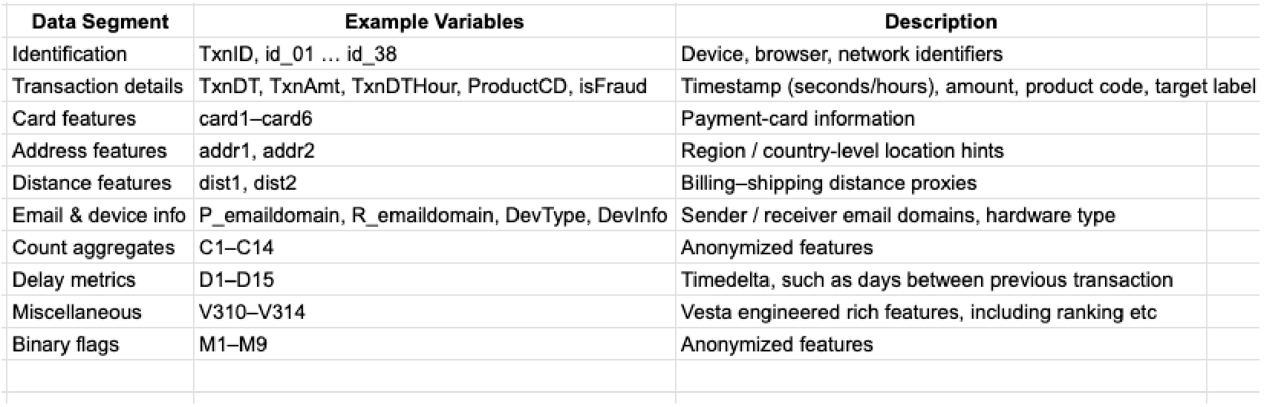
Introduction

As the digital payments industry has grown, it has generated enormous amounts of transactional data essential for detecting fraudulent behaviors. This report/mini case study presents a detailed **Exploratory Data Analysis (EDA), and Machine Learning Modeling** workflow on one such example taken from the IEEE-CIS Fraud Dataset in order to develop a robust, deployable fraud detection framework, especially for credit card frauds. The primary objective of this case study is to *i) to prepare the dataset for modeling that reflects both benchmark and deployment realities, ii) address any class imbalance without distorting temporal structure, iii) to build and compare multiple models, iv) to interpret the drivers of model decisions, and v) to articulate recommendations, including threshold governance and non-data fraud controls.*

Dataset Overview

The provided dataset consists of **100,000** payment events across **101** variables. Each record represents a unique transaction labeled by the target variable **isFraud** - 1 for fraudulent and 0 for legitimate. An initial review revealed that **11.3%** of all records are fraudulent transactions, confirming a highly imbalanced dataset which is consistent with real-world financial data distributions. Dataset contains the following main categories:



Data Segment	Example Variables	Description
Identification	TxnID, id_01 … id_38	Device, browser, network identifiers
Transaction details	TxnDT, TxnAmt, TxnDTHour, ProductCD, isFraud	Timestamp (seconds/hours), amount, product code, target label
Card features	card1–card6	Payment-card information
Address features	addr1, addr2	Region / country-level location hints
Distance features	dist1, dist2	Billing–shipping distance proxies
Email & device info	P_emaildomain, R_emaildomain, DevType, DevInfo	Sender / receiver email domains, hardware type
Count aggregates	C1–C14	Anonymized features
Delay metrics	D1–D15	Timedelta, such as days between previous transaction
Miscellaneous	V310–V314	Vesta engineered rich features, including ranking etc
Binary flags	M1–M9	Anonymized features

Data Cleaning and Preparation

The raw dataset required heavy cleaning hence the following steps were taken.

Missing-Value Profiling
A column-wise missing value analysis identified several fields - particularly the **M series** having over **90% missing values** (refer to appendix: figure 2 for a quick look at the missing value distributions). These variables were removed due to their limited discriminative power and potential to bias model training.

Removal of Constant/Low-Variance Features
Columns with nans, empty or only single values were removed. Similarly fields that had a high variance among themselves and no clear relation with the **isFraud** column were dropped as well, refer to figure 3 and 4 in appendix. These variables provide no information entropy for classification.

Domain Knowledge Based Pruning
Using domain knowledge, features dominated by placeholder or redundant values were excluded to enhance signal quality and prevent noise inflation in modeling. Final pruning reduced dimensionality from **101 to 58** useful features.

Imputation Strategy
The following strategies were used based on data type and distribution:

- For numeric features with skewness < 1, mean values are used as they are nearly symmetrical.
- For numeric features with skewness >= 1, median values are used as they are heavy tailed.

- For categorical features, missing values are replaced with “Unknown” to preserve category integrity.

Outlier Profiling
Extreme values were identified using the IQR rule however they are not truncated as extreme values are indicative of fraudulent activity.

