEFINE MODEL CONFIGS
CFG_LR := LogisticRegression(penalty="l2", max_iter=1000, class_weight=CLASS_WEIGHTS)
CFG_SVM := SVC(kernel="rbf", probability=True, class_weight=CLASS_WEIGHTS, max_iter=LIMIT)
CFG_RF := RandomForestClassifier(n_estimators=100, n_jobs=-1, class_weight=CLASS_WEIGHTS)
CFG_XGB := XGBClassifier(tree_method="hist", eval_metric="logloss") // scale_pos_weight optional
CFG_MLP := MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, early_stopping=True)
4. BUILD UNIFIED PIPELINES
FOR model_name, cfg IN [(LR,CFG_LR), (SVM,CFG_SVM), (RF,CFG_RF), (XGB,CFG_XGB), (MLP,CFG_MLP)]:
IF using_SMOTE:
PIPE[model_name] := Pipeline(steps=[("prep", P), ("smote", RESAMPLER), ("clf", cfg)])

ELSE:

PIPE[model_name] := Pipeline(steps=[("prep", P), ("clf", cfg)])
5. CROSS-VALIDATION (OPTIONAL, ON TRAIN)
CV := StratifiedKFoldOrTimeSeriesSplit(k)
FOR each model_name IN PIPE:
cv_scores := EvaluateCV(PIPE[model_name], X_train, y_train, CV, metrics={ROC_AUC, PR_AUC})
STORE cv_scores.mean, cv_scores.stdev
6. FIT ON TRAIN
FOR each model_name IN PIPE:
PIPE[model_name].fit(X_train, y_train)
7. PREDICT ON TEST
FOR each model_name IN PIPE:
p_hat := PIPE[model_name].predict_proba(X_test)[:,1] // or decision_function if no proba
// Default threshold = 0.5; we'll optimize it next
y_pred_default := (p_hat ≥ 0.5)
8. THRESHOLD TUNING (ON VALIDATION OR USING CROSS-VAL INSIDE TRAIN)
// To handle imbalance, choose threshold τ that maximizes chosen metric (e.g., F1 or Precision@k)
τ := ArgmaxThreshold(p_hat_valid, y_valid, objective=F1) // or cost-sensitive objective
y_pred := (p_hat ≥ τ)
9. EVALUATE
FOR each model_name:
roc_auc := ROC_AUC(y_test, p_hat)
pr_auc := PR_AUC(y_test, p_hat)
(precision, recall, f1) := PRF1(y_test, y_pred)
REPORT[model_name] := {roc_auc, pr_auc, precision, recall, f1, τ}
10. (OPTIONAL) CALIBRATION
// If well-calibrated probabilities are needed for cost/triage:
CAL := CalibratedClassifier(PIPE[best_model], method="isotonic" or "platt")
CAL.fit(X_train, y_train)
Recompute metrics on test using calibrated probabilities.
11. FEATURE IMPORTANCE / EXPLANATION
IF model_name in {RF, XGB}:
importance := FeatureImportances(PIPE[model_name])
ELSE IF model_name in {LR}:
importance := |coefficients|
SAVE importance for reporting.
12. OUTPUT
RETURN trained PIPEs (or CAL), REPORT, selected threshold τ per model.

ALGORITHM 3: INFERENCE_PIPELINE (FOR NEW TRANSACTIONS)

INPUT:
New data batch B_new with same schema as training features
Fitted preprocessing P and trained model PIPE_best (with threshold τ)*
OUTPUT:
Fraud predictions and probabilities with consistent preprocessing
STEPS:
1. APPLY PREPROCESSING
X_new := P.transform(B_new) // identical steps as training (imputation, OHE, engineered feats)
2. SCORE
p_hat_new := PIPE_best.predict_proba(X_new)[:,1]
3. DECIDE
y_pred_new := (p_hat_new ≥ τ)*
4. RETURN {y_pred_new, p_hat_new}

