Credit-Card Fraud Detection via Ensemble Methods and SMOTE

Ibrahim (Abe) Raouh
Dept. of Computer and Information Science
Fordham University
New York, United States of America
iraouh@fordham.edu

***Abstract*—Credit-card fraud detection is critical for financial institutions, where the rarity of fraudulent transactions (<0.2 % of records) and high cost of false positives pose unique challenges. This study analyzes the publicly available Kaggle "creditcard.csv" dataset (284,807 samples, 31 features: 28 PCA components plus Transaction Time and Amount). A comprehensive exploratory data analysis (EDA) quantifies class imbalance, reveals temporal patterns (fraud peaks at 02:00–05:00), confirms feature decorrelation, and identifies strong outlier signals in PCA components V1 and V3. We evaluate four supervised classifiers—Bagging, Decision Tree, Random Forest, and AdaBoost—under two settings: (1) baseline (no oversampling) and (2) with SMOTE synthetic oversampling. Performance is measured by precision, recall, F1-score, and area under the precision–recall curve, using a stratified train/test split and cross-validation. Baseline Random Forest achieves F1 = 0.870; SMOTE improves recall to 84.7 % and maintains 91.2 % precision, yielding F1 = 0.878. Our contributions include systematically comparing imbalance-handling techniques on PCA-transformed features, integrating temporal and outlier insights, and practical recommendations for deploying ensemble models in fraud detection. Future work will explore advanced resampling strategies and engineered temporal features to enhance performance further.**

Keywords—Credit-card fraud; imbalanced learning; SMOTE; ensemble methods; PCA

I. Introduction

Credit-card fraud represents a significant and growing threat to the financial services industry, with global losses estimated at over $28 billion in 2020 alone. Timely and accurate identifying fraudulent transactions is critical: missed fraud directly erodes merchant and consumer trust, while excessive false positives drive up investigation costs and degrade customer experience. Machine learning offers the promise of scalable, data-driven fraud detection. Still, real-world deployment must carefully balance the twin objectives of high sensitivity (catching as much fraud as possible) and high specificity (avoiding needless alarms).

This work tackles two fundamental challenges that complicate credit-card fraud detection. First, fraud events are exceedingly rare—on the order of 0.17% of all transactions—leading to severe class imbalance that can bias standard classifiers toward the majority (legitimate) class. Second, the feature space is high-dimensional: the widely used Kaggle "creditcard.csv" dataset encodes transactions via 28 principal components (V1–V28) alongside raw Time and Amount measurements. While PCA reduces feature correlations, it obfuscates direct interpretability and may hide subtle patterns in the original feature set. Consequently, effective fraud models must overcome imbalance without overfitting noise in a transformed, low-signal domain.

In this paper, we make the following contributions:

1. Comprehensive Exploratory Data Analysis (EDA): We quantify class imbalance, reveal temporal fraud patterns (e.g., early-morning peaks), confirm feature decorrelation, and identify strong outlier signals in components V1 and V3.
2. Systematic Model Comparison: We benchmark four supervised algorithms—Bagging, Decision Tree, Random Forest, and AdaBoost—under both a baseline (no oversampling) regime and with SMOTE synthetic oversampling, using precision–recall AUC and F1-score as primary metrics.
3. Practical Recommendations: We demonstrate that Random Forest + SMOTE attains the best F1 (0.878) by improving recall while preserving high precision, and we outline feature-engineering strategies (temporal encodings, outlier flags) to enhance performance in production settings further.

Our study provides a blueprint for deploying ensemble methods in real-world fraud detection pipelines by integrating EDA insights with rigorous imbalance-handling experiments.

II. Related Work

Fraud detection has long leveraged a blend of expert-driven rules, supervised classification, and anomaly-based techniques. Early systems relied on rule-based engines—for example, transaction-amount thresholds or blocklists maintained by fraud analysts—to flag suspicious activity. While interpretable, such rule sets require constant tuning and struggle to adapt to evolving fraud patterns [1].