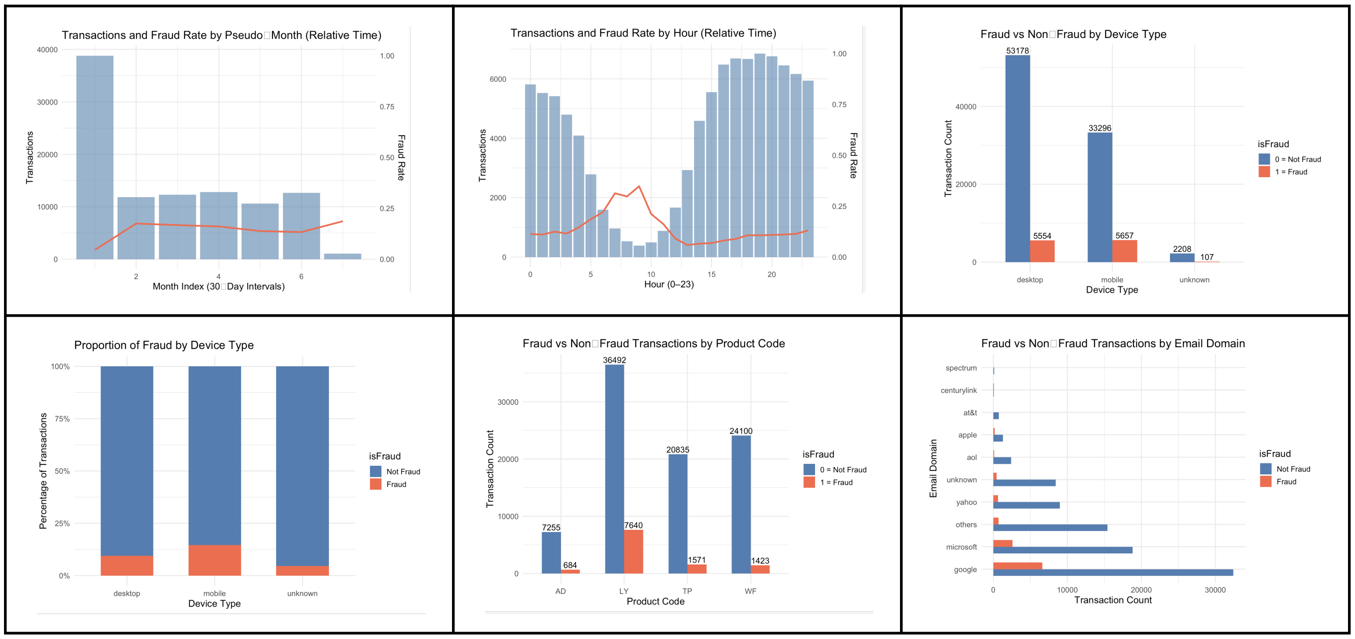
Exploratory Data Analysis

A univariate, bivariate and multivariate analysis was conducted to explore variance patterns and relationships within the dataset. Visualizations for **fraud rate relative to engineered time-date features** from the TxnDT column revealed temporal trends. Since the dataset lacked an explicit start date, pseudo-weekdays and months were constructed from TxnDt to identify temporal patterns. The analysis showed that fraudulent activity **peaks during low-traffic hours**, especially late at night and early in the morning. Over time, transaction counts declined while the fraud rate increased, suggesting concept drift or evolving fraud tactics.

Device-level analysis indicated that mobile and desktop transactions occur in similar volumes, but, fraud is proportionally higher on mobile, aligning with industry findings that mobile devices are vulnerable due to weaker identity verifications and risks such as **SIM-swap** attacks**.** Product analysis revealed that the **‘LY’ product code** had disproportionately high fraud count, while email domain patterns depicted that fraudulent transactions were more frequent among ‘Google’ email domains.

Geographic fields also provided insights such as addr1 appeared to represent regions and addr2 countries, with most transactions originating from a single country (code **73).** Furthermore, transaction amount analysis showed overlapping ranges between legitimate and fraudulent cases, though fraud tended to skew slightly toward higher values.

Correlation matrices and pairwise plots also highlighted strong dependencies among several features such as D1 and D2, confirming redundancy and inter-relatedness. These relationships also revealed that fraudulent observations slightly distort normal feature correlations, suggesting nonlinear interactions are key to detecting fraud. Some of this data is summarized in the following picture below with graphs and for further graphs, please see appendix:



Feature Engineering

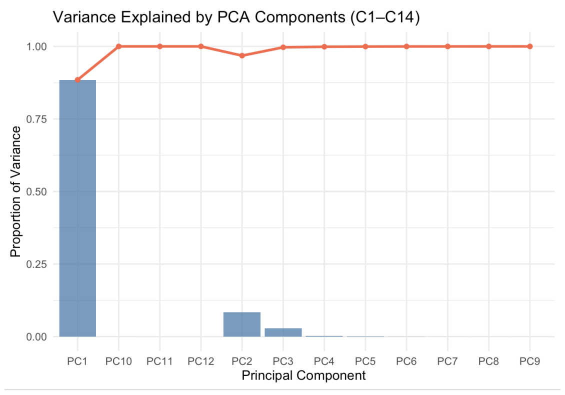
Various feature engineering techniques were used so that raw data can be transformed into variables that directly capture behavioral mechanisms of fraud.

For monetary features, they are transformed as the following:

- Log1p transform was done to stabilize variance and handle magnitude differences resulting in the TxnAmt_log feature.
- Standardized z-score method was used to ensure comparability for distance-based models resulting in the TxnAmt_z feature.

For temporal behavior features, the TxnDt and TxnHour fields were transformed to derive several time-based variables capturing transactional patterns. TxnDay and TxnHour represent periodic activity, while DayOfWeek models weekly cycles, and IsWeekend captures leisure-time spending behavior. To enhance interpretability, hours were grouped into HourBucket categories (night, morning, afternoon, evening), and a binary flag IsNightTxn to compress risk of overnight activity. Furthermore, email related features were also engineered features such as SameEmailFlag which indicates if purchaser and receiver email domains are matching and KnownDomainP and KnownDomainR to check if fraud probability is higher in known or unknown email domains. Moreover, the rationale behind having SameEmailFlag is that internal transfers within the same domain may be safer and transfers between different domains may signal phishing.

Lastly, a principal component analysis (PCA) was also carried out to compress correlated 'C' columns to reduce dimensionality and capture their shared variance. The result showed that the first principle component explained nearly all the variation, which can be seen in the graph below, denoting strong interdependence among these variables. Higher CPCA_1 values likely correspond to users with closely linked identities or repeated transactions behaviors. As CPCA_1 captures all the data provided by C columns, we removed all the C columns and used CPCA_1 as the final column to represent C columns.



