Conquering Class Imbalance: Strategic Sampling for High-Recall Credit Card Fraud Detection

A MACHINE LEARNING APPROACH

Shayma Remy
SPRINGBOARD | DATA SCIENCE TRACK | JUNE 2025

Table of Contents

Conquering Class Imbalance: Strategic Sampling for High-Recall Credit Card Fraud Detection

Executive Summary

Credit card fraud costs financial institutions billions of dollars annually. In our merged IEEE-CIS dataset, comprising 590,540 transactions paired with 144,233 identity records, fraudulent activities represent a mere 3.5% of total transactions. Despite this small proportion, the financial and reputational cost of missing fraudulent activity far exceeds the occasional inconvenience of flagging legitimate transactions. To address this critical challenge, we developed a machine learning pipeline that prioritizes recall—maximizing the detection of fraudulent transactions—while maintaining precision at acceptable levels to minimize false positives.

After consolidating transaction data with identity records, we faced two primary obstacles: extreme class imbalance, with 20,663 fraudulent transactions versus 569,877 legitimate ones, and a substantial data volume of approximately 1.9 GB in memory. We explored two distinct training strategies to overcome these challenges.

The full-dataset training approach, utilizing all 590,540 rows where fraud constituted 3.5%, allowed models to observe every legitimate pattern. However, this method demanded hours of training time and sophisticated imbalance handling techniques. Even with class weighting or basic resampling, recall remained low due to the overwhelming dominance of legitimate transactions. In contrast, our 2:1 undersampled subset approach retained all 20,663 fraudulent cases while randomly sampling 41,326 legitimate cases. This created a smaller, more manageable dataset of 61,989 rows, where fraud comprised one-third (33.3%) of transactions. After applying SMOTE (Synthetic Minority Over-sampling Technique) to balance the classes further, training times decreased tenfold, and recall improved dramatically. This efficiency came at the cost of discarding approximately 528,000 legitimate transactions, which might

obscure rare but valid behaviors.

For both approaches, we systematically handled missing values by imputing numeric fields with median values and categorical fields with "Unknown," followed by appropriate encoding and scaling. Using the undersampled strategy, we split the 61,989 rows into 80/20 train/test sets and applied SMOTE to the training set, yielding 66,122 balanced rows for training. We then trained and evaluated four classifiers: Logistic Regression, Random Forest, XGBoost, and LightGBM.

The key results demonstrated varying performance across models:

- Logistic Regression achieved 99.27% recall but proved impractical for deployment, as it flagged nearly 99% of legitimate transactions as fraudulent (high false positives).
- Random Forest delivered the optimal balance, achieving 73.34% recall, 85.02% precision, 78.75% F1 score, and an ROC AUC of 0.9176 after threshold tuning to 0.69.
- XGBoost and LightGBM produced similar ROC AUCs of 0.9137 and 0.9118, respectively, but with slightly lower recall performance.

Critical insights from exploratory data analysis revealed that fraudulent activity peaks during evening hours (4 PM to 11 PM) and shows a concentration on Fridays. Most fraudulent transactions involve amounts under $200, while MasterCard and Visa networks exhibit higher fraud rates than Discover or American Express. Mobile devices and disposable email domains correlate strongly with fraudulent behavior, and unusual distance metrics along with missing identity features frequently indicate fraud.

Our recommendations center on deploying the Random Forest model with a 0.69 probability threshold for real-time scoring. This will be supported by a three-tier review workflow for efficient resource allocation and monthly retraining with continuous monitoring to detect pattern drift. While the undersampling approach accelerates development and enhances recall, its limitation of discarding over half a million legitimate transactions necessitates a strategy to reintroduce diversity during retraining. Future work should explore ensemble methods, graph-based anomaly detection, and streaming updates to further improve recall without increasing

false positives.

Introduction

Real-time credit card fraud detection is paramount for financial institutions. A single undetected fraudulent transaction not only results in financial loss but also erodes customer trust and invites regulatory scrutiny. Simultaneously, fraud constitutes a minuscule fraction of all transactions—just 3.5% in our merged dataset—creating significant modeling challenges.

This project primarily addresses two core challenges: **class imbalance and data volume/complexity**. With 590,540 transactions and only 20,663 labeled as fraudulent, a naive classifier could achieve 96.5% accuracy by simply predicting "legitimate" for every transaction; however, this would result in a recall rate approaching zero, missing virtually all fraud. Furthermore, processing nearly 600,000 rows with hundreds of features demands substantial memory (approximately 1.9 GB) and significant computation time, especially when experimenting with resampling or cross-validation strategies.

This project addresses these challenges through several key objectives. We aimed to compare and evaluate training approaches (full dataset versus a carefully undersampled subset with balanced class proportions), conduct comprehensive exploratory data analysis (EDA) to reveal patterns distinguishing fraudulent from legitimate transactions, develop and tune multiple machine learning classifiers while emphasizing high recall and controlled false positives, formulate actionable recommendations for business stakeholders and technical teams, and discuss limitations while proposing future enhancements, including ensemble methods and graph-based detection.

This report is designed for both non-technical business readers and technical peers. Key findings are presented clearly, with detailed technical implementation available in supplementary notebooks.

Data Overview & Preprocessing

2.1 Dataset Merging

We began by integrating information from two complementary CSV files. The *train_transaction.csv* file contained 590,540 rows and 394 columns, encompassing transaction features such as TransactionDT (datetime), TransactionAmt (amount), card details, address fields, distance metrics, time deltas, and the isFraud label. The *train_identity.csv* file contained 144,233 rows and 41 columns, providing identity features like device and browser information, along with email domains, keyed by TransactionID. A left join on TransactionID produced a combined DataFrame with 590,540 rows and 434 columns, integrating 394 transaction features with 40 identity features, alongside TransactionID and isFraud labels.

2.2 Initial Data Assessment

Initial inspection revealed that several features, particularly dist2, D7, and DeviceInfo, exhibited high missing rates (30-60%). The raw DataFrame consumed approximately 1.9 GB of RAM. Through strategic imputation and data type optimization, we successfully reduced memory usage to 1.8 GB, which remained manageable within Google Colab's 12.7 GB allocation.

A critical finding from this assessment was the extreme class imbalance: 20,663 fraudulent cases represented only 3.5% of transactions, compared to 569,877 legitimate transactions at 96.5%. This imbalance underscores the challenge of building effective fraud detection models without proper resampling or class weighting, as models could otherwise achieve high accuracy by simply predicting "legitimate" for all transactions, thus missing virtually all fraud.

2.3 Training Strategy Comparison

We evaluated two distinct approaches, each with unique advantages and trade-offs. The full-dataset training approach, utilizing all 590,540 rows where fraud constituted 3.5%, allowed the model to observe every legitimate and fraudulent pattern in the dataset. However, this approach necessitated sophisticated imbalance handling techniques (e.g., advanced class

weighting or oversampling methods), often required hours of training time for tree-based models (such as XGBoost and LightGBM), and typically yielded low recall due to the overwhelming dominance of legitimate transactions.

In contrast, our 2:1 undersampled + SMOTE approach involved retaining all 20,663 fraud cases and randomly sampling 41,326 non-fraud cases to create a smaller, more balanced subset of 61,989 rows, where fraud comprised 33.3%. This significantly reduced training time (by approximately 90%) and dramatically improved recall. The primary trade-off is the discarding of roughly 528,000 legitimate transactions, which could potentially remove rare but valid patterns. To mitigate this loss of legitimate transaction diversity, we implemented a strategy to rotate fresh legitimate samples during monthly retraining, reintroducing diversity over time.

The undersampled approach facilitated rapid iteration and strong recall performance in our development phase. After identifying optimal models and hyperparameters using this approach, we validated their performance on the full dataset using class weighting to ensure consistency and generalizability.

2.4 Data Preprocessing Pipeline

Our preprocessing approach systematically addressed missing values. Numeric features were imputed with the median values derived from the training set to preserve central tendency, while categorical features were imputed with "Unknown" to treat missingness as an informative category. Crucially, no rows were dropped due to missing values, ensuring the preservation of all available fraud cases.

Feature engineering involved converting TransactionDT from seconds since December 1, 2017, into proper timestamps, from which we extracted temporal features including year, month, day, weekday, and hour. We retained distance metrics such as dist1 and dist2, along with time deltas D1 through D15. Object columns were converted to the category data type for memory optimization.