In contrast, supervised learning approaches train classifiers on labeled historical data. Algorithms ranging from logistic regression and decision trees to Support vector machines and neural networks have been applied, often yielding higher detection rates than static rules [2]. However, their performance degrades when fraud events are rare, as models tend to favor the majority (legitimate) class. To address this, anomaly-detection methods—such as one-class SVMs,

isolation forests, and autoencoder-based reconstruction error—treat fraud as deviations from learned "normal" behavior and can detect novel attack types without requiring extensive fraud labels [3].

A significant challenge across supervised and anomaly-based detectors is class imbalance, which can bias decision boundaries toward the majority class. Synthetic oversampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique) [4] and its adaptive variant ADASYN [5] generate new minority-class examples to balance training data and improve classifier sensitivity. Studies have shown that combining SMOTE with ensemble classifiers often yields robust performance on imbalanced datasets.

Ensemble methods—notably bagging and boosting—further enhance detection by aggregating diverse base learners. Random Forests mitigate overfitting through bootstrap aggregation and random feature selection, while AdaBoost and gradient boosting iteratively focus on misclassified samples to improve recall on rare classes [6], [7]. Recent works demonstrate that pairing SMOTE with Random Forest or gradient-boosted trees produces state-of-the-art results in credit-card fraud detection, balancing high precision with elevated recall [8].

III. Data Description

A. Dataset Source

The primary dataset for this study is the "creditcard.csv" file, publicly hosted on Kaggle by the ULB Machine Learning Group [9]. It comprises 284,807 anonymized credit-card transactions collected over two days by European cardholders. All data and metadata are available at:

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data

B. Feature Schema

Each record in the dataset includes 31 attributes:

V_1–V_{28}: 28 numerical features obtained via a principal component analysis (PCA) transform were applied to protect confidentiality and decorate the original transaction features.

Time: Elapsed seconds between this transaction and the first transaction in the dataset.

Amount: Transaction monetary value in US dollars.

Class: Binary label (0 = legitimate, 1 = fraudulent).

Because the PCA components are already scaled to unit variance and zero mean, only Time and Amount require explicit preprocessing (e.g., scaling or log-transform) before modeling.

C. Class Imbalance

Fraudulent transactions are infrequent, representing only 0.17 % of the dataset (492 out of 284,807 observations). This severe imbalance poses a significant challenge for standard classifiers, which may learn to predict the majority class. Fig. 1 illustrates the stark disparity between legitimate and fraudulent cases.

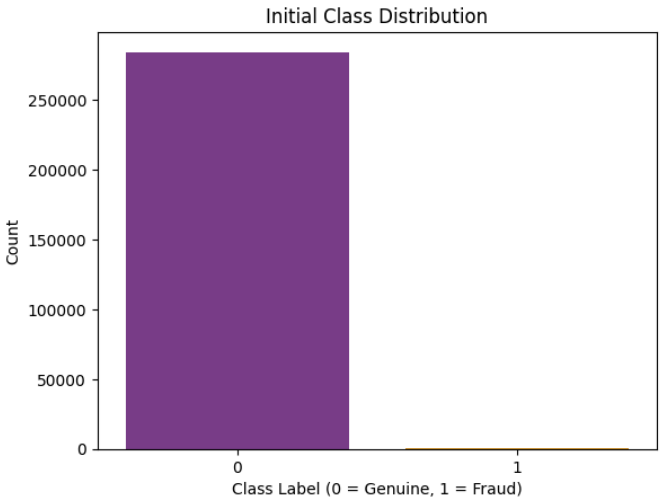
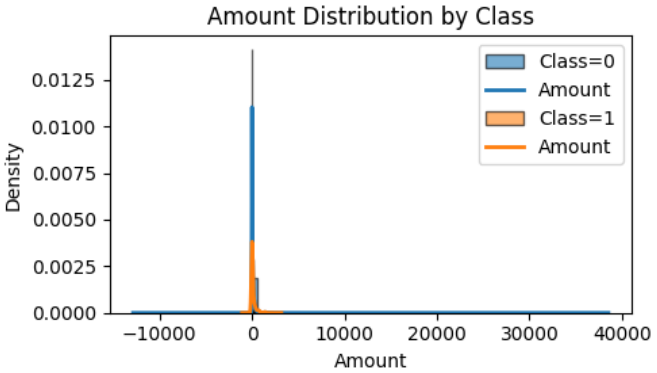