EDA Summary

The EDA revealed that fraudulent transactions make up about **11.3%** of all data, showing significant class imbalance. Temporal trends indicated that fraud usually occurs during the low-traffic hours and the fraud rate increases overtime, suggesting concept drift or evolving fraud tactics. Mobile transactions showed a higher proportion of fraud than desktop, confirming that mobile channels are more vulnerable.

Fraud cases were also more frequent among users with **'Google'** email domains and linked certain products like '**LY',** hinting at systematic vulnerabilities and weaker identity verification. Transaction amounts for fraud overlapped with legitimate ones but leaned slightly higher. Feature analysis showed that many variables were highly correlated, and PCA confirmed that these could be compressed into single **(CPCA_1)** summarizing user-level behavioral patterns.

Data Preparation for Modeling

The modeling process begins from the post clean dataset comprising **100,000 transactions and 58 features**. The feature engineering from the EDA process was preserved and extended for modeling. However, some velocity and interaction features such as TxnAmt_per_Hour, Weekend_Night etc were introduced to target the behavioral facets of fraud such as abnormal activity during off-peak hours and email divergence.

In addition to that, 2 setups were used along with 4 different models to balance rigor with realism. For instance, firstly, a standard 80:20 split was done which stratified on '*isFraud'* offering a comparable benchmark. On the other hand, for the second setup a time-aware split sorted by TxnDT was used where the first 80% was used as training and the last 20% as test, better mimicking the way a production system predicts into the future under the potential concept drift. In both setups, all character predictors were converted to factors, and high cardinality geographical identifiers (addr1, addr2) were excluded from the model to avoid sparse expansions. Lastly, all the feature construction preceded a time-aware split to avoid any leakage.

Addressing Class Imbalance

Given that 11.3% positive rate, class imbalance had to be addressed carefully. In the random-split setup, **ROSE** was applied, which synthesizes samples to produce near balanced classes while preserving variability [1]. Post-ROSE, the training distribution was approximately **50% per class**. In the time-aware setup, which is setup 2, SMOTE was applied on training period only, transforming factors to numeric as required to synthesize realistic minority examples without peeking at the future. Moreover, when gradient boosting models were used for modeling in setup 2, *scale_pos_weight* was further specified based on the class ratio to bias the loss towards correctly classifying minority cases during training.

Models and Training Strategy

Modeling was divided into 2 setups/tiers where in each setups 2 different models were used. The first tier establishes baselines and in this setup, logistic regression (GLM) was used for interpretability and a non-parametric random forest was used to capture nonlinearity and interactions. These two models were both trained in setup 1, which used a random-split branch after ROSE balancing and evaluated on the untouched test fold. The second tier adopted gradient-boosted decision trees, XGBoost and LightGBM, and these were trained in setup 2, which used time-aware settings with SMOTE balancing along with early stopping and class weighting [2]. Furthermore, for boosted trees, label encoding for factor inputs and tuned core hyperparameters such as tree depth, learning rate, subsampling, and regularization was also used. Crucially, subsequently the classification threshold on the test horizon was optimized to maximize the F1, acknowledging that the default 0.5 threshold often underperforms in imbalance tasks where the relative costs of false positives and false negatives differ.

Furthermore, XGBoost and LightGBM models were used for this dataset because the dataset contains numerous mixed, engineered and sparse features. In addition to that, boosted trees handle non-linearities and high order interactions among these heterogeneous signals without extensive feature scaling and their tree-based splitting copes well with high-cardinality categorical encodings and skewed monetary/temporal features. Therefore, in practice, these models can deliver strong ranking performance under severe class imbalance and scale efficiency to millions of rows via histogram binning or via mature regularization and stable optimization. Hence, using these models reduces model risk and lets us select the stronger performer on the time-aware horizon.

Evaluation Metrics and Threshold Tuning

An emphasis on precision, recall and their harmonic mean F1, alongside PR-AUC was put on as overall accuracy is a poor guide under the imbalance dataset [1]. PR-AUC was chosen instead of ROC-AUC because PR-AUC summarizes ranking quality for the positive class and is more indicative than ROC-AUC in skewed settings. In addition to that, in order to align model operations with trade-offs, threshold sweeps between 0.1 and 0.90 were performed and selected operating points near the F1 peak. An inspection of the shape of precision and recall as functions of the threshold was also done to assess how much recall must be given up to gain precision in practice.

Results

The baseline models in the random-split branch demonstrated the non-linearity. The logistic regression achieved an accuracy of 0.715 with a precision of 0.248, a recall of 0.749, a F1 score of 0.373, and ROC-AUC of 0.811. Its high recall but low precision profile indicates a tendency to flag many legitimate transactions, which would strain manual review capacity. While for the random forest, the performance seemed to be improved to an accuracy of 0.791 with a precision of 0.330, recall of 0.821, F1 of 0.471, and ROC-AUC of 0.881. This baseline seemed to be substantially better balanced, reflecting the model's ability to capture interactions and non-linear signals that GLM cannot. All the information is summarized in the table below:

