

Online Fraud Payment Detection Using Class-Balanced Machine Learning Algorithms

RAMOJU VENKATA SATYA RATNA KUMAR

Assistant Professor

Department Computer Science and Technology

Sanketika Vidya Parishad Engineering College

ratnakumar.cse@svpec.edu.in

JUHI VIJAYALAKSHMI SHRENITA ADIREDDY

Student(M.TECH)

Department Computer Science and Technology

Sanketika Vidya Parishad Engineering College

juhiereddy@gmail.com

Abstract—Online payment platforms have become a crucial component of modern financial services, but they are increasingly targeted by fraudulent activity. Detecting fraudulent payments is difficult because of the class imbalance between genuine and fraudulent transactions: fraud forms a small minority of any transaction log, so a classifier trained on the raw distribution learns to favour the majority class and misses precisely the cases it was built to catch. This project presents a machine-learning fraud-detection system that addresses this imbalance directly by applying the Synthetic Minority Over-sampling Technique (SMOTE) before training, and it quantifies the benefit by evaluating each classifier both with and without balancing. The system is implemented as a Django web application in Python, with scikit-learn for modelling, imbalanced-learn for SMOTE, NumPy and Pandas for data handling, Matplotlib for visualisation, and MySQL for user management. A transaction dataset of 28,213 records labelled Fraud and Non-Fraud is normalised using standard scaling, encoded with a label encoder, and split 80:20, giving 22,570 training and 5,643 test records; SMOTE then expands the training partition to 31,886 records with equal class representation. Random Forest and Naive Bayes are each trained on both the original and the balanced training sets and evaluated on the same held-out test partition. Random Forest with SMOTE performs best, reaching 99.344% accuracy, 99.016% precision, 99.371% recall, and 99.192% F-score, which corresponds to 5,606 of 5,643 test transactions classified correctly, against 97.572% for Random Forest without balancing, 78.611% for Naive Bayes with SMOTE, and 77.884% for Naive Bayes alone. The trained model is then applied to uploaded test data and labels each record as a Fraud or Normal transaction. Overall, the results show that class balancing improves both classifiers and that a balanced Random Forest provides accurate, interpretable, and deployable online fraud detection.

Keywords—Online Payment Fraud; Class Imbalance; SMOTE; Random Forest; Naive Bayes; Machine Learning; Django; Financial Security.

I. INTRODUCTION

With the rapid growth of e-commerce and digital transactions, online payment systems have become an essential part of everyday life. The same convenience that makes them valuable, however, has drawn a corresponding rise in fraudulent activity, creating significant financial and security risk for consumers and businesses alike. Online fraud takes many forms — identity theft, phishing, unauthorised transfers, and account takeover among them — and detecting it reliably has become a central concern in financial technology.

The defining difficulty in this problem is not the choice of classifier but the shape of the data. Fraudulent transactions form a small fraction of any transaction log, and a model trained on that raw

distribution can achieve a high headline accuracy simply by predicting 'legitimate' almost every time. Such a model is worthless in practice: it performs well on the majority class and poorly on the minority class, which is the only class anyone cares about. This is why accuracy alone is a misleading metric in fraud detection, and why precision, recall, and F-score must be reported alongside it.

Two families of remedy exist. Under-sampling discards records from the majority class until the classes are comparable, but in doing so it throws away legitimate information that the model could have used. Naive oversampling duplicates minority records, which balances the counts but encourages the model to memorise the repeated examples rather than generalise from them. SMEAR-free synthetic oversampling avoids both failure modes: the Synthetic Minority Over-sampling Technique (SMOTE) creates new minority examples by interpolating between existing ones in feature space, so the minority class is enlarged with plausible variation rather than exact copies, and no majority data is discarded.

This project builds and evaluates an online fraud-detection system around that idea. Two classifiers — Random Forest and Naive Bayes — are each trained twice, once on the original imbalanced training data and once on a SMOTE-balanced version of it, and all four models are evaluated on the same untouched test partition. This design isolates the contribution of