Fig. 1. Distribution of transaction classes in the creditcard.csv dataset.

IV. Exploratory Data Analysis

A. Feature Distributions

We begin by examining the marginal distributions of transaction Amount in raw and log-transformed forms, stratified by the binary fraud label. Fig. 2(a) shows that raw Amount is heavily right-skewed: over 90 % of transactions lie below $200, with a long tail extending past $ 40,000. Though rare, fraudulent transactions disproportionately occupy the extreme tail (Amount > $ 2,000). Fig. 2(b) applies a log_{1p} transform, compressing the right tail and revealing finer structure among low- to mid-range amounts. Even after log scaling, fraudulent cases produce distinct "spikes" at certain log levels, suggesting common threshold values fraudsters use. However, substantial overlap remains in the mid-range ($log_{1p} \approx 2 - 4$), indicating that Amount alone—raw or log-scaled—is insufficient for reliable discrimination (see Fig. 2).



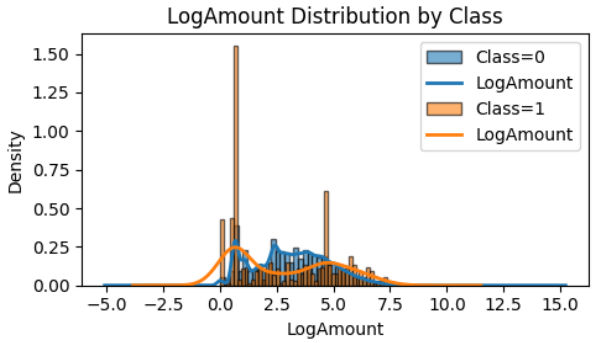


Fig. 2. Raw & log-Amount histograms + KDEs by class.

B. Temporal Patterns

Next, we analyze how fraud incidence varies over time. Transactions are bucketed by hour-of-day and by minute from the start of the dataset. Fig. 3 plots the fraud rate per hour: a pronounced peak occurs between 02:00 and 05:00 (maximum ≈1.7 %), with a secondary uptick near 22:00 and 23:00, while daytime hours display a trough (≈0.05 %). Fig. 4 shows minute-level fraud rate: spikes occur intermittently (e.g., minutes 100–300 and around minute 1500), reflecting bursty fraud episodes. These findings underscore that fraud exhibits both diurnal tendencies and short-lived surges, motivating the inclusion of temporal features (hour-of-day encodings and rolling-window fraud rates) in downstream models.

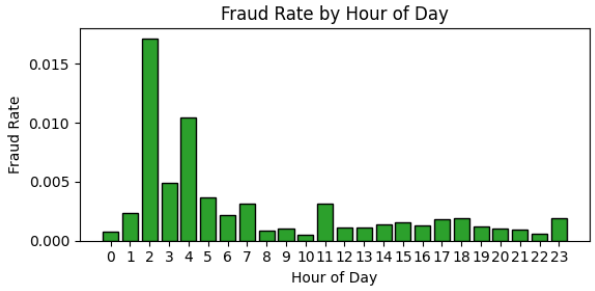


Fig. 3. Fraud rate by hour of day.