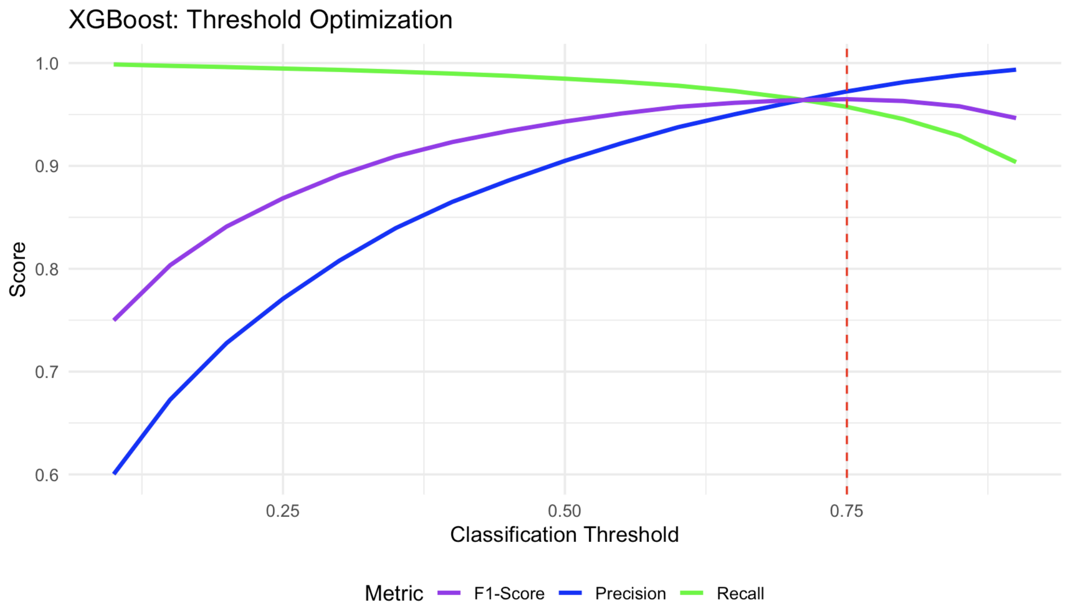
Models	Accuracy	Precision	Recall	F1	AUC
Logistic Regression	0.7149857	0.2482789	0.7490057	0.3729373	0.8111862
Random Forest	0.7909895	0.3299024	0.8214759	0.4707521	0.8808295

For the setup/tier 2 which was the time-aware branch, gradient boosting models offered a larger step up and a more realistic sense of future performance. With a default threshold of 0.5, XGBoost achieved accuracy of 0.840 with a precision of 0.443, recall of 0.749, F1 of 0.557, ROC-AUC of 0.889, PR-AUC of 0.664, MCC of 0.491 and a balanced accuracy of 0.802. For LightGBM, at the same threshold performed slightly better on most metrics: accuracy of 0.863, precision of 0.494, recall of 0.755, F1 of 0.597, ROC-AUC of 0.893, PR-AUC of 0.669, MC of 0.536, and balanced accuracy of 0.817. After threshold optimization, both models increased precision meaningfully while preserving acceptable recall. XGBoost at an optimal threshold near 0.75 delivered accuracy of 0.907, precision of 0.69, recall of 0.55, F1 of 0.61, PR-AUC of 0.664, and MCC of 0.560. LightGBM at a similar threshold achieved accuracy of 0.907, precision of 0.695, recall of 0.54, F1 of 0.613, PR-AUC of 0.669, and MCC of 0.570. Confusion matrices confirm that these operating points lift precision to roughly two-thirds while maintaining sensitivity near 0.60 and specificity around 0.95, an appealing trade-off when manual review capacity is finite. A model comparison of scores can be seen in the **appendix figure 20** and table below:

Models	Accuracy	Precision	Recall	F1	AUC_ROC	AUC_PR	MCC	Balanced Accuracy
XGBoost (thresh=0.5)	0.83965	0.4427095	0.7494415	0.5566155	0.8892100	0.6644286	0.4906390	0.8015430
LightGBM (thresh=0.5)	0.86340	0.4943902	0.7546538	0.5974064	0.8930406	0.6690229	0.5362694	0.8174620
XGBoost (thresh=0.75)	0.90710	0.6911357	0.5573343	0.6170651	0.8892100	0.6644286	0.5691284	0.7593475

LightGBM (thresh=0.75)	0.90720	0.6955702	0.5495160	0.6139767	0.8930406	0.6690229	0.5671050	0.7561026

The threshold curve, as seen in the figure below, illustrates the inherent tension, as the decision threshold increases, precision rises and recall falls, with F1 peaking in the 0.65-0.75 range. This behavior is desirable, because it allows operations to select an operating point that fits current capacity and risk tolerance: lowering the threshold captures more fraud at the cost of more false positives while raising it reduces friction but risks missing cases.



Model Interpretation

Across random forest, XGBoost, and LightGBM, the most influential features *(please see appendix figure 19 for top 20 features)* align closely with the hypothesis and findings during EDA [1]. Monetary measures - TxnAmt, its log and z-score variants, and a simple velocity proxy TxnAmt_per_Hour, rank consistently high, indicating that the magnitude and cadence of spending provide a strong yet nuanced signal once stabilized. Temporal variables such as TxnHour, isNightTxn, DayOfWeek, IsWeekend and others are repeatedly important, echoing the EDA's observation that fraud is concentrated at night and in low traffic windows. Channel and product context matter as well as DevType, ProductCD, and the Device_Product interaction help discriminate risky combinations.

Email-related features manifest strongly as well. For instance, Email_Mismatch between payer and payee and indicators for known consumer domains elevate risk which is consistent with phishing and mule activity patterns observed in EDA. The engineered Card_Email_Match also contributes, suggesting that the coherence between payment instrument and communication channel is informative. Finally, identity/time-delta variables emerge as meaningful signals, capturing rhythm and recency nuances. The PCA-based C_PCA1 compacts user-level behavioral consistency and identity linkage from the highly correlated C-series and its importance confirms that collapsing collinear blocks can preserve substantive signals while improving stability.