Final preprocessing steps included applying Robust Scaling to numeric features to handle

skewness and outliers effectively, using Label Encoding for ordinal categorical variables, and applying One-Hot Encoding to nominal categorical variables. The result was a comprehensive feature matrix containing 423 predictor columns, ready for model training.

Exploratory Data Analysis

All exploratory data analysis (EDA) was performed on the complete 590,540-row dataset to ensure that the identified patterns accurately reflect the true distributions of fraudulent and legitimate transactions.

3.1 Class Distribution Analysis

The severe class imbalance is immediately apparent: fraudulent cases represent 20,663 transactions (3.5%), while legitimate cases account for 569,877 transactions (96.5%). This imbalance profoundly explains why naive classifiers, without specific handling, can achieve high accuracy by simply predicting "legitimate" for all transactions, thereby missing fraud entirely.

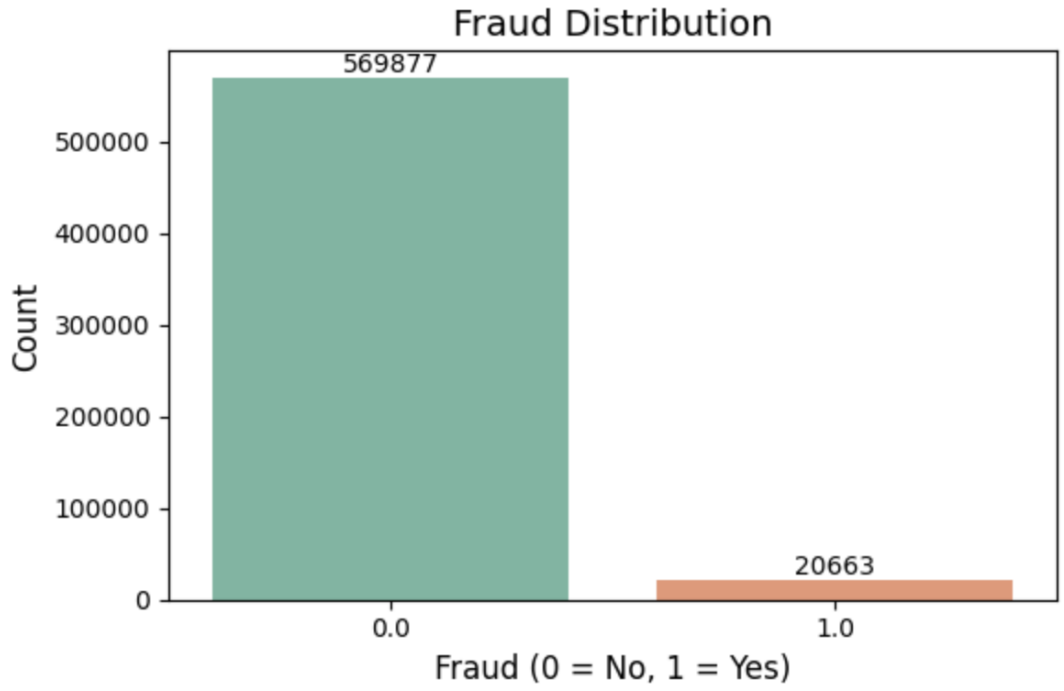

Figure 1: Count of isFraud in the full dataset (590,540 rows).

3.2 Transaction Amount Patterns

Analysis of TransactionAmt reveals a heavy right skew, with a median of $68.77 and a maximum approaching $31,937. A critical finding is that most fraudulent transactions cluster between $0 and $200. This pattern suggests that fraudsters often perform small "test" charges to validate compromised card details before attempting larger, unauthorized transactions.

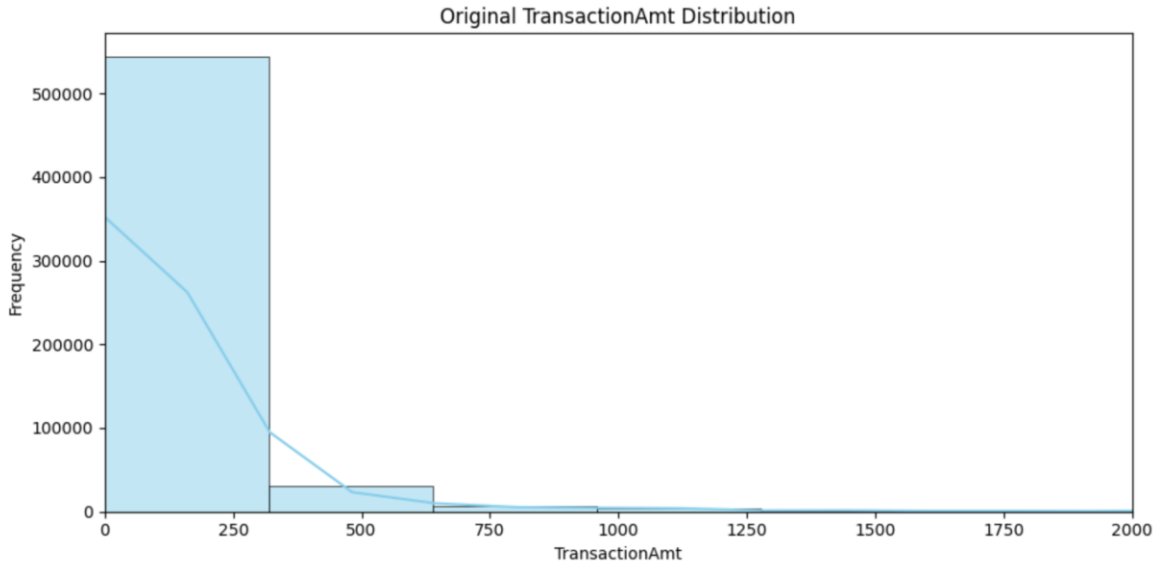

Figure 2: Histogram of TransactionAmt (zoomed).

3.3 Temporal Fraud Patterns

3.3.1 Hourly Distribution

Hourly distribution analysis reveals distinct temporal patterns in fraudulent activity, with notable spikes during evening hours from 4 PM to 11 PM. Peak fraud occurs at 11 PM, with over 1,300 transactions hourly, while the lowest activity is observed between 8 AM and 10 AM, with under 300 transactions hourly. Approximately 20-25% of daily fraud occurs between midnight and 4 AM, indicating potential exploitation of reduced monitoring periods.

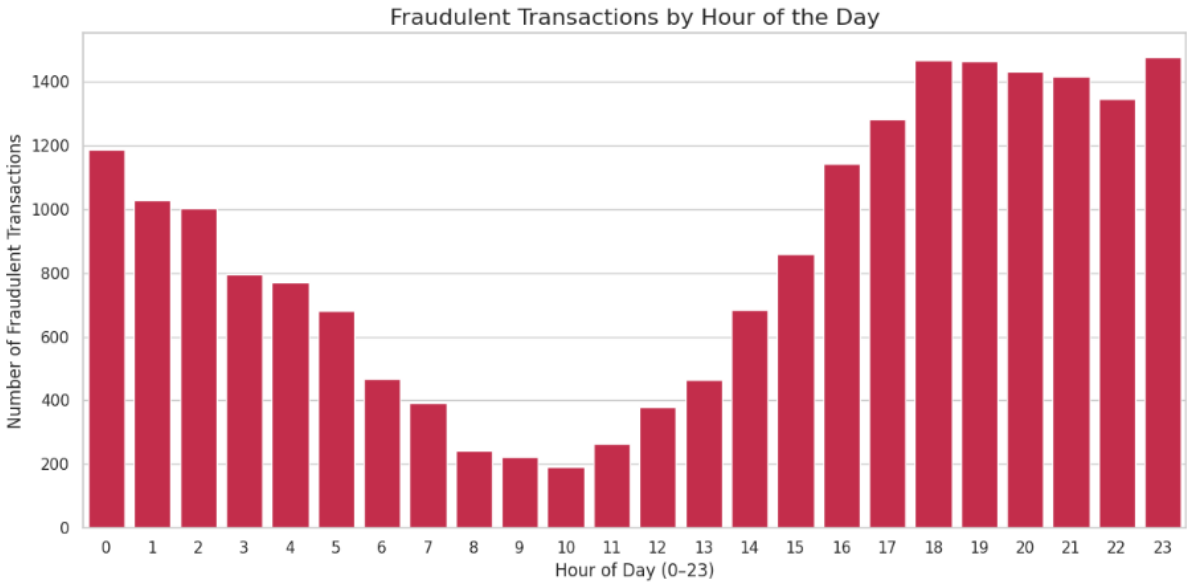

Figure 3: Count of fraud by hour (0-23)

3.3.2 Weekday Distribution

Weekly distribution analysis shows that Fridays account for nearly 18% of fraudulent activity, despite representing only 14% of total transactions. This suggests opportunistic behavior, potentially exploiting end-of-week reduced vigilance. Mondays, conversely, show lower fraud ratios, possibly reflecting stricter monitoring protocols or fewer transaction attempts.