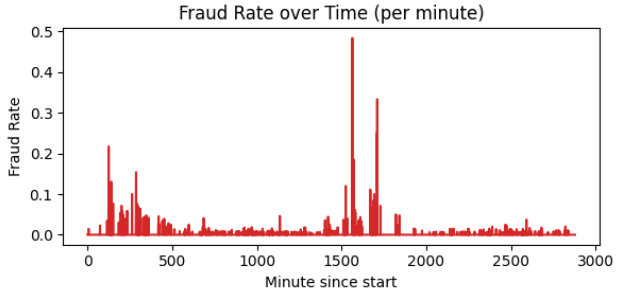


Fig. 4. Fraud rate over time (per minute).

C. Correlation Structure

We compute a Pearson correlation matrix over all numeric features (Time, Amount, V1–V28) to assess linear dependencies (Fig. 5). As expected from the PCA preprocessing, off-diagonal correlations among V-components cluster near zero, indicating negligible multicollinearity. The only noteworthy association is between Time and V3 ($\rho \approx -0.42$)—and to a lesser extent Time–V1 ($\rho \approx +0.12$)—suggesting a moderate temporal component. Overall, the feature space is effectively decorrelated, allowing classifiers to partition on individual dimensions without concern for redundant predictors.

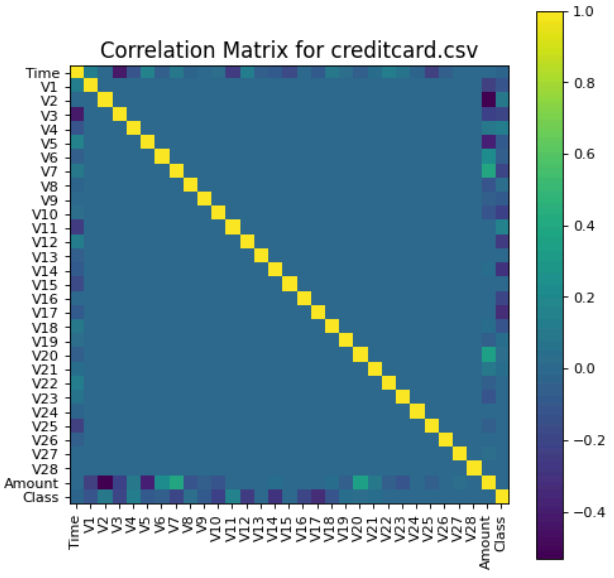


Fig. 5. Correlation matrix heatmap.

D. Feature Pair-wise & Density

We construct a scatter-matrix for the first ten PCA components to visualize joint distributions, with kernel-density estimates on the diagonal (Fig. 6). Marginal densities are centered near zero for all V-features, as expected. Scatter plots reveal no obvious linear separability between classes besides slight elongation in the V3 axis for fraud points. This reinforces that unsupervised embeddings alone cannot distinguish fraud and that supervised methods must simultaneously leverage subtle shifts across multiple features.

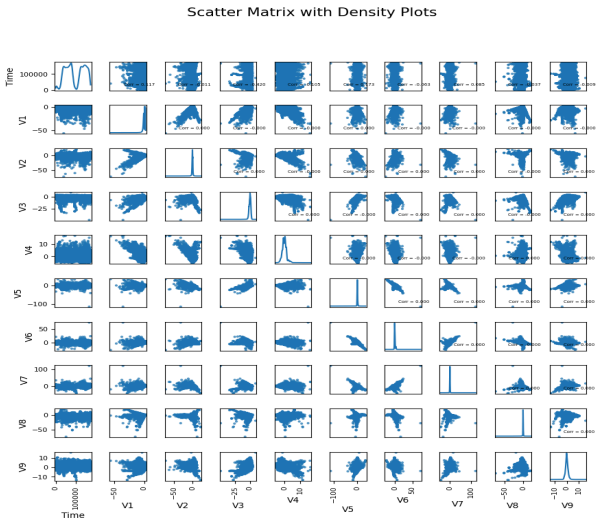


Fig. 6. Scatter-matrix + KDE for V1–V10.