Robustness, Limitations and Future Improvements

Several methods in the modeling phase have improved confidence in the results produced. First, the use of a time-aware split avoids the optimistic bias that often creeps into random hold-outs when concept drift is present. This choice is also consistent with EDA findings that fraud rates evolve over time. Secondly, imbalance handling is layered i.e. SMOTE and ROSE provide a richer training distribution, while scale_pos_weight in boosting further adjusts the loss to the true class proportions.

Nonetheless, there remain some limitations and areas of improvement. For example, label encoding for high-cardinality categories can be improved via target encoding with nested cross-validation or by adopting CatBoost to natively handle categorical variables. Probability calibration (Platt scaling or isotonic regression) would yield scores more suitable for cost-aware decisioning and segmentation. Furthermore, as fraud patterns evolve, it is recommended to use rolling-window retraining, population stability monitoring for key features and a champion-challenger framework to continuously test alternatives. Finally, future optimization should explicitly encode economic cost by maximizing expected savings, not only F1.

Deployment and MLOps Considerations

It is recommended to use LightGBM as the initial champion model due to its slightly stronger F1 and MCC at similar recall along with efficient training and robust generalization in the time-aware test [3]. The production fraud detection system should also expose a calibrated probability. Decisioning should be tiered such as transactions below a lower threshold are auto-approved while a middle band triggers setup-up authentication methods such as biometrics and MFA while the high-risk band is held for manual review or declined depending on amount or customer profile. Thresholds should be context-sensitive which is that they are tighter during night hours and weekends or on the mobile channel and or 'LY' product - mirroring the strongest drivers found in EDA and modeling.

In terms of monitoring, it must include PR-AUC, F1, recall@top-k, false positive rates by segments, and population stability indices for top features. Alerts should trigger when drift or performance breaches predefined guardrails. Documentation, validation reports, and explainability artifacts should be maintained under model risk management standards.

Non-Data Analytic Measures to Prevent Fraud

No model alone is sufficient to curb fraud. It must be embedded within a broader control environment. The analysis suggests targeted risk-based controls. Because overnight and weekend activity carries elevated risk, step-up authentication should be mandatory in those windows for certain products and channels. Device integrity checks such as detecting rooted or jailbroken environments, and enforcing current app/OS versions along with cryptographic device binding can help reduce account takeover and synthetic identity abuse. Email and identity hygiene should include domain risk scoring along with additional friction on disposable or newly created domains and strict verification whenever payer and payee domains differ. Adaptive velocity limits should scale with account tenure and risk tier, and cooling-off period should follow changes to sensitive credentials or SIM swaps. Negative files for devices, emails and cards linked to confirmed fraud should be maintained and shared responsibly across merchants or lines of business where policy allows, with geofencing and merchant/product tiering for rapid mitigation during spikes.

Furthermore, governance is equally important. Segregation of duties between model developers and validators, auditable change control, and incident response playbooks ensure that controls can be tightened quickly during attacks. Customer facing communication helps as well such as timely push notifications of unusual activity and a one-tap “Not me” response materially decrease loss and improve customer trust. Finally, data privacy and regulatory alignment must be preserved, with explainability available for adverse action contexts.

Conclusion

This case study and modeling pipeline transforms EDA insights into a deployable fraud detection solution. Baseline models demonstrated the limits of linear separability, while gradient-boosted trees, trained with time-awareness, imbalance handling, and the tuned thresholds, deliver a meaningful uplift. Significantly, the model’s most influential features mirror the EDA narrative i.e. temporal off-peak risks, mobile/device and product interactions, consumer email domain patterns and mismatches, and identity coherence captured by C_PCA1, providing a coherent, explainable story across exploration and predictions. Coupled with targeted controls and robust MLOps, the proposed system offers a practical and scalable foundation for reducing fraud losses while managing customer friction.

References

[1] PDF arXiv:2105.06314v1 cs.LG 13 May 2021**(**https://arxiv.org/pdf/2105.06314)

[2] An Imbalanced Financial Fraud Data Model Based on Improved XGBoost and RUS Boost Fusion Algorithm with Pairwise (https://bcpublication.org/index.php/BM/article/view/5445)

[3] PDF A Bayesian Simulator for Payment Card Fraud Detection Research (https://www.federalreserve.gov/econres/feds/files/2025017pap.pdf)