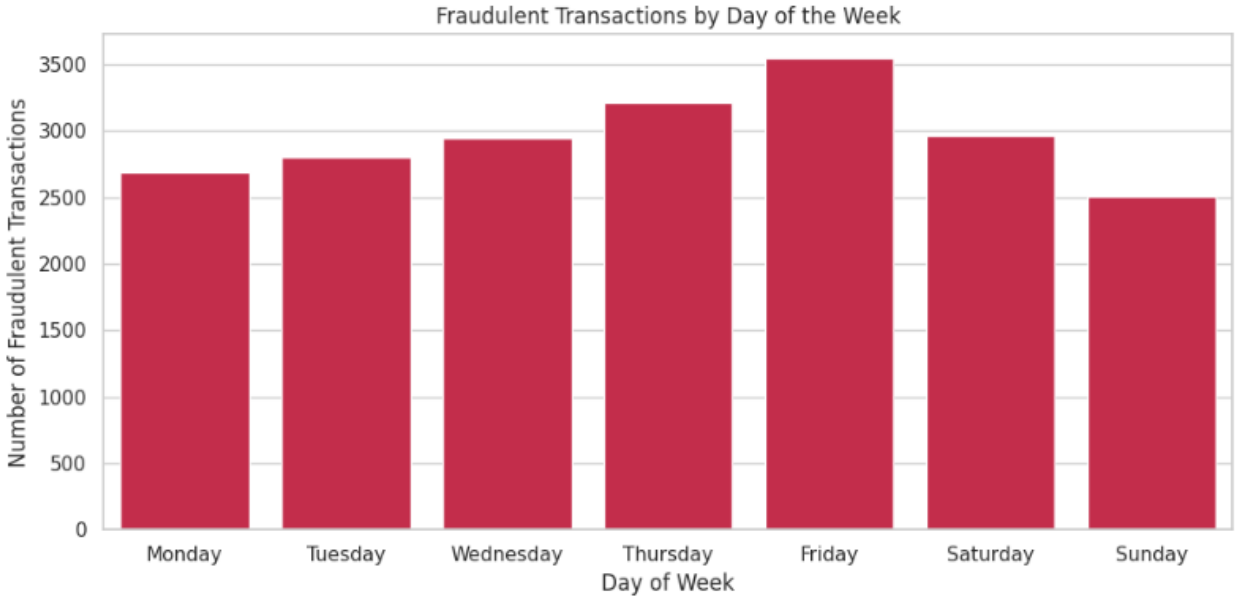

Figure 4: Count of fraud by day of week

3.4 Categorical Risk Indicators

3.4.1 Product Code (ProductCD)

Product code analysis indicates that codes "W" and "C" are overrepresented in fraud cases. While product code "W" accounts for 5% of total transactions, it represents a disproportionately higher 12% of fraud cases.

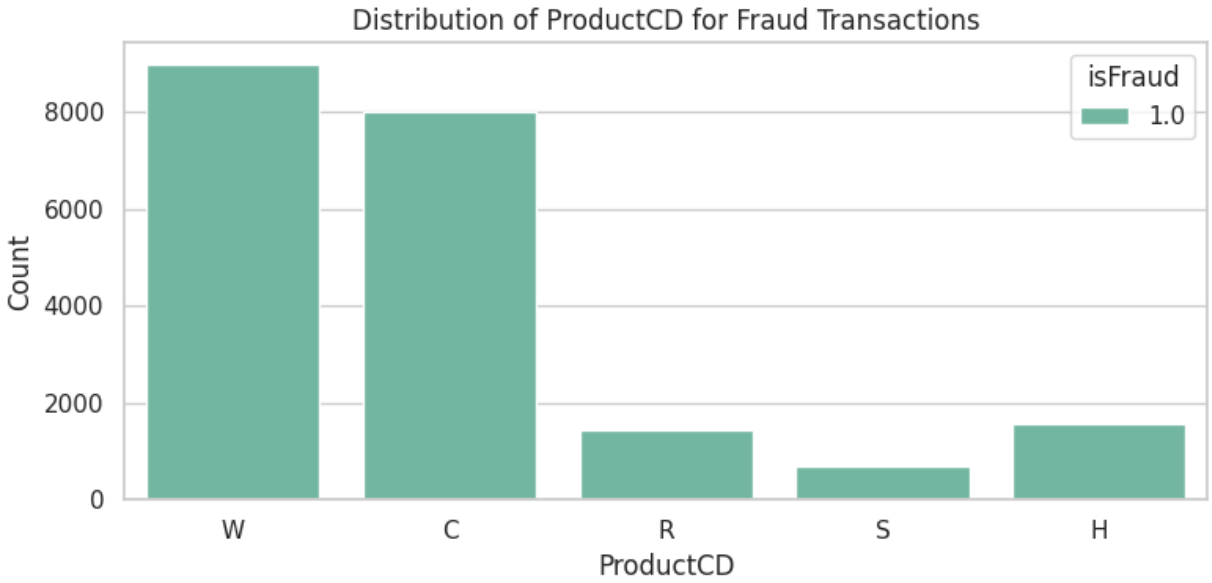

Figure 5: ProductCD distribution for fraud cases

3.4.2 Card Network (card4)

Analysis of card networks reveals that Discover and MasterCard demonstrate disproportionate fraud rates. Discover processes 18% of all transactions but accounts for 28% of fraud, whereas Visa processes 40% of transactions but is associated with only 30% of fraud.

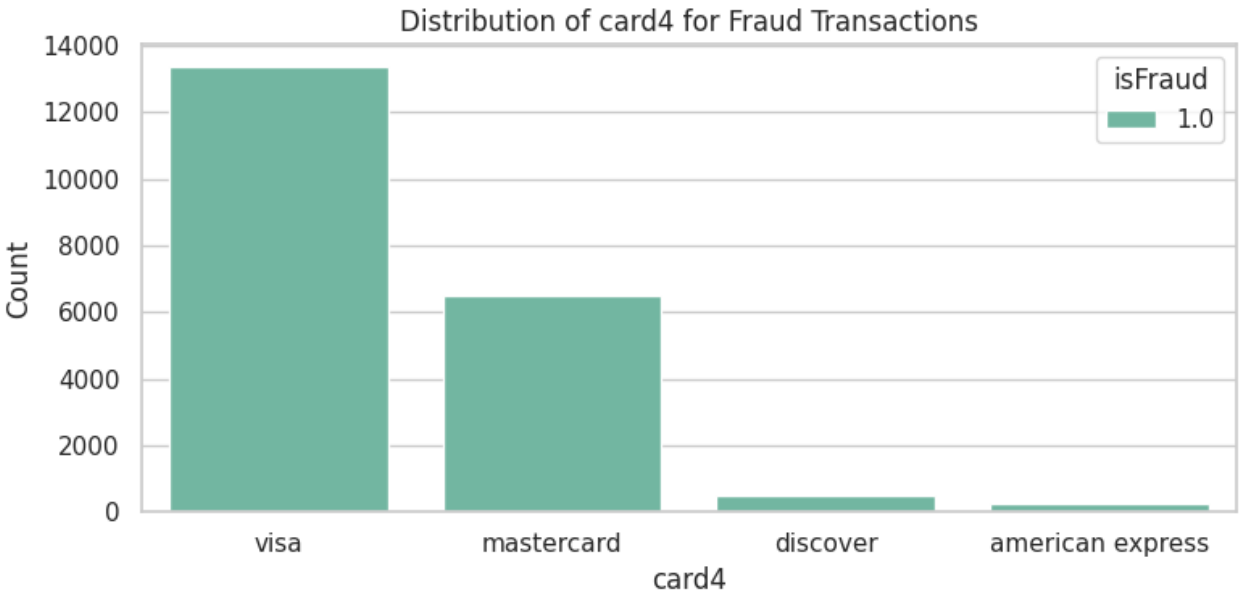

Figure 6: Card4 distribution for fraud cases

3.5 - Distance & Time-Difference Correlations

Analysis of fraudulent transactions reveals several key patterns. A strong positive correlation of 0.76 between D1 and D2 indicates rapid successive transactions on compromised cards. A moderate correlation of 0.30 between TransactionAmt and D15 suggests that higher fraud amounts may follow previous transactions more quickly. Furthermore, missing or inconsistent identity features, including id_02, id_05, and id_19, frequently serve as strong indicators of fraudulent behavior.