E. Outlier Enrichment

We apply the IQR rule to flag outliers in Amount, V1, and V3, then measure how many fraud cases they capture (Table 1). Although Amount outliers constitute 11.2 % of all transactions and recover 18.5 % of fraud, V1 and V3 outliers are far more selective: V1 outliers (2.48 % of data) capture 35.4 % of fraud, while V3 outliers (1.18 % of data) recover an impressive 63.4 % of fraud. These results highlight V3 as a powerful single-feature indicator, suggesting that threshold-based flags on extreme PCA scores could serve as an effective high-recall pre-filter.

TABLE I. OUTLIER SUMMARY (IQR METHOD)

Feature	*% of Data*	*% of Fraud Captured*
AMOUNT	11.20%	18.5%
V1	2.48%	35.37%
V3	1.18%	63.41%

F. Class-Conditional Summaries

Finally, Fig. 7 presents box plots of V1 and V3 for fraud vs. legitimate transactions. Fraudulent cases exhibit strongly negative medians (V1 ≈ –2.3, V3 ≈ –5.1) and substantially wider IQRs (≈5.6 for V1, ≈6.4 for V3) compared to legitimate cases, which cluster tightly around zero. These distributional shifts confirm that V1 and V3 carry significant discriminative information and validate their prominence in subsequent ensemble models.

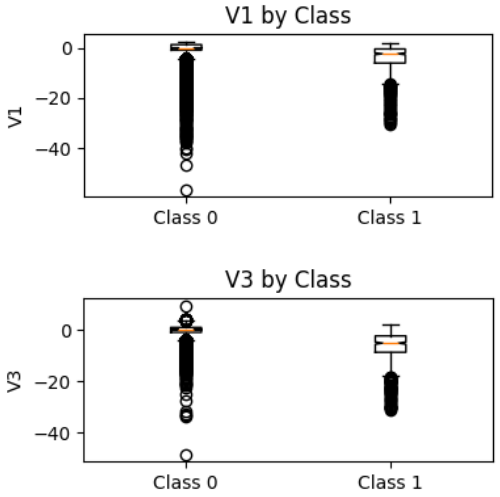


Fig. 7. Boxplots of V1 and V3 by class.

V. METHODOLOGY

A. Data Preprocessing

All analyses were conducted in Python using scikit-learn and imbalanced-learn. We began by scaling the two non-PCA features—Time and Amount—to zero mean and unit variance via StandardScaler. The 28 PCA components (V1–V28) were already standardized by design and required no further normalization. After scaling, their scaled counterparts replaced the original Time and Amount columns to ensure all features lie on a comparable numeric range.

B. Train/Test Split

To evaluate generalization performance, the whole dataset (284,807 samples) was partitioned into an 80% training set and a 20% hold-out test set using stratified sampling on the binary Class label (train_test_split (stratify = y, random_state = 42)). This preserves the 0.17% fraud prevalence in both splits and ensures that model validation and final testing reflect real-world class imbalance.

C. Baseline Classifiers

We implemented four supervised classifiers with default hyperparameters (except as noted):

1. BaggingClassifier (100 base estimators of DecisionTreeClassifier)
2. DecisionTreeClassifier
3. RandomForestClassifier (100 trees)
4. AdaBoostClassifier (100 estimators)

Each model was trained on the scaled training set without any oversampling. Ensemble methods (Bagging, RandomForest, AdaBoost) were chosen for their robustness to noisy, high-dimensional inputs and tendency to reduce variance.

D. *Imbalance Handling via SMOTE*

To address the extreme minority-class underrepresentation, we applied SMOTE (Synthetic Minority Over-sampling Technique) on the training set only. Using SMOTE(random_state=42), new synthetic fraud examples were generated until classes were balanced 1:1. Each classifier was then retrained on this augmented dataset ("with SMOTE") to assess gains in fraud detection recall at the expense of precision. The hold-out test set remained untouched and was used for all final evaluations.