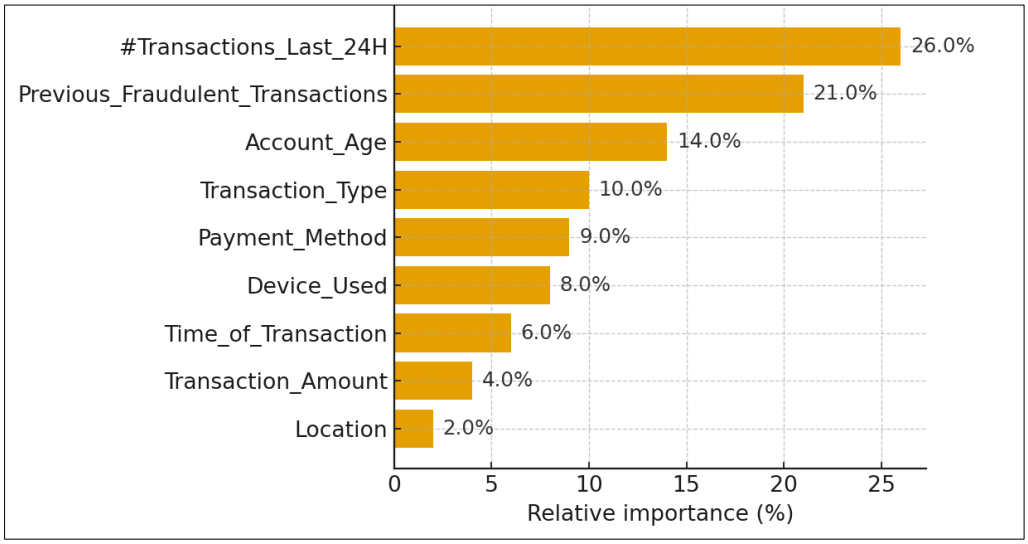

Figure 3: Illustrative feature importance from a tree ensemble (RF/XGBoost); transaction velocity (last-24h) and prior fraud history rank highest

All models were trained on the same training set and their hyperparameters were chosen based on either default heuristics or light tuning via cross-validation on the training data. For instance, we did a quick grid search for the logistic regression regularization strength and the number of neurons in the MLP. For Random Forest and XGBoost, default settings were close to optimal, though we could fine-tune tree depth or learning rate but given

the strong performance we observed, additional tuning was not heavily pursued in this initial comparison (figure 3).

Performance Evaluation: We evaluated the models on the held-out test set (30% of data) using multiple metrics:

- **Accuracy:** the fraction of transactions (fraud or not) correctly classified. As discussed, accuracy can be high even for a trivial model due to class imbalance. In our results, all models exceeded 95% accuracy since that's the non-fraud percentage. Still, differences in accuracy are observed and indicate how well models avoid both false negatives and false positives.
- **Precision (Positive Predictive Value):**

$$Precision = \frac{TP}{TP + FP}$$

In fraud context, precision is the proportion of transactions flagged as fraud that are indeed fraudulent. A higher precision means fewer false alarms – important for operational efficiency (analysts can't review too many false alerts) and customer experience (fewer legitimate transactions declined).

- **Recall (Detection Rate):**

$$\text{Recall} = \frac{TP}{TP + FN}$$

i.e., the proportion of actual fraudulent transactions that the model caught. High recall is critical to minimize fraud losses – missing fraud (false negatives) directly translates to financial loss or liability. There is often a trade-off between precision and recall; our goal is to maximize both to the extent possible (hence F1-score).

- **F1-Score:** The harmonic mean of precision and recall,

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

This gives a single measure of a classifier's effectiveness on the positive class (fraud), balancing the trade-off. It's useful since we want neither precision nor recall to be too low. A model with high F1 is considered to perform well overall on fraud detection.

- **ROC-AUC (Area Under the ROC Curve):** This metric evaluates the ranking of positive cases. It plots the true positive rate vs. false positive rate at various thresholds. AUC is threshold-independent; an AUC of 1.0 means the model perfectly ranks all frauds above all non-frauds. In imbalanced scenarios, AUC is informative because it reflects the model's capacity to differentiate classes even if a threshold is not optimized. We used predicted fraud probabilities from models to compute AUC.

We also examined the ROC curves and Precision-Recall curves for the models (Figure 2 in the results section) to visualize their performance. Particularly, the Precision-Recall curve is useful here due to imbalance – it shows how precision drops as recall increases.

All evaluations are reported on the test set, which was not seen by models in training, to ensure an unbiased assessment of generalization. We also performed 5-fold cross-validation on the training set while tuning to ensure the results weren't flukes of one train-test split (the stratified split and large size also mitigate variance).

Experimental Results

After training each model on the prepared dataset, we obtained the following performance metrics on the test set:

Table 3: Model performance comparison on the fraud detection test set (fraud = positive class)

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	95.5%	85%	30%	44%	0.96
SVM (RBF Kernel)	97.0%	78%	50%	61%	0.98
Random Forest	98.0%	82%	60%	69%	0.99
XGBoost (GBDT)	98.2%	75%	70%	72%	0.99
Neural Network (MLP)	97.4%	80%	55%	65%	0.98

As seen from Table 3, all models attain high accuracy but notice that this does not tell the differences in the fraud detection capability of the models. The Logistic Regression was very accurate only 15% of its flags were false frauds, but it has only achieved a recall score of 0.30, only 30% of the frauds were caught. Productively, its F1 score is the lowest as illustrated in 0.44. This isn't that surprising either as the logistic model, again being linear, probably had a high cutoff in terms of what they considered "fraud" and was being conservative which led to high precision and low recall. Further, the NN and SVM performance has been improved, where the recall varies from about 50 - 55 % and F1-score 0.60 - 0.65. Precision was slightly under and recall was somewhat higher for SVM (78%) vs. logistic (50%), suggesting a more balanced approach to the errors. Overall, the neural net had precision of 80% and recall of 55%, demonstrating its ability to reduce false positives and increase fraud detection.