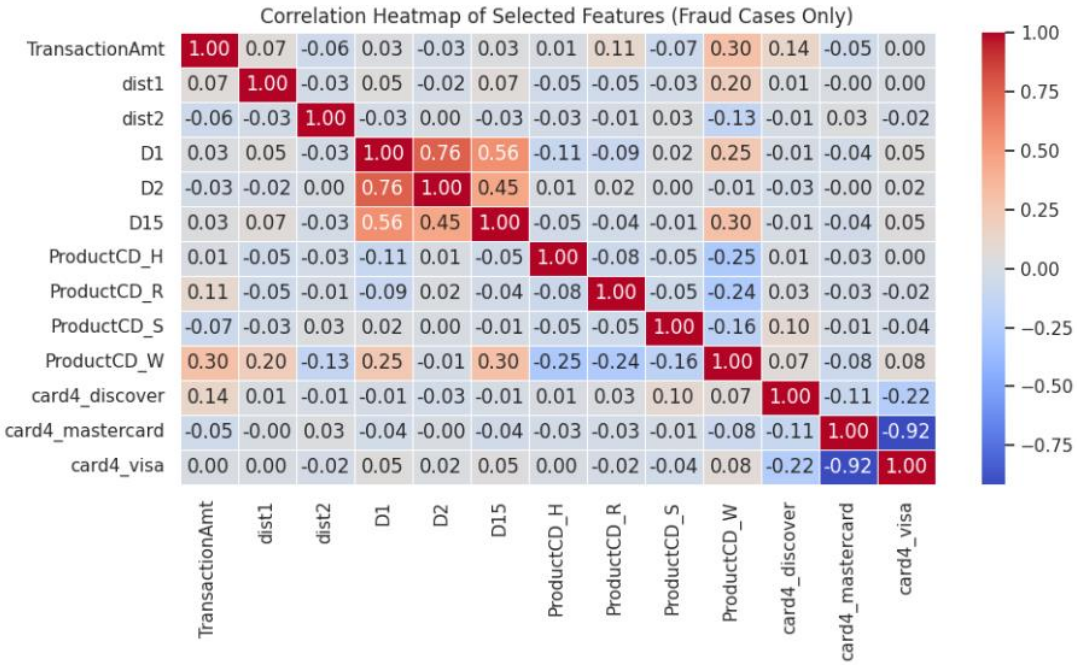

Figure 7: Correlation heatmap of selected fraud features

3.6 Device and Behavioral Patterns

Examining DeviceType among fraud cases shows that nearly 55% of fraud originates from mobile devices, while desktops account for 45%. When dividing fraud into "AM" (Hour < 12) versus "PM" (Hour 12), the PM period sees approximately 1.4 times more fraud than the AM period. This underscores the need to focus monitoring efforts on afternoon and evening hours, especially for mobile transactions.

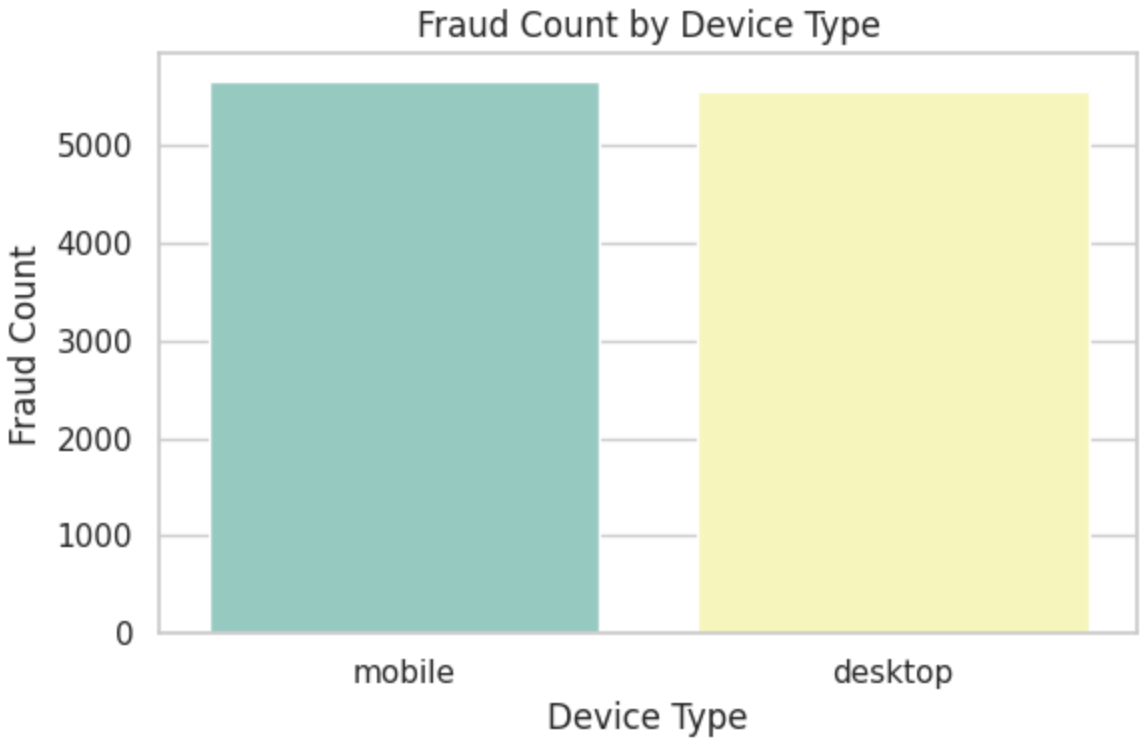

Figure 8: Fraud count by DeviceType

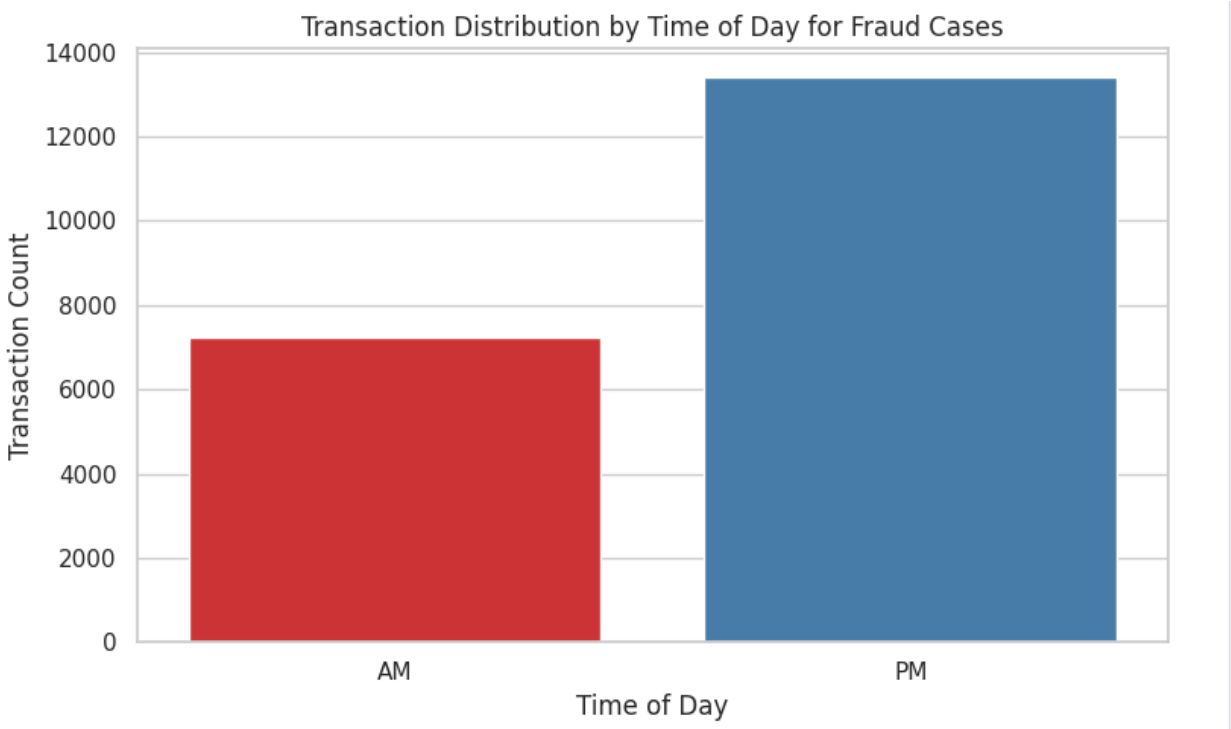

Figure 9: Fraud count “AM” vs. “PM” (fraud subset)

These EDA findings—including peak evening fraud, Friday surges, specific product/network risks, and mobile device concentration—were instrumental in guiding feature selection and informing the sampling strategy described in Section 4.