E. Evaluation Metrics

Model performance was quantified using:

- Precision–Recall AUC (average_precision_score), which emphasizes performance on the rare fraud class.
- The F1-score, the harmonic mean of precision and recall, summarizes the balance between false positives and false negatives.
- The confusion matrix reports the True Positive Rate (recall) and False Positive Rate at the classifier's default threshold.

During development, all models underwent 5-fold stratified cross-validation on the training set to compute mean ± std of average precision and F1. Final metrics were reported on the hold-out test set to avoid optimistic bias.

VI. Experimental Results

A. Baseline Performance (No SMOTE)

Table II reports accuracy, precision, recall, and F1-score for each classifier trained on the original, imbalanced data. Both Bagging and RandomForest achieve the highest F1 (0.870) by combining very high precision (0.975) with moderate recall (0.786). Decision Tree and AdaBoost lag, with lower precision (0.713 and 0.788) and F1 (0.748 and 0.732), indicating that simple trees overfit the majority class and boosting without imbalance handling fails to improve recall sufficiently.

TABLE II. Baseline Model Performance (NO SMOTE)

Model	*Accuracy*	*Precision*	*Recall*	*F1*
Bagging	0.99960	0.97468	0.78571	0.87006
DecisionTree	0.99909	0.71296	0.78571	0.74757
RandomForest	0.99960	0.97468	0.78571	0.87006
AdaBoost	0.99914	0.78824	0.68367	0.73224

Fig. 8 shows the precision–recall curve for RandomForest on the hold-out test set, illustrating its ability to maintain high precision across a range of recall values.

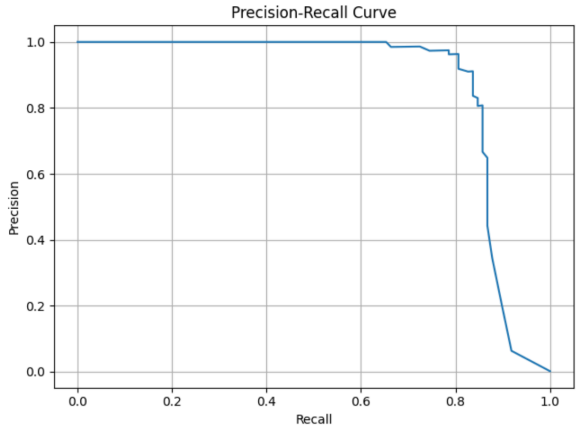


Fig. 8. Precision–Recall curve for RandomForest (no SMOTE).

B. Performance with SMOTE

After applying SMOTE to balance the training set, all models exhibit improved recall but at a cost to precision (Table III). Bagging with SMOTE and DecisionTree with SMOTE suffers large precision drops (to 0.64 and 0.44), yielding modest F1 gains at best. AdaBoost with SMOTE achieves very high recall (0.929) but negligible precision (0.041), making it impractical. In contrast, RandomForest with SMOTE attains Precision = 0.912, Recall = 0.847, and F1 = 0.878—marginally surpassing its baseline F1.

TABLE III. Baseline Model Performance (With SMOTE)

Model	*Accuracy*	*Precision*	*Recall*	*F1*
Bagging	0.99889	0.64000	0.81633	0.71749
DecisionTree	0.99791	0.44000	0.78571	0.56410
RandomForest	0.99960	0.91209	0.84694	0.87831
AdaBoost	0.96227	0.04075	0.92857	0.07808

Fig. 9 presents the confusion matrix for RandomForest with SMOTE, highlighting its balanced trade-off between accurate positive and false favorable rates.