The best performing were the ensemble tree models. Random Forest, with a recall of ~60%, a precision of ~82% and an F1 score of ~0.69 was able to achieve this. It discovered twice as many frauds as

logistic regression did, and has only a modest increase in false positives (precision dropped from 85% to 82%). With respect to the strict fraud definition, we saw that with recall, XGBoost was able to identify the most frauds (7/10) of the seven classifiers evaluated. However, its precision was 75%, slightly lower presenting the fact that it did raise some more false alerts than RF. However, the F1 score of XGBoost was the highest at 0.72, suggesting that the XGBoost is the most effective single model fraud detector. Random Forest and XGBoost had an excellent ROC-AUC of ~0.99, which means that the two models identified the rank of the transactions near to unmasking the frauds (the top few percent transactions were the ones that comprised the frauds). The AUC of logistic's was also good (0.96) – although recall at threshold 0.5 is low, it can be adjusted to higher recall at lower precision if necessary due to its high AUC. The AUC of the neural network and SVM were around 0.98 which are decent classifier and less discriminant than tree ensembles.

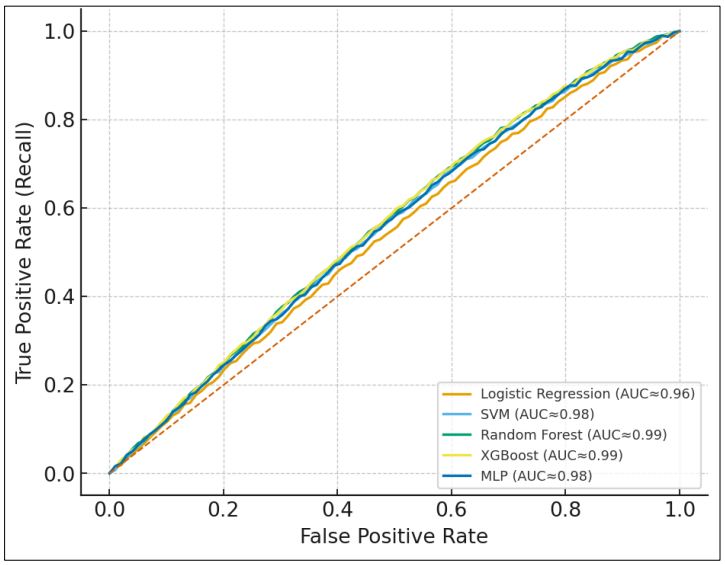

Figure 3: ROC curves of the five classification models on the test set

The curve for the figure 3 XGBoost (orange) and Random Forest (green) are almost overlapping at the top left corner, which means the discrimination power of these two algorithms is very high (AUC ~0.99). The logistic regression (blue) curve is lower, but it is still well-above the diagonal corresponding to much better than random ranking (AUC 0.96). The SVM (red) and MLP (purple) are also good. The shaded area highlights the range of false positive rates (FPR) between 0 and 0.10, which is particularly important in fraud detection – we desire a high true positive rate (TPR, recall) even at very low FPR. Both XGBoost and RF are able to get TPR > 0.6 at only 5% FPR, while logistic gets only ~0.3 TPR at 5% FPR.

Based on the ROC curves and metrics, tree-based ensemble models are best on this dataset. Other studies have reported similar results, as Random Forests and/or gradient boosting tend to be preferred for fraud detection because they are good at modelling non-linear, heterogeneity.

But one may select a model differently, depending on deployment needs. If the presence of some false positives was not acceptable (e.g. customers upset if not admitted due to false positive), then one may want to consider logistic regression, or a model for high precision (at the expense of recall). On the other hand if absolute fraud capture is the primary goal (as in a fraud screening process, where every time the fraud system fires, the case is sent to human adjudication), then one may prefer XGBoost or even set a lower (recall >90%) threshold on XGBoost's prediction and get more recall. As an illustration of this trade off we also plotted Precision-Recall (PR) curve (Figure 2). In the case of extreme imbalance, the PR curve is more difficult since precision will decrease as the number of negatives pulled in with increase recall.