Appendix

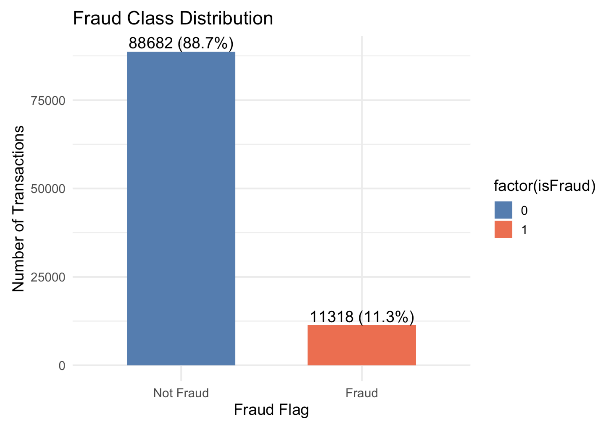


Figure 1: *Fraud class distribution*

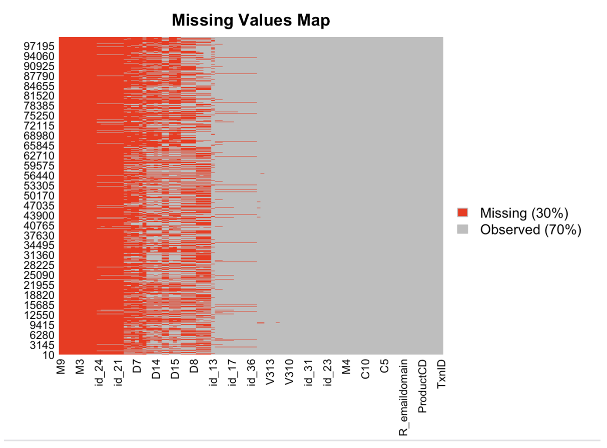
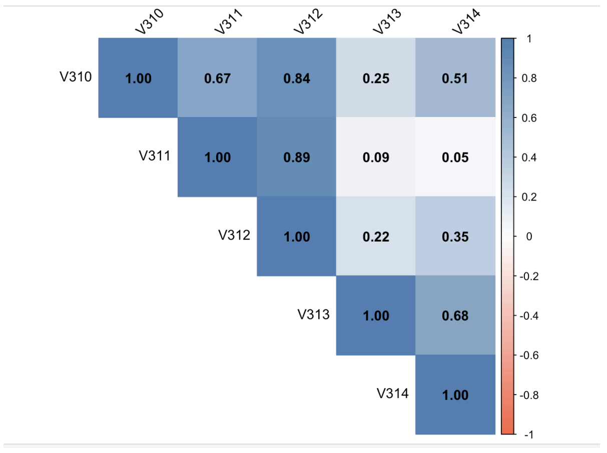


Figure 2: Missing values overview



	V310	V311	V312	V313	V314
V310	1.00	0.67	0.84	0.25	0.51
V311		1.00	0.89	0.09	0.05
V312			1.00	0.22	0.35
V313				1.00	0.68
V314					1.00