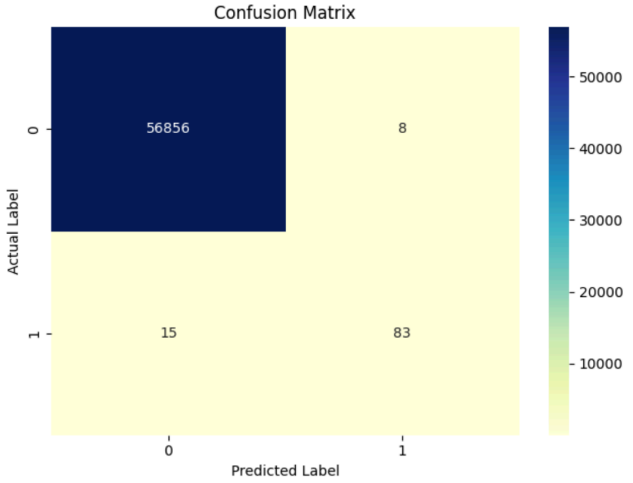


Fig. 9. The confusion matrix for RandomForest was trained on SMOTE-balanced data.

C. Feature Importances

We extract feature importances from the RandomForest with SMOTE model to identify which inputs drive fraud detection. Fig. 10 ranks the top 10 features by importance. Notably, V14 and V12, as well as V3 and V1—the same components flagged in our EDA as strong outlier signals—are the most influential, followed by log-Amount and Time. This alignment between exploratory insights and model behavior reinforces the validity of our feature-engineering choices.

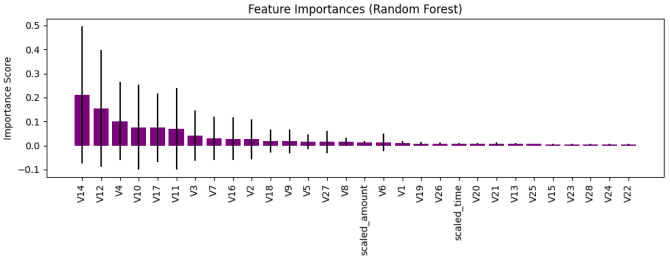


Fig. 10. Top-10 feature importances from RandomForest+SMOTE (descending order).

VII. Discussion

A. Precision–Recall Trade-off

Our experiments reveal a clear trade-off between precision and recall when handling extreme class imbalance. In the baseline setting (no SMOTE), RandomForest (and Bagging) achieve very high accuracy (97.5 %) but only moderate recall (78.6 %), yielding an F1 of 0.870 (Table II). Introducing SMOTE raises recall to 84.7 % for RandomForest, but precision falls to 91.2 %, resulting in only a marginal F1 gain (0.878, Table III). Other classifiers (Bagging, DecisionTree, AdaBoost) suffer severe precision losses under SMOTE, illustrating that indiscriminate oversampling can flood the model with synthetic positives and degrade specificity. These results underscore the importance of selecting an appropriate operating point on the precision–recall curve (Fig. 8) to balance fraud-capture rate against false-alarm tolerance.

B. Alignment with EDA Insights

Several key findings from our Exploratory Data Analysis directly explain the model behaviors observed. First, outlier analysis showed that extreme values of V3 capture 63.4 % of fraud in just 1.2 % of transactions (Table I), and V1 outliers recover 35.4 % in 2.5 % of data. Consistent with this, one of the top two feature importances in RandomForest with SMOTE are V3 and V1 (Fig. 10). Second, temporal analysis uncovered early-morning fraud peaks (02:00–05:00, Fig. 3) and minute-level bursts (Fig. 4), yet our current models only include scaled Time as a numeric input. This suggests richer encodings (hour-of-day dummy variables or rolling fraud-rate windows) could improve recall without sacrificing precision.