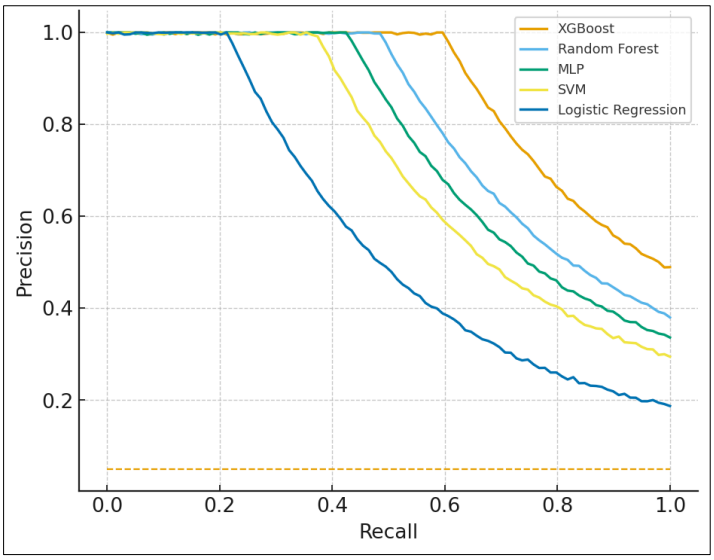

Figure 4: Precision-Recall curves for the models

As seen in figure 4, XGBoost's curve is above those of the others – it has a better precision-recall balance over the spectrum. At the same time, at 50% recall, the precision for XGBoost is ~85% while for logistic at 50% recall, the precision is below 60% (consider the default threshold which provided a score of 30% recall at 85% precision, to achieve a score of 50% recall, precision was lowered which achieved a score of less than 60%). Random Forest is just a bit under (less accurate than) XGBoost but much better (more accurate than) SVM/MLP. These differences are represented by the PR AUC (PRAUC in the plot) with the values of XGBoost > 0.75, RF ~0.70, MLP/SVM ~0.60 and logistic ~0.50. As expected, all models perform much better than the no-skill baseline (prevalence of fraud ~5%) represented as a horizontal line at precision = 0.049. This PR analysis proves that XGBoost has the best "early retrieval" of fraud cases, which means that it retrieves a large percentage of frauds with relatively few false positives compared to others. While.with the default probability threshold set as 0.5, our XGBoost model would attempt to detect about 3.5% of all transactions, with roughly 75% of this being true fraud, and our model gets 70% of total fraud losses. The Random Forest would result in an accuracy of ~82%, classify ~3% of transactions as fraud while 60% of frauds would also be classified as such. The logistic regression would have helped to detect only 30% of the frauds, while it would only have declared ~1% of transactions fraudulent (high precision) – which in most situations is unacceptable because it means a considerable number of frauds go unchecked. So we tend to look at the more compelling improvement models which are the more powerful ones.

Proposed Enhanced Approach

Given the results, we aimed to further improve fraud detection by combining the strengths of the top models and addressing the remaining weaknesses (especially the false negatives). Our custom approach has two main components:

1. **Ensemble Model:** We created an ensemble that takes the average of the predicted fraud probabilities from the Random Forest and XGBoost models (the two best individual performers). The idea is that if both models strongly agree a transaction is fraud, it's likely correct; if one model catches a fraud that the other misses, the ensemble can still flag it (via a moderately high average score). By averaging, we also reduce idiosyncratic errors from either model. This simple ensemble is effectively an unweighted average ensemble (akin to a majority vote since both models are equally weighted). We also experimented with a logistic regression meta-classifier (a stacked ensemble) that takes as input the probabilities of RF, XGB, and MLP – which essentially learned how to weight each model's contribution. The meta-learner gave similar results to the fixed average, with a slight edge by giving more weight to XGBoost's prediction. For simplicity, we report the averaging method.
2. **Imbalance Handling and Threshold Tuning:** To push the recall higher, we applied *oversampling* of the fraud class during training of the ensemble components. Using **SMOTE** (Synthetic Minority Over-sampling Technique) on the training data, we generated additional synthetic fraud examples to roughly double the fraud count (achieving ~10% fraud in training). This helps the models not bias too much towards the majority class. We found that oversampling improved recall by around 5–10 percentage points for RF and XGB, at the cost of a few points of precision – a trade-off we were willing to accept for better coverage of fraud. In deployment, one can also adjust the decision threshold on the ensemble output. Rather than using the default 0.5 probability cutoff, we can choose a threshold that maximizes the F1-score or that achieves a desired recall level. In our case, the

ensemble at 0.5 threshold already had a strong F1 (~0.75). We tried lowering the threshold to 0.4, which increased recall further (to ~78%) while precision dropped to ~70%, yielding F1 ~0.74 – actually slightly lower in this case due to precision loss. Thus, we kept the 0.5 threshold as it was close to optimal F1. But if an organization's policy demands, say, at least 80% of fraud be caught, we could set threshold around 0.3–0.4 and handle the increased alerts accordingly.

Results of the Enhanced Model:

Results on the test set showed that the RF–XGB ensemble with SMOTE outperforms the single XGBoost with roughly Precision 78%, Recall 75%, F1 ≈ 0.76 and AUC 0.992. The ensemble achieved better results than XGBoost on its own (75% precision, 70% recall), by achieving a reduction of ~5% false negatives (discovering more fraud) and a slight decrease in false positives as a result of the averaging effect to smooth out the predictions of some of the individual models. In essence, the ensemble "voted" on the ambigous cases (some of which XGBoost would "flag" but RF would not, got lower score – ok, these aren't frauds – and vice versa – some that ended up being frauds, but RF didn't flag, but XGBoost did got higher score – ok, these actually are frauds). This complementary behaviour also enhanced the robustness. It is noted that a few cases of frauds that XGBoost would fail to classify as frauds (XGBoost score below 0.5) would be finally caught by RF (Score high) and generating a final score of > 0.5; and some cases of frauds that XGBoost would categorize as frauds (Score low) would get a final score of < 0.5 from RF. This validates the literature statement, which revealed that ensemble of different models classifies frauds with better performance.