Figure 3: *Feature correlation matrix*

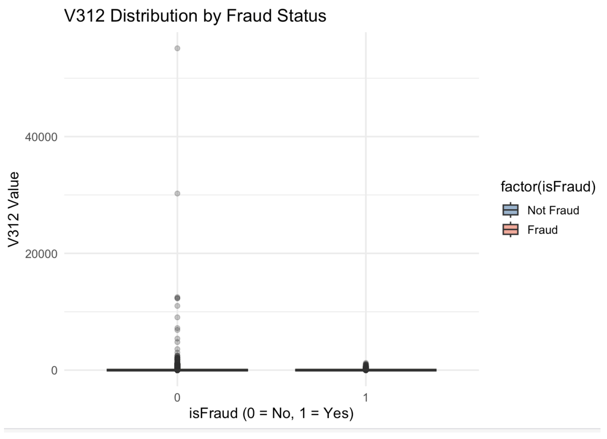


Figure 4: *V312 fraud comparison*

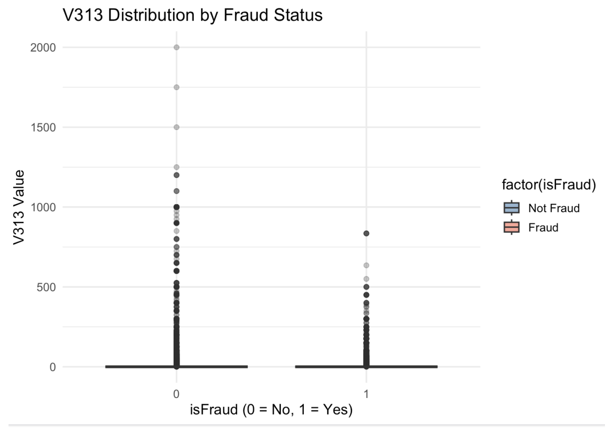


Figure 5: *V313 fraud comparison*

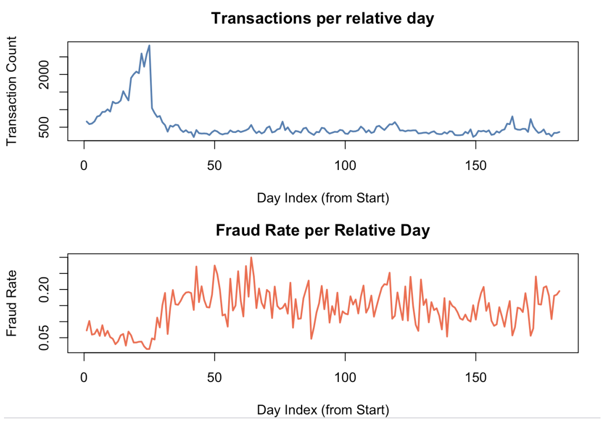


Figure 6: *Transactions and Fraud rate per day*

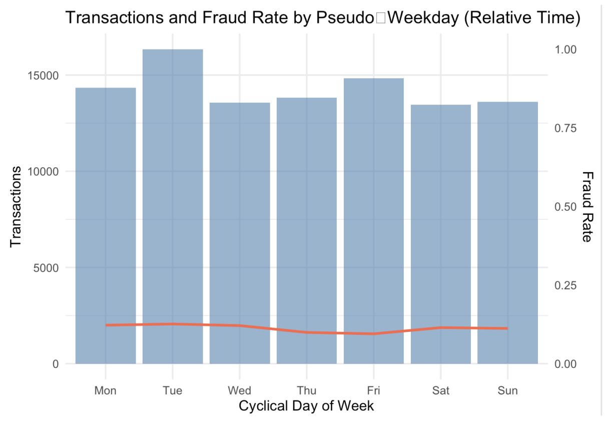


Figure 7: *Transactions and Fraud rate per weekday*

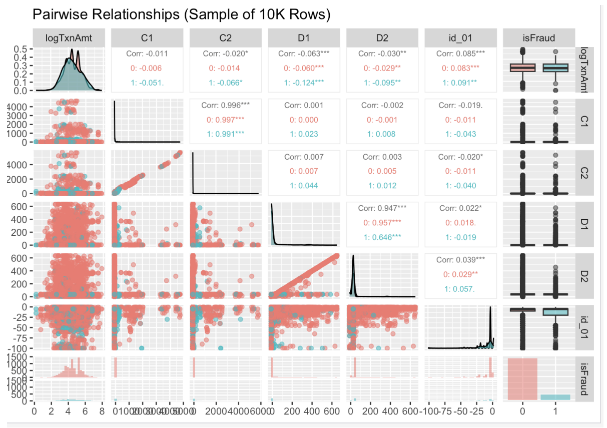


Figure 8: *Pairwise Relationship Matrix*

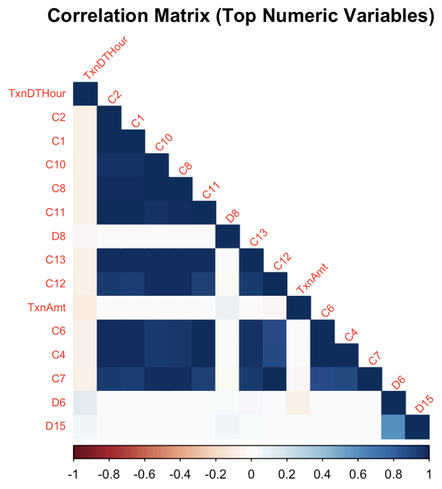


Figure 9: *Numeric Variables Correlation Matrix*

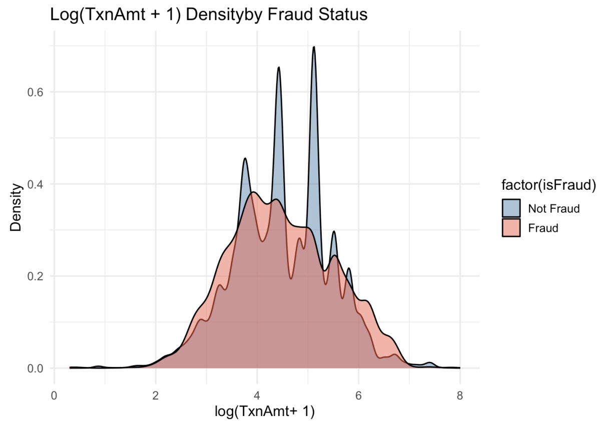


Figure 10: *Density of Transaction by Fraud Status*

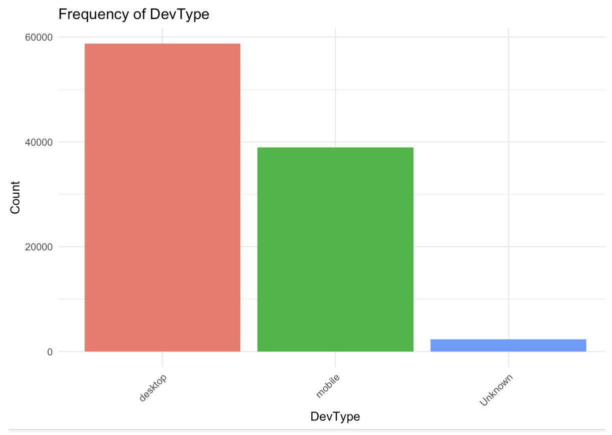


Figure 11: *Device Type Frequency*