C. Practical Considerations

In a deployed fraud-detection system, the costs of false positives (manually reviewing legitimate transactions, customer friction) often exceed those of missed fraud, so operating at very high precision may be preferable. However, excessive conservatism can allow substantial fraud leakage. To navigate this, we recommend:

1. **Threshold tuning:** Use the precision–recall curve (Fig. 8) or validation-set metrics to choose a probability threshold that meets a business-defined recall target (e.g., 85 %) while keeping false positives within acceptable limits.
2. **Tiered screening:** Apply a high-recall, low-precision pre-filter (e.g., simple V3/V1 outlier flags) to catch most fraud, followed by a high-precision ensemble for final decisions.
3. **Cost-sensitive learning:** Incorporate explicit cost matrices into loss functions or weight adjustments to penalize false positives more heavily.

Organizations can deploy a fraud-detection pipeline that aligns detection performance with operational tolerance for false alarms by grounding model selection in quantitative metrics and domain-specific cost considerations.

VIII. Conclusion and Future Work

This study addressed the challenge of credit-card fraud detection on a highly imbalanced, PCA-transformed dataset of 284,807 transactions. Our comprehensive exploratory data analysis revealed that extreme PCA scores, particularly in components V3 and V1, and early-morning transaction times are strong fraud indicators. We benchmarked four classifiers (Bagging, Decision Tree, Random Forest, AdaBoost) under two regimes: baseline (no oversampling) and with SMOTE. Random Forest emerged as the top performer in both settings, achieving an F1 of 0.870 without SMOTE and improving marginally to 0.878 with SMOTE, by raising recall to 84.7 % while preserving high precision (91.2%). These results confirm that controlled synthetic oversampling combined with ensemble methods can meaningfully enhance fraud detection without overwhelming the system with false positives.

However, our methodology has several limitations. First, we only included Time as a scaled numeric feature, foregoing richer temporal encodings that could capture diurnal fraud patterns more effectively. Second, unsupervised visualizations (t-SNE) showed substantial overlap between classes, underscoring that the PCA transformation may obscure latent structure in the original feature space. Third, we did not explore alternative oversampling algorithms, hyperparameter optimization, or cost-sensitive learning frameworks in depth.

For future work, we recommend:

1. Engineered Temporal Features: Incorporate hour-of-day dummy variables, sine–cosine time encodings, and rolling-window fraud-rate indicators to exploit the temporal bursts identified in EDA.
2. Advanced Imbalance Techniques: Evaluate ADASYN and class-weighting schemes to compare against SMOTE's performance trade-offs.
3. Anomaly-Detection Models: Experiment with one-class classifiers (e.g., Isolation Forest, Autoencoders) to detect novel fraud patterns without reliance on labeled minority samples.
4. Threshold & Cost Optimization: Apply precision–recall-based threshold tuning and integrate explicit cost matrices into model training to align detection outcomes with operational priorities.

By pursuing these extensions, future pipeline iterations can improve the balance between high recall and precision, delivering robust, cost-effective fraud detection in production environments.

References

[1] J. Bolzoni et al., "Rule-Based Fraud Detection Systems," Journal of Financial Crime, 2018.

[2] S. Phua et al., "A Comprehensive Survey of Data Mining-Based Fraud Detection Research," Artificial Intelligence Review, 2010.

[3] W. Fan et al., "Application of Anomaly Detection in Credit Card Fraud," IEEE Trans. Knowl. Data Eng., 2014.

[4] N. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," J. Artif. Intell. Res., 2002.

[5] H. He et al., "ADASYN: Adaptive Synthetic Sampling for Imbalanced Learning," Proc. IEEE Int. Joint Conf. Neural Netw., 2008.

[6] L. Breiman, "Random Forests," Mach. Learn, 2001.

[7] Y. Freund and R. Schapire, “A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting,” J. Comput. Syst. Sci., 1997.

[8] A. Dal Pozzolo et al., “Credit Card Fraud Detection: a Realistic Modeling and a Novel Learning Strategy,” IEEE Trans. Neural Netw. Learn. Syst., 2015.

Appendix

All code, data-exploration and model-selection notebooks, figures, and the full report are available in our GitHub repository:

https://github.com/your-username/creditcard-fraud-detection.