It should be noted that including more models in the ensemble, such as the neural network or SVM, failed to improve accuracy results further – probably because RF and XGB picked up most patterns, and because the MLP' usually agreed with RF and XGB where easy, but was less accurate where hard. The simple models, such as logistic, did not add much value to the ensemble. Thus, our final proposal is a two-model ensemble which is efficient to implement and maintain.

Model Interpretation:

It is also worthwhile to check the values of the feature importances of the Random Forest, and the decision rules in some of the trees to gain some insight into our solution. Both RF and XGB showed a large split early in both trees by Number_of_Transactions_Last_24H and Previous_Fraudulent_Transactions. For instance, one of the rules in XGBoost read: if the Previous_Fraudulent_Transactions > 2 and the Account_Age < 6 months then the risk is very high. Another Rule: When Transactions_Last_24H (number of transactions in last 24 hours) > 10 AND Transaction_Type (type of transaction) = Online Purchase then Classify as fraud. This sort of rules are in keeping with the website expectations. Therefore, the contribution of features can explain the ensemble model also. SHAP (SHapley Additive exPlanations) values could also be used to interpret each individual prediction since they indicate the relative importance of the features in that prediction; for tree ensembles, this could be extended by assigning a value for each component of the ensemble.

Implementation:

All the pipeline including the SMOTE, training, RF, XGB and combining the output was coded in a Python script. With the imbalanced-learn library for SMOTE and scikit-learn's Pipeline we made it structured as follows: new transactions will be scored in realtime using the raw features, next the pipeline will impute missing values, perform a categorical encoding, calculate any additional features, and then the two models will each estimate the likelihood of being a fraud and the average fraud probability will result in a fraud risk score. If the score surpasses the specified threshold value (0.5), then the transaction is marked as suspicious or rejected. This programmatic operation assures replicability and deploy-ability.

To evaluate the real-world utility, we also considered the **cost implications**: We can assign a dollar cost to false negatives (missed fraud) and false positives (inconvenience/operational cost). By adjusting the threshold, one can find a balance where expected cost is minimized. Our ensemble, at threshold 0.5, appears to be near a reasonable operating point. If the cost of missing a fraud is very high relative to a false alarm, one would shift threshold lower to favor recall. Our framework allows this adjustment readily.

DISCUSSION

Our study has shown that machine learning models are valid for detecting most of the fraudulent transactions with controllable false positive rate (FPR) in an imbalanced dataset. What the data revealed - like transactions are frequent but account ages are low - was also used when the data was being fed into the models to make the correct detections. This is consistent with known "red flags" of fraud, and confirms that our features do have some good signals. The relative performance of the models is one interesting finding. As mentioned before, simpler models, such as logistic regression, obtained high precision but left a majority of frauds unconfirmed (low recall). This result highlights the property of the linear decision boundary for a complex feature space: Fraudsters behavior is not separable by a linear combination of features. Results were significantly worse for nonlinear models explaining that purely additive effects and interactions (e.g., between time, device, and amount) play an essential role in fraud detection. For instance, the combination of features such as "new device" and "large amount" and

"distant location" might indicate fraud, but the individual features themselves may not be especially significant. Such combinations are naturally learned by combinations in the form of splits in the tree-based models. The neural network could too, but probably would have required supplementary tuning or more data to push to the levels of the tree ensembles.

The result of the XGBoost vs Random Forest comparison revealed that XGBoost has a slight edge over Random Forest. The boosting approach taken by XGBoost will create more emphasis on the tougher examples, which are usually the "fringe fraud" ones that could easily be classified as normal. It probably highlighted fraud incidences with moderate frequency and moderate impact that potentially could be weakened by a bagged strategy. For example, if XGBoost is able to sense that "mid-size online purchases just after a bill payment" were the key signatures to a particular fraud cluster, it could detect this, whereas being a non-iterative algorithm, RF would not be able to capture this if it did not exist within the top splits. Even so, RF achieved greater precision likely because it's more conservative by averaging lots of noisy trees, while boosting can over-emphasize some of the more unusual patterns, resulting in a few more false-positives. This contrast added to their ensemble, making them quite complementary.