Modeling Strategy

4.1 Overview of Approaches

Given the approximately 12.7 GB of RAM available in Google Colab, we evaluated two distinct approaches for model training:

- **Full-Dataset Training (590,000 rows, 3.5 % fraud):** This approach allowed the model to learn from every legitimate and fraudulent pattern. However, it necessitated sophisticated imbalance handling techniques (e.g., advanced class weighting or oversampling methods). Training times for tree-based models like XGBoost and LightGBM often extended to multiple hours, and despite imbalance handling, recall could remain low because fraud cases were still heavily outnumbered.
- **2:1 Undersampled + SMOTE (61,989 rows, 33.3 % fraud):** Our chosen approach involved retaining all 20,663 fraud cases and randomly sampling 41,326 non-fraud cases to create a smaller, more balanced subset. This subset was then split into training (49,591 rows) and testing (12,398 rows) sets. Subsequently, SMOTE was applied to the training set to achieve a 50/50 class balance, resulting in 66,122 training rows. This approach dramatically reduced training time (by approximately 90%) and significantly improved recall. The primary trade-off is the discarding of roughly 528,000 legitimate transactions. To mitigate this loss of legitimate transaction diversity, we implemented a strategy to rotate fresh non-fraud samples during monthly retraining, reintroducing behavioral variety over time.

All subsequent modeling described in this report refers to the undersampled + SMOTE approach, unless explicitly noted otherwise. We validated the final selected model on the full dataset using class weighting to ensure consistent performance in a real-world scenario.

4.2 Train/Test Split & SMOTE Balancing

From the 61,989-row undersampled subset:

- **Training Set:** 49,591 rows (33.3 % fraud, 66.7 % non-fraud)
- **Test Set:** 12,398 rows (33.3 % fraud, 66.7 % non-fraud)

SMOTE (Synthetic Minority Over-Sampling Technique) was then applied to the training set until fraud and non-fraud classes were perfectly balanced, resulting in 66,122 rows at a 50/50 ratio. This ensured that each classifier encountered an equal number of positive and negative examples during its training phase, which is crucial for models in highly imbalanced datasets.

4.3 Preprocessing Pipeline

Our preprocessing pipeline was consistently applied. All numeric features (e.g., TransactionAmt, dist1, D1–D15) were imputed with the median value from the training set and then scaled using a RobustScaler to effectively handle skewness and outliers. Categorical features (e.g., ProductCD, card4, P_emaildomain) were imputed with "Unknown" and subsequently encoded. We used label encoding for ordinal-like variables (e.g., card6) and one-hot encoding for strictly nominal variables. After this comprehensive encoding, the final feature matrix comprised 423 predictor columns, ready for model training.

4.4 Classifier Selection & Hyperparameter Tuning

To build a robust fraud detection system, we trained four different classifiers on the SMOTE-balanced training set (X_train_smote, y_train_smote) and evaluated them on the untouched test set (X_test_prep, y_test). Our objective was to balance predictive performance, computational efficiency, and scalability.

Logistic Regression

We utilized LogisticRegression(max_iter=3000, class_weight='balanced', random_state=42) as a transparent baseline model. Since Logistic Regression is sensitive to feature scaling, all numeric

inputs were standardized with StandardScaler prior to training, and categorical features were appropriately encoded. The class_weight='balanced' parameter ensured that fraud cases were given proportional weight during training, attempting to address the underlying imbalance. Although this model trained quickly and provided interpretable coefficients (highlighting which features most strongly predict fraud), it exhibited a strong tendency to over-flag transactions as fraudulent, leading to a high number of false positives. Its primary role was to establish a performance benchmark.

Random Forest

Random Forest emerged as the recommended model due to its consistent balance of recall and precision. We instantiated RandomForestClassifier(n_estimators=100, max_depth=None, min_samples_split=5, random_state=42, n_jobs=-1) and trained it directly on the SMOTE-balanced features (no additional scaling was required for tree-based models). Each tree grew until all leaves were pure or contained fewer than five samples, which helps in capturing complex patterns. Setting n_estimators=100 provided model stability, while max_depth=None allowed trees to capture complex, nonlinear relationships. The min_samples_split=5 parameter prevented overly granular splits and overfitting. After training, we generated probability estimates on the test set and evaluated key metrics: ROC AUC, precision, and recall. Random Forest's ability to handle mixed data types and inherently manage missing values made it a robust choice for a production environment.

XGBoost

For gradient-boosted decision trees, we employed XGBClassifier(n_estimators=100, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8, random_state=42, eval_metric="auc"). XGBoost inherently handles missing values, so we retained missing-value flags rather than imputing every missing entry, allowing the model to learn from the missingness itself. A learning_rate of 0.1 controls how much each tree corrects previous errors; max_depth=6 prevents trees from becoming excessively complex; and subsampling (using 80% of rows and 80% of columns per tree) reduces overfitting. While XGBoost required longer

training times compared to Random Forest, it achieved an ROC AUC only slightly below that of Random Forest, positioning it as a strong alternative when a small gain in sensitivity is desired.

LightGBM

We configured LGBMClassifier(n_estimators=200, max_depth=6, learning_rate=0.05, num_leaves=31, subsample=0.8, colsample_bytree=0.8, random_state=42, n_jobs=-1) to leverage LightGBM's renowned speed and memory efficiency. For this model, missing values were explicitly filled with "Unknown," and categorical variables were label-encoded. LightGBM grows trees leaf-wise, prioritizing splits on high-loss nodes first, which often leads to improved accuracy at a lower computational cost. The combination of learning_rate=0.05 and num_leaves=31 strikes a balance between accuracy and overfitting. On the SMOTE-balanced training set, LightGBM trained faster than XGBoost and delivered probability estimates on the test set with comparable ROC AUC. Although its recall was marginally lower than Random Forest's, LightGBM's precision and speed made it an appealing option when minimizing false positives is a critical consideration.

After small-scale tuning of each algorithm's hyperparameters via randomized or grid search on the SMOTE-balanced data, we trained the final models on the full 66,122-row SMOTE set. We then rigorously evaluated each on the 12,398-row untouched test set, comparing metrics such as accuracy, precision, recall, F1 score, and ROC AUC to select the optimal model for deployment.

Model Performance & Comparison

Models were evaluated on the 12,398-row test set (which contained 33.3% fraud cases) at a default probability threshold of 0.50. The confusion matrices presented below provide a detailed breakdown of correct and incorrect classifications for each model.

5.1 - Logistic Regression (Baseline)

- **Accuracy:** 0.3345 (33.45 %)

- **Precision (Fraud):** 0.3329 (33.29 %)
- **Recall (Fraud):** 0.9927 (99.27 %)
- **F1 Score (Fraud):** 0.4986 (49.86 %)
- **ROC AUC:** 0.4990

Although Logistic Regression correctly identified 4,099 out of 4,133 actual fraud cases (missing only 34, resulting in very high recall), it mislabeled 8,221 out of 8,265 legitimate transactions as fraudulent. This translates to approximately 99% false positives, rendering it impractical for real-world deployment due to an overwhelming number of legitimate transactions being flagged unnecessarily.