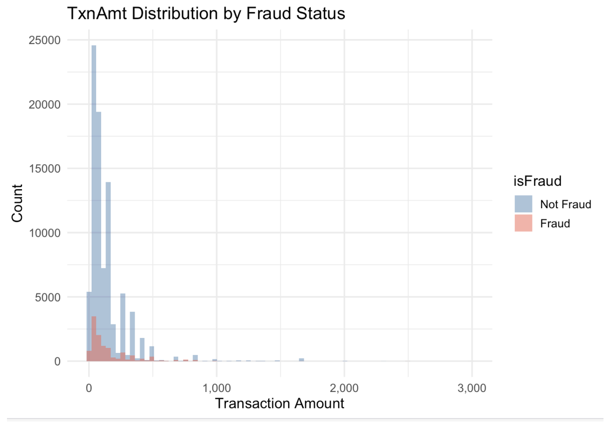


Figure 12: *TxnAmt Distribution by Fraud Status*

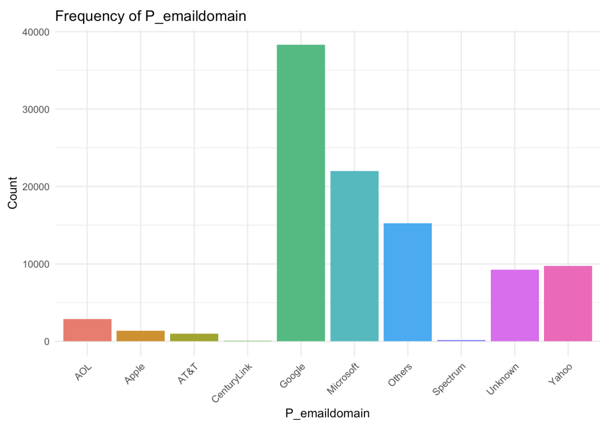


Figure 13: *Purchaser Email Domain Frequency*

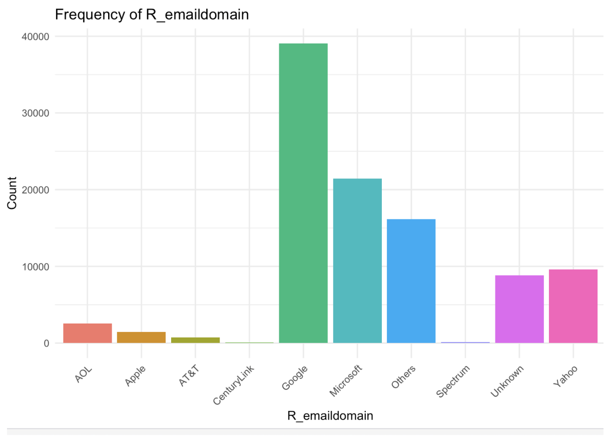


Figure 14: *Receiver Email Domain Frequency*

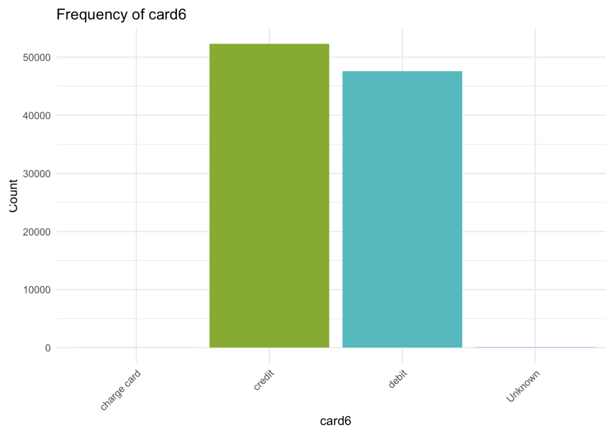


Figure 15: *Debit/Credit card Frequency*

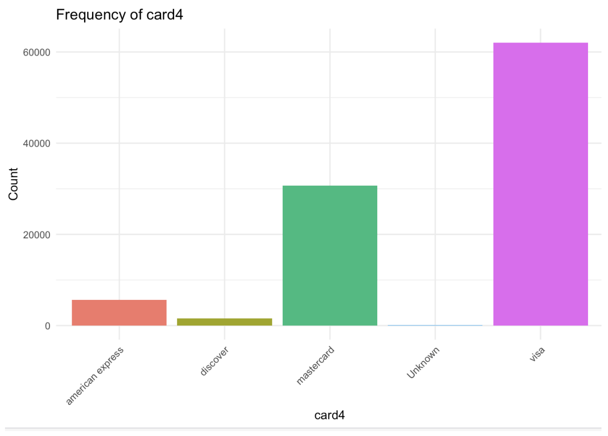


Figure 16: *Card Type Frequency*

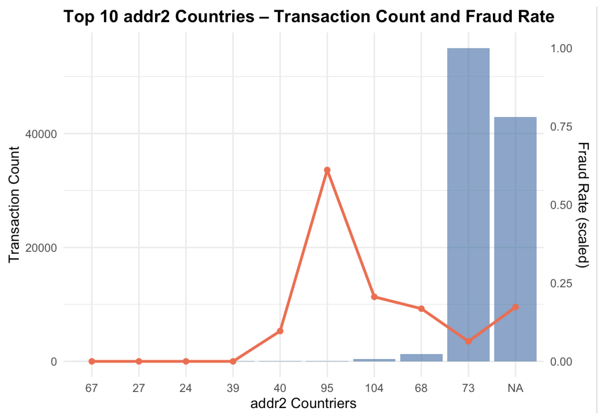


Figure 17: *Transactions and Fraud Rate by Country*

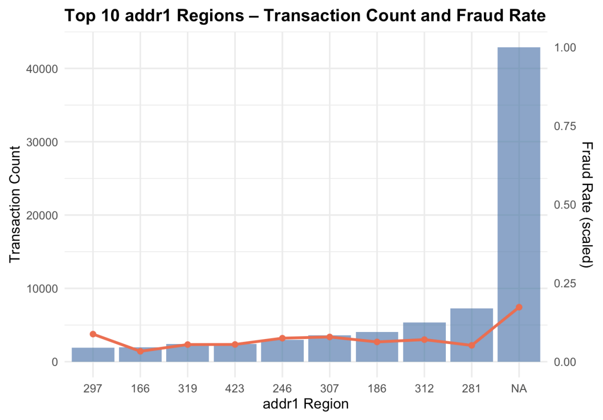


Figure 18: *Transactions and Fraud Rate by Region*

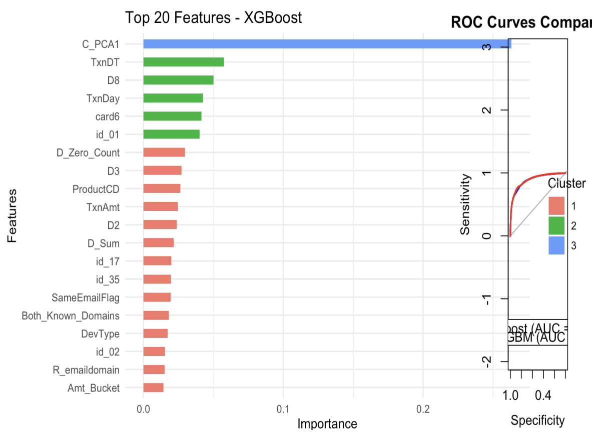


Figure 19: *Top 20 Features*

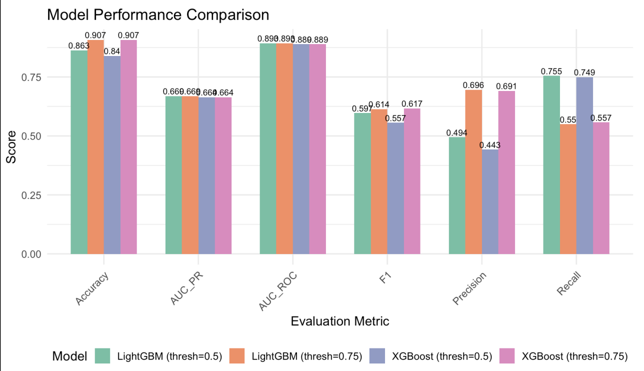


Figure 20: *Model Comparisons*