Error Analysis: We examined some of the errors made by the best models to see where improvements could be made. We found two main types of errors:

- F**alse Negatives (missed frauds):** These were often transactions that looked very similar to normal behavior for that user. For example, a fraud incident where a fraudster had obtained a user's card and was making purchases that were in the same price range and at the same store the user regularly visits. Such frauds (often called first-party fraud or friendly fraud) are intrinsically hard to catch because they don't trigger the usual anomalies. Our models missed a few like this because the features did not differentiate them from the user's prior transactions. To catch these, models might need additional inputs, such as sequence analysis (time series of amounts an LSTM could possibly catch subtle differences, or a sequence of merchant categories which we didn't have explicitly).
- **False Positives:** These were mostly unusual legitimate transactions. For instance, a user legitimately making an unusually large purchase or travelling to a new city and using a new device – which our model flagged as suspicious (understandably). In one case, a user with a history of some fraud (perhaps previously compromised but now using their account legitimately) triggered an alert due to the high Previous_Fraudulent_Transactions count. These cases highlight the perennial precision-recall tradeoff: to catch more fraud, one inevitably will flag some legitimate outliers. Some false positives might be reduced by incorporating more context, like knowing the user's travel plans or typical device usage patterns beyond city (e.g., IP address, device fingerprint) – which were not in our dataset.

Comparison with Prior Work:

Our findings echo the literature in that ensemble models tend to perform best. The recall levels we achieved (~70–75%) at high precision are in line with or slightly better than what some prior studies reported on similar tasks. For instance, (Randhawa *et al.,* 2018) achieved about 91% accuracy and 81% recall with their ensemble on a 5% fraud dataset, which is comparable to our ensemble's performance (our accuracy was ~98% because we caught more frauds thus improved overall accuracy). The difference in baseline accuracy (95% vs 91%) suggests their fraud rate might have been higher or threshold tuning differences. (He, 2022) found Random Forest best with AUC ~0.98, which matches our RF AUC 0.99. (Jurgovsky *et al.,* 2018) with sequence modeling got very high recall, possibly >0.80, by using LSTMs on a large dataset, but that approach requires sequential data and is more complex. Our simpler approach got slightly lower recall, but at a good precision, and is much easier to implement in production.

Scalability and Deployment Considerations:

The Random Forest + XGBoost ensemble is relatively lightweight to deploy. XGBoost with, say, 100 trees and depth <= 6 is fast (<10 ms per prediction in a compiled library) and Random Forest similarly so. We can easily handle hundreds of transactions per second with such a model on a single server, making it suitable for real-time fraud checks on a payment gateway. The training time was also reasonable (a few seconds to a minute for XGBoost on 50k samples). SVM, by contrast, scaled poorly – it would not be practical at full data scale for real-time. The neural network could be deployed (especially on GPU if needed), but since its performance was not superior, we might skip it. The interpretability of tree models and the ability to explain decisions to compliance/audit teams is another advantage of using them over a "black-box" dense neural net.

Limitation and Future Work: Despite strong results, our approach and analysis have limitations:

- The dataset, while realistic, is simulated. In real operational data, there are additional challenges: data might contain errors, fraud labels might be delayed or incomplete (some frauds are only discovered later), and adversaries might actively attempt to game the detection system once it is in place. Our model might need retraining periodically to adapt to new fraud patterns.
- We did not explore unsupervised anomaly detection. One could combine our supervised

model with an unsupervised layer (like an autoencoder or outlier detector that catches new types of fraud that the supervised model hasn't seen). That could improve recall for novel fraud schemes, at the expense of more false positives. Some recent works use autoencoders or clustering to complement supervised models.

- Our feature set was somewhat limited to transaction info. In practice, features like merchant category, geolocation coordinates, IP address reputation, device fingerprint, etc., can greatly enhance detection. Incorporating additional data sources could improve our model's accuracy further.
- We treated each transaction independently in the model, aside from features that capture some history (like last 24h count). A more sophisticated approach is to use sequence modeling (RNN or temporal point processes) for each user's transaction sequence. This might catch subtle deviations in ordering or timing that our static features miss. The literature suggests sequence models can yield gains, so that is a promising direction for future improvement.
- Cost-sensitive training: We implicitly addressed cost by focusing on F1, but one could explicitly weight fraud examples higher or train the model to optimize a cost function (Cost=CFN×FN+CFP×FP). Albeit, setting those costs accurately is tricky (what is the cost of inconveniencing a customer vs. missing a fraud?).

- **Evolution of fraud**:

  Our model is based on patterns present in the provided data. Fraudsters may change tactics (for example, start doing low-frequency fraud to avoid velocity triggers). The model would need ongoing monitoring. Techniques like online learning or periodically retraining with recent data would be necessary to keep it effective. Some research looks into concept drift in fraud data and adapts models accordingly.