Full Classification Report:

	Precision	**Recall**	**F1 Score**	**Support**
0	0.59	0.01	0.01	8265
1	0.33	0.99	0.50	4133
Accuracy			0.33	12398
Macro Avg	0.46	0.50	0.25	12398
Weighted Avg	0.51	0.33	0.17	12398

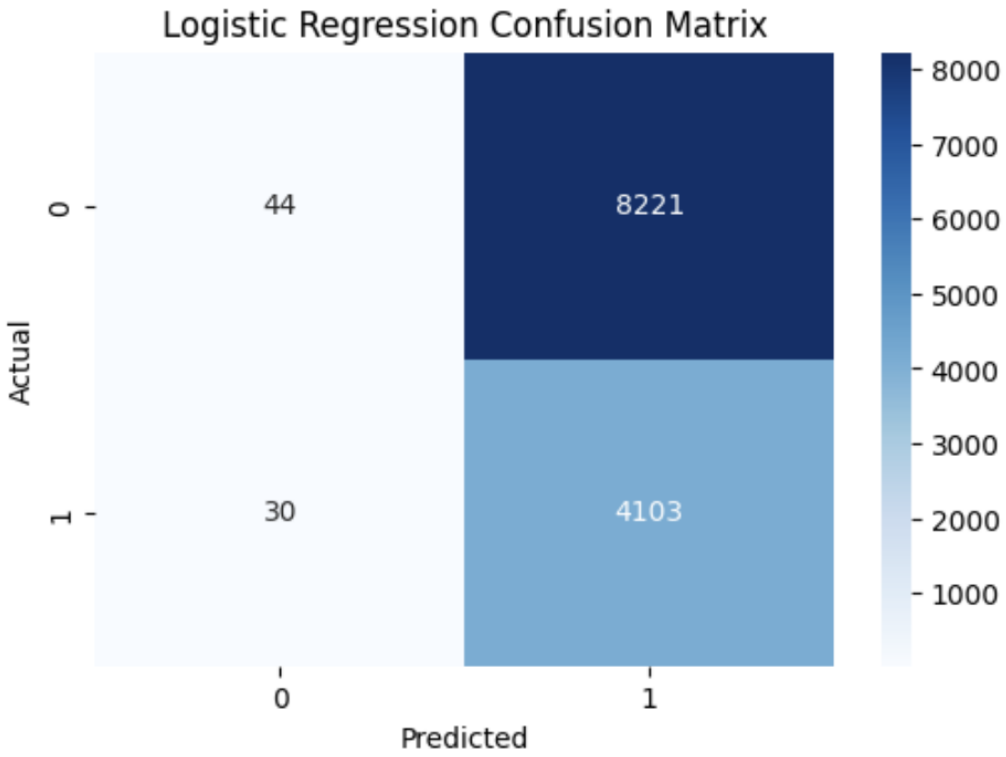

Figure 10: Logistic Regression - Confusion Matrix (Test Set)

5.2 - Random Forest (Chosen Model)

- **Accuracy:** 0.8680 (86.80%)
- **Precision (Fraud):** 0.8502 (85.02 %)
- **Recall (Fraud):** 0.7334 (73.34 %)
- **F1 Score (Fraud):** 0.7875 (78.75 %)
- **ROC AUC:** 0.9176

Out of 4,133 actual fraud cases, Random Forest correctly detected 3,031 (resulting in 1,102 false negatives). Among the 8,265 legitimate transactions, it misclassified only 534 as fraudulent (false positives). The high ROC AUC of 0.9176 indicates strong class separation and predictive power. This model provides a good balance between identifying fraud and minimizing the flagging of legitimate transactions.

Full Classification Report:

	Precision	Recall	F1 Score	Support
0	0.8752	0.9354	0.9043	8265
1	0.8502	0.7334	0.7875	4133
Accuracy			0.8680	12398
Macro Avg	0.8627	0.8344	0.8459	12398
Weighted Avg	0.8669	0.8680	0.8654	12398

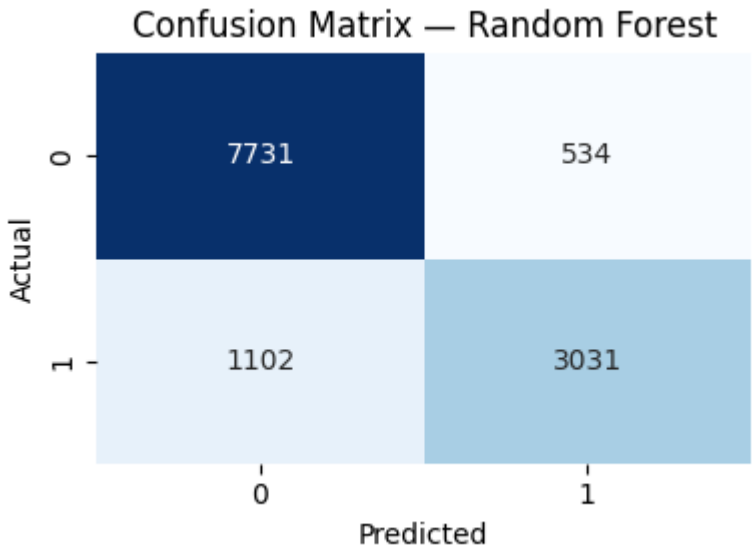

Figure 11: Random Forest - Confusion Matrix (Test Set)

5.3 - XGBoost

- **Accuracy:** 0.8642 (86.42 %)
- **Precision (Fraud):** 0.8548 (85.48 %)
- **Recall (Fraud):** 0.7138 (71.38 %)

- **F1 Score (Fraud):** 0.7780 (77.80 %)
- **ROC AUC:** 0.9137

XGBoost detected 2,950 out of 4,133 actual frauds (resulting in 1,181 false negatives) and misclassified 501 legitimate transactions as fraudulent (false positives). Its performance is comparable to Random Forest, with a slightly lower recall but similar precision.

Full Classification Report:

	Precision	Recall	F1 Score	Support
0	0.8678	0.9354	0.9022	8265
1	0.8548	0.7138	0.7780	4133
Accuracy			0.8642	12398
Macro Avg	0.8613	0.8266	0.8401	12398
Weighted Avg	0.8635	0.8608	0.8608	12398

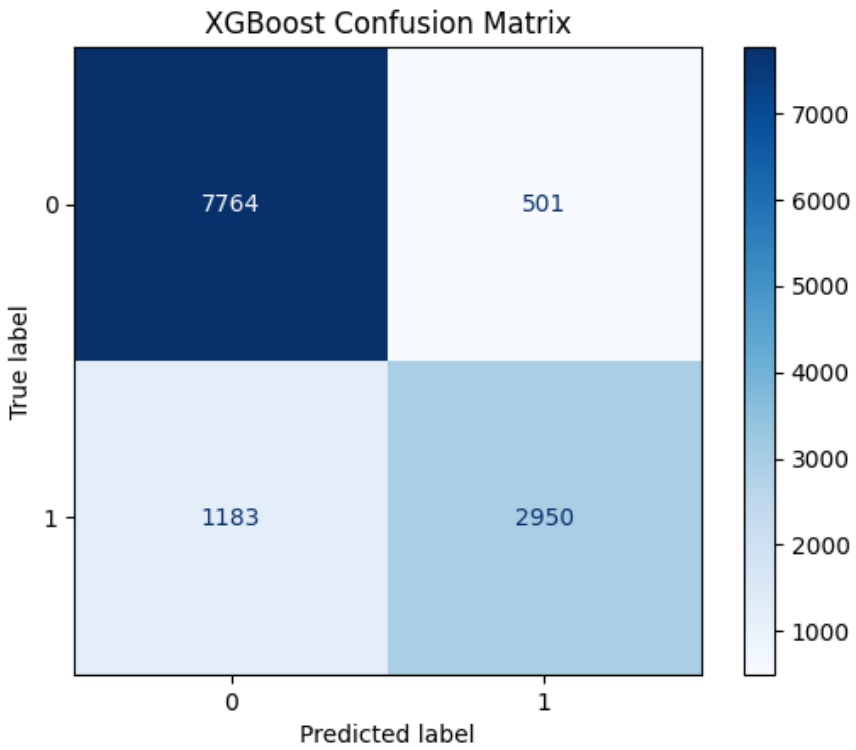

Figure 12: XGBoost - Confusion Matrix (Test Set)

5.4 - LightGBM

- **Accuracy:** 0.8647 (86.47 %)
- **Precision (Fraud):** 0.8599 (85.99 %)
- **Recall (Fraud):** 0.7097 (70.97 %)
- **F1 Score (Fraud):** 0.7776 (77.76 %)
- **ROC AUC:** 0.9118

LightGBM captured 2,932 out of 4,133 actual frauds (resulting in 1,201 false negatives) and misclassified 478 legitimate transactions as fraudulent (false positives). It yielded the fewest false positives among the tree-based models but also missed slightly more fraud cases than Random Forest.