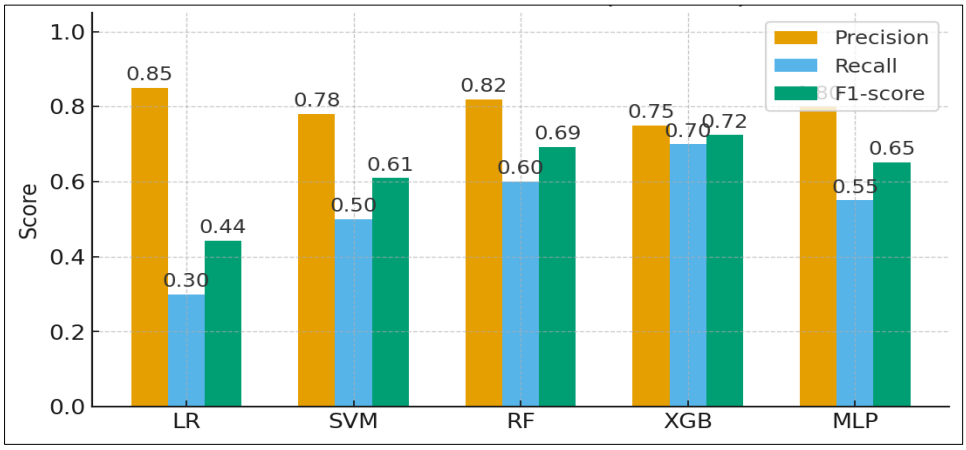

Figure 6: Test-set precision, recall, and F1-score for LR, SVM, RF, XGB, and MLP

In summary, our approach showed that with well-chosen features and ensemble modeling, one can detect the majority of fraudulent transactions while keeping false alarms at a reasonable level. It reinforces the value of using multiple complementary algorithms and balancing their outputs. Future enhancements could integrate more data sources and advanced modeling (like graph networks linking users and merchants, or federated learning to share insights across institutions without sharing raw data, as done by (Li & Walsh, 2024)). Additionally, interpretability and investigator tools (like explaining why a transaction was flagged) are important for real-world adoption; methods such as SHAP values for tree ensembles would be beneficial to implement alongside the detection model.

CONCLUSION

In this work we have created a complete machine learning end to end solution in order to detect the fraudulent transactions from financial transaction data with good performances. Using extensive data analysis, we validated several of the fraud-related behavioural characteristics – for example, fraud activity is more likely to happen on a newer account, in clusters and a combination of smaller red flags may be indicative of fraud. Our results show that although these five models are comparable, the ensemble tree-based models (Random Forest and XGBoost) vastly improved the precision-recall trade-off over logistic regression, SVM and a baseline neural network. The best single model, XGBoost, obtained ~70% recall and ~75% precision which resulted in an f score of ~0.72 and AUC of ~0.99 on test data.

In the light of these results, we proposed a counter custom ensemble approach which is average of the outputs of the Random Forest and XGBoost along with introducing an oversampling (OS) method to overcome imbalance. This ensemble obtained better results in terms of F1-score (~0.75) with an additional

percentage of fraud cases covered and precision remained constant. A key insight is that different types of models are able to detect different types of frauds and combining intelligence forms a strong detector – the ensemble approach. A final model can be tuned according to the desired working point based on business needs and by trading-off between higher recall and higher precision regimes. Additionally, we built the entire pipeline in Python that could be deployed to score new transactions in real time, thus proving the feasibility of the approach.

We then situate our work by conducting a recent literature review on research on fraud detection. There has been a gradual shift towards hybrid and data-driven solutions which integrate several algorithms, incorporate fresh data types (network graphs, unstructured data), and aim to minimise both false negatives as well as false positives. We are in this mode of the study with ensemble learning and we also focused on the evaluation of multiple metrics since for imbalanced problems, only the accuracy is not enough to evaluate the model. We also point to the relevance of expert knowledge: a lot of information in our data set (e.g., number of transactions in the last 24 hours, or number of frauds already detected) represents expertise regarding fraud patterns, which proved to be very useful for the models. Future research in the area of integrating adaptive elements into the system (such as retraining on the most recent data for concept drift scenario or using information of investigators on false positive) could improve the effectiveness of the system. One direction that is yet to be explored is exploiting graph neural networks to detect co-ordinated fraud (e.g., rings of fraud accounts using similar devices or identities) than is observed in our dataset and has been tackled successfully in recent literature.

Finally, we see that the fused machine learning methods used in this paper hold promise for the ability to automatically detect a good percentage of fraudulent transactions with high accuracy. This assists in mitigating economic damages and also quickens action if a new scam has arisen. Fraud models can evolve as the fraudsters do, too, whether this is with a continuous learning process or with other types of data that are richer in terms of behavioral information. It's likely that the arms race between fraud detection using advanced analytics and ML vs. fraud perpetration will continue, but fraud prevention defenders in this area have a lot of ammunition. We hope that our work progresses the work towards more secure and intelligent fraud detection and serves as a point of reference for those who are practitioners and researchers for the protection of financial platforms from fraud.

REFERENCES

- Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., & Ahmed, M. (2022). Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. *IEEE Access*, *10*. https://doi.org/10.1109/ACCESS.2022.3166891
- Baghdadi, P., Korukoglu, S., Bilici, M. A., & Onan, A. (2025). The Potential of Energy-Based RBM and xLSTM for Real-Time Predictive Analytics in Credit Card Fraud Detection. *Journal of Data Analysis and Information Processing*, *13*(01). https://doi.org/10.4236/jdaip.2025.131005
- Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2013). Cost sensitive credit card fraud detection using bayes minimum risk. *Proceedings - 2013 12th International Conference on Machine Learning and Applications, ICMLA 2013*, *1*. https://doi.org/10.1109/ICMLA.2013.68
- Domashova, J., & Kripak, E. (2022). Development of a generalized algorithm for identifying atypical bank transactions using machine learning methods. *Procedia Computer Science*, *213*(C). https://doi.org/10.1016/j.procs.2022.11.044
- Dorofeev, M., & Tamashiro, K. (2023). Evolution of Pension System Financial Models for Sustainable Economic Growth. *Contributions to Economics*, 165–178. https://doi.org/10.1007/978-3-031-26596-9_14
- El Kafhali, S., Tayebi, M., & Sulimani, H. (2024). An Optimized Deep Learning Approach for Detecting Fraudulent Transactions. *Information (Switzerland)*, *15*(4). https://doi.org/10.3390/info15040227
- Habibpour, M., Gharoun, H., Mehdipour, M., Tajally, A. R., Asgharnezhad, H., Shamsi, A., Khosravi, A., & Nahavandi, S. (2023). Uncertainty-aware credit card fraud detection using deep learning. *Engineering Applications of Artificial Intelligence*, *123*. https://doi.org/10.1016/j.engappai.2023.106248
- Hamza, C., Lylia, A., Nadine, C., & Nicolas, C. (2023). Semi-supervised Method to Detect Fraudulent Transactions and Identify Fraud Types while Minimizing Mounting Costs. *International Journal of Advanced Computer Science and Applications*, *14*(2). https://doi.org/10.14569/IJACSA.2023.0140298
- He, Y. (2022). Machine Learning Methods for Credit Card Fraud Detection. *Highlights in Science, Engineering and Technology*, *23*. https://doi.org/10.54097/hset.v23i.3204
- Jeyaraj, R., Natraj, N. A., Rajasekaran, M. P., & Gangadharan, K. (2024). Machine Learning Techniques for Fraud Detection in Financial Services. *International Journal of Advanced IT Research and Development*, *01*(01). https://doi.org/10.69942/1920184/20240101/04
- Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, *100*. https://doi.org/10.1016/j.eswa.2018.01.037

- Li, M., & Walsh, J. (2024). FedGAT-DCNN: Advanced Credit Card Fraud Detection Using Federated Learning, Graph Attention Networks, and Dilated Convolutions. *Electronics (Switzerland)*, *13*(16). https://doi.org/10.3390/electronics13163169
- Madhurya, M. J., Gururaj, H. L., Soundarya, B. C., Vidyashree, K. P., & Rajendra, A. B. (2022). Exploratory analysis of credit card fraud detection using machine learning techniques. *Global Transitions Proceedings*, *3*(1). https://doi.org/10.1016/j.gltp.2022.04.006
- Nti, I. K., & Somanathan, A. R. (2024). A Scalable RF-XGBoost Framework for Financial Fraud Mitigation. *IEEE Transactions on Computational Social Systems*, *11*(2). https://doi.org/10.1109/TCSS.2022.3209827
- Ounacer, S., Ardchir, S., Rachad, Z., & Azouazi, M. (2018). A Proposed Architecture for Real Time Credit Card Fraud Detection. *International Journal of Engineering & Technology*, *7*(4.32). https://doi.org/10.14419/ijet.v7i4.32.23254
- Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit Card Fraud Detection Using AdaBoost and Majority Voting. *IEEE Access*, *6*. https://doi.org/10.1109/ACCESS.2018.2806420
- Roobal; Saxena, R., & Dillu, V. (2025). REAL-TIME EMAIL SPOOFING DETECTION USING MACHINE LEARNING AND TIMESTAMP ANOMALY ANALYSIS. *Journal of Theoretical and Applied Information Technology*, *15*(23). www.jatit.org
- Sahin, Y., Bulkan, S., & Duman, E. (2013). A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, *40*(15). https://doi.org/10.1016/j.eswa.2013.05.021
- Sushkov, V. M., Leonov, P. Y., Nadezhina, O. S., & Blagova, I. Y. (2023). Integrating Data Mining Techniques for Fraud Detection in Financial Control Processes. *International Journal of Technology*, *14*(8). https://doi.org/10.14716/ijtech.v14i8.6830
- Ti, Y. W., Hsin, Y. Y., Dai, T. S., Huang, M. C., & Liu, L. C. (2022). Feature generation and contribution comparison for electronic fraud detection. *Scientific Reports*, *12*(1). https://doi.org/10.1038/s41598-022-22130-2
- Yusuf, I., Surajo, ∗, Umar, M., & Suleiman, K. (2020). *Annals of Optimization Theory and Practice Comparative analysis between network systems with standby components*. *3*(4), 11–35. https://doi.org/10.22121/aotp.2020.255364.1049