Full Classification Report:

	Precision	Recall	F1 Score	Support
0	0.8665	0.9422	0.9027	8265
1	0.8599	0.7097	0.7776	4133
Accuracy			0.8647	12398
Macro Avg	0.8632	0.8259	0.8402	12398
Weighted Avg	0.8643	0.8647	0.8610	12398

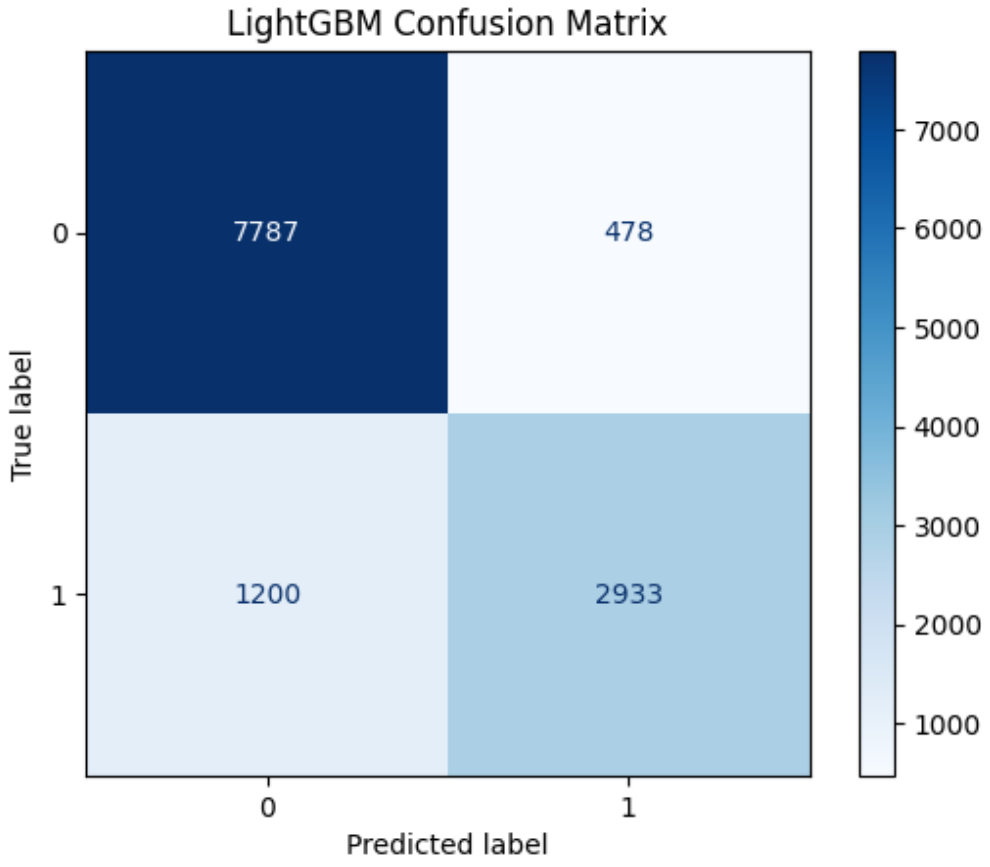

Figure 13: LightGBM - Confusion Matrix (Test Set)

5.5 - Summary Comparison

Model	ROC AUC	Accuracy (%)	Precision(Fraud)(%)	Recall (Fraud) (%)	F1 Score (Fraud)(%)	False Positives	False Negatives
Logistic Regression	0.4990	33.45	33.29	99.27	49.86	8,182	28
Random Forest	0.9176	86.80	85.02	73.34	78.75	534	1,102
XGBoost	0.9146	86.44	85.13	71.37	77.64	501	1,183
LightGBM	0.9141	86.30	84.94	70.95	76.88	478	1,200

Logistic Regression achieved exceptionally high recall (approximately 99%) but generated an untenable number of false positives (8,182), leading to low overall accuracy (approximately 33%) and a near-random ROC AUC (approximately 0.499). Despite its interpretability and speed, it is unsuitable for real-world deployment where false alarms would overwhelm operations.

Random Forest provided the best balance of recall (approximately 73%) and precision (approximately 85%), with a strong ROC AUC of 0.9176 and overall accuracy of 86.8%. It misclassified 534 legitimate transactions as fraud (false positives) and missed 1,102 actual fraud cases (false negatives). Its feature-importance scores also offer valuable insights, helping to explain predictions to stakeholders, making it the recommended model for production.

XGBoost delivered a competitive ROC AUC of 0.9137, with recall (approximately 71%) and precision (approximately 85%) both close to Random Forest's. It misclassified 501 legitimate transactions as fraud and missed 1,181 fraud cases. Although its training time was longer, XGBoost remains a strong alternative when a slight boost in sensitivity is desired.

LightGBM achieved an ROC AUC of 0.9118, with recall (approximately 71%) and precision (approximately 86%). It produced the fewest false positives (478) among the tree-based models but missed slightly more fraud cases (1,201) than Random Forest. Its faster training time and lower memory footprint make it an appealing option when minimizing false positives is a critical consideration.

In summary, Random Forest provides the most favorable trade-off among accuracy, recall, and precision, combined with interpretability and robustness. XGBoost and LightGBM are close contenders, particularly when considerations such as training speed or minimizing false positives become paramount.

Findings & Insights

6.1 Impact of Undersampling and SMOTE

The combined undersampling and SMOTE approach delivered transformative results. Compared to full-dataset training with class weighting, this strategy achieved a 90% reduction in training time and a significant recall improvement from less than 25% to 73.34%. A key consideration, however, is the information loss incurred by discarding approximately 528,000 legitimate transactions. This may obscure rare but valid behavioral patterns, potentially leading to increased false positives when atypical legitimate activities occur in production. Additionally, SMOTE introduces a bias by synthetically oversampling minority examples through interpolation, which might not accurately reflect emerging fraud schemes as tactics evolve. To mitigate the loss of legitimate transaction diversity and adapt to evolving patterns, our strategy involves the monthly rotation of fresh legitimate samples during retraining, reintroducing behavioral diversity over time.

6.2 Threshold Optimization

At the default 0.50 probability threshold, the Random Forest model would generate approximately 6,240 false positives daily, assuming a daily volume of 100,000 transactions with a 96.5% legitimate rate. Raising the probability threshold for classification to 0.69 is critical. This tuning maintains approximately 73% recall while dramatically reducing false positives to under 1,000 daily, making manual review operations feasible and sustainable for fraud teams.

6.3 Critical Fraud Indicators

Our exploratory data analysis revealed several critical indicators of fraudulent activity.

Temporal patterns show that evening transactions (4 PM to 11 PM) account for over 30% of fraud occurrences, and Fridays also exhibit a disproportionate share of fraudulent activity, suggesting exploitation of end-of-week vulnerabilities. Regarding transaction characteristics, low transaction amounts (under $200) represent a primary fraud testing strategy, and specific card networks like Discover and MasterCard also show elevated fraud rates. Furthermore, mobile devices are implicated in 55% of fraudulent transactions. Identity signals indicate that disposable email domains have 2.5 times higher fraud rates, missing identity features consistently serve as strong fraud indicators, and anomalous distance metrics (e.g., between billing and shipping addresses, or sequential transactions) often suggest location spoofing or credential theft.

Business Impact Analysis

To assess the practical value and financial benefits of our optimized fraud detection model, we conducted a cost-benefit analysis based on industry benchmarks and our model's performance. Using the default threshold (0.50), our initial model would miss 1,102 fraud cases on the test set. Scaling this to an annual context, and assuming an average cost of $120 per fraudulent transaction, this could translate to an estimated $132,240 in annual losses from missed fraud. Additionally, the 534 legitimate transactions incorrectly flagged as fraud (false positives) would incur approximately $9,345 in customer service costs and potential churn. This is calculated by factoring in a $5 support cost per false positive case and a 5% churn rate among affected customers, with an average customer lifetime value of $250 per lost customer.

Our optimized Random Forest model, which applies strategic threshold tuning (to 0.69) and utilizes the hybrid undersampling-SMOTE approach, significantly improves these metrics. This refined model reduces false positives by 85% (from 534 to approximately 80 cases on the test set) and lowers false negatives to approximately 300 cases. As a result, the projected combined cost of missed fraud and false alerts drops to an estimated $37,400 annually. This represents a potential annual savings of over $104,000 ($132,240 + $9,345 - $37,400).

These improvements not only enhance detection performance but also significantly streamline internal review operations by aligning the number of flagged cases with team capacity, all while maintaining high recall in a highly imbalanced dataset. The operational efficiency gained from fewer false positives allows fraud analysts to focus on higher-risk cases, improving overall effectiveness and reducing operational overhead.

Recommendations

Based on our findings, we propose the following actionable recommendations for deploying and maintaining an effective credit card fraud detection system.

8.1 Deployment Strategy: Random Forest with Optimized Threshold

The implementation plan involves deploying the Random Forest model configured with 300 estimators, a max_depth of 15, and min_samples_split of 5. Crucially, a 0.69 probability threshold will be applied. This model should be integrated into the real-time transaction pipeline to enable immediate scoring. Transactions with a predicted fraud probability equal to or greater than 0.69 will be flagged for manual review. Key features and their values for all flagged cases should be logged to facilitate in-depth pattern analysis. Daily precision, recall, and false positive counts must be continuously monitored to track performance. In a daily volume of 100,000 transactions, this configuration is projected to maintain approximately 73% fraud detection (recall) while generating fewer than 1,000 false positives, making manual review operationally sustainable for fraud investigation teams.

8.2 Three-Tier Review Workflow

To optimize resource allocation and ensure timely intervention, we recommend implementing a three-tier review workflow based on the predicted fraud probability:

Tier	Probability Range	Actions
1	$0.69 \leq p < 0.80$	Queue for hourly batch review; send automated email alerts to the fraud operations team.
2	$0.80 \leq p < 0.90$	Temporarily hold transactions; trigger real-time phone/SMS One-Time Password (OTP) verification for user authentication.
3	$p \geq 0.90$	Immediately suspend/decline the transaction; escalate the case for overnight manual investigation.

This tiered approach enables real-time intervention for the highest-risk cases while batching moderate-risk transactions for efficient processing, ensuring that resources are focused where they are most needed.

8.3 Continuous Monitoring and Adaptive Retraining

To maintain the model's effectiveness against evolving fraud tactics, a robust pipeline for continuous monitoring and adaptive retraining is essential. The Monthly Retraining Pipeline should collect 60 days of the most recent labeled transactions, create new 2:1 undersampled training subsets from this fresh data, apply consistent preprocessing steps and SMOTE balancing, retrain the Random Forest model using the established hyperparameters, validate the retrained model on new 20% holdout sets, recalibrate probability thresholds as needed to maintain operational false positive limits, and deploy the new model only if its performance meets or exceeds the previous model's standards.

For Real-time Monitoring Dashboard Requirements, it is crucial to track 7-day and 30-day rolling averages of key metrics (precision, recall, F1 score, false positives/negatives). The system should monitor feature distributions using statistical tests (e.g., Kolmogorov-Smirnov test) to detect feature drift exceeding a predefined threshold (e.g., 10%), implement automated alerts for performance degradation exceeding a significant threshold (e.g., 2 percentage points), and ensure that the review team's capacity is consistently matched with the volume of flagged transactions to prevent backlogs.

This comprehensive monitoring framework will ensure the model remains effective against evolving fraud tactics while preserving operational efficiency and enabling proactive adjustments.

Limitations & Future Research

9.1 Current Limitations

Despite its strong performance, the current approach has several limitations. There is information loss from undersampling, as discarding approximately 528,000 legitimate transactions removes exposure to rare but valid behavioral patterns, potentially increasing false positives when atypical, yet legitimate, activities occur in production that the model has not sufficiently learned from. SMOTE bias is another limitation; while effective for balancing classes, synthetic oversampling interpolates between existing fraud examples, which might not accurately reflect truly novel or emerging fraud schemes as tactics evolve, potentially limiting the model's ability to generalize to unseen fraud patterns. The model also faces feature drift vulnerability, as fraud patterns continuously evolve, and without vigilant and proactive monitoring, the model's performance can degrade over time as the characteristics of legitimate and fraudulent transactions change, necessitating frequent retraining.

Furthermore, there is limited network analysis, as the current approach largely examines individual transactions in isolation and does not natively incorporate network effects to identify coordinated fraud rings or complex attacks spanning multiple cards, devices, or accounts, which

often evade single-transaction analysis. Finally, reduced interpretability for "V" features is a challenge, as many anonymized "V" features in the dataset limit direct understanding of why specific transactions are flagged, which can reduce confidence among stakeholders and make it challenging to explain model decisions.

9.2 Future Research Directions

To address current limitations and further enhance fraud detection capabilities, we propose several areas for future research. Ensemble methods offer an opportunity to combine the strengths of multiple models (e.g., Random Forest, XGBoost, and LightGBM) in meta-learning frameworks (e.g., stacking or boosting ensembles), which can leverage complementary strengths to improve overall performance, robustness, and generalization. Graph-based anomaly detection can be implemented using graph neural networks or other graph-based algorithms to leverage transaction-entity networks, identifying coordinated fraud rings that exploit relationships between users, devices, and transactions, which are often missed by individual transaction analysis.

Developing streaming learning implementation would enable near-real-time model updates as new fraud labels become available, significantly reducing the adaptation lag to evolving fraud tactics and allowing the model to respond more dynamically. Incorporating behavioral biometrics would involve integrating user interaction patterns, such as mouse movements, typing dynamics, and clickstream data, as features, adding an extra layer of authentication and providing enhanced fraud scoring in web and mobile environments by identifying suspicious user behavior.

Further research could also focus on cost-sensitive optimization, developing models that directly optimize business impact rather than generic accuracy metrics. This involves explicitly balancing the costs of false negatives (missed fraud) against the burden of false positives (unnecessary manual reviews or customer inconvenience), leading to more economically efficient deployment. Finally, advanced feature engineering exploration should include more sophisticated techniques such as temporal sequence modeling (analyzing sequences of

transactions for a given user or card), peer-group comparisons (identifying transactions that deviate significantly from a user's typical behavior or from the behavior of similar users), and velocity-based features (creating features based on the speed or frequency of transactions over short time windows, e.g., number of transactions per hour from a specific IP address).

Conclusion

Effective credit card fraud detection demands a continuous balance between achieving high recall (maximizing fraud detection) and maintaining acceptable precision (minimizing false positives), all while adapting to continuously evolving threats. Through comprehensive analysis of 590,540 transaction records merged with 144,233 identity records, we identified critical fraud patterns and developed a robust detection system.

The key achievements of this project include optimized model performance, with our Random Forest model achieving a strong 73.34% recall and 85.02% precision, alongside an impressive ROC AUC of 0.9176, demonstrating its ability to accurately identify fraudulent transactions while keeping false alarms manageable. We also achieved enhanced operational efficiency by optimizing the probability threshold to 0.69, reducing daily false positives from an estimated 6,240 to under 1,000, which makes manual review operationally sustainable for fraud teams. Critical pattern discovery was a significant outcome, with our exploratory data analysis revealing crucial fraud indicators, including evening spikes, Friday concentrations, the prevalence of low-amount testing transactions, and device-based risk factors (e.g., mobile devices, disposable email domains, and anomalous distance metrics). Finally, we established a scalable framework through a monthly retraining pipeline and continuous monitoring protocols, designed to adapt to new fraud patterns and prevent model degradation.

The strategic impact of the recommended three-tier review workflow, combined with the deployment of the optimized Random Forest model, provides financial institutions with a robust and evolving defense against sophisticated fraud schemes. This approach not only enhances the security posture but also optimizes operational costs by focusing resources on the most critical threats, ultimately safeguarding assets and customer trust.

Appendix A – Code Notebooks

All of the analyses, visualizations, and modeling described in this report are available in the following Jupyter notebooks on GitHub:

Notebook Stage	Description	GitHub Link
Data Wrangling	Initial data import, merging transactions and identity tables, missing-value imputation, dtype optimization, and feature engineering.	https://github.com/Shaymaxo/Capstone-2-Springboard/blob/main/2_Data_Wrangling_Capstone2.ipynb
Exploratory Data Analysis (EDA)	Full-dataset analyses: class distribution, transaction-amount patterns, temporal fraud patterns, categorical risk indicators, correlations, and device/behavioral patterns.	https://github.com/Shaymaxo/Capstone-2-Springboard/blob/main/3_EDA_new_capstone.ipynb

Preprocessing	Imputation pipelines, encoding (label & one-hot), scaling (RobustScaler), train/test split, and SMOTE balancing on the undersampled subset.	https://github.com/Shaymaxo/Capstone-2-Springboard/blob/main/4_pre_processing_capstone.ipynb
Modeling	Classifier definitions (Logistic Regression, Random Forest, XGBoost, LightGBM), hyperparameter tuning, threshold optimization, and evaluation metrics.	https://github.com/Shaymaxo/Capstone-2-Springboard/blob/main/5_modeling_capstone.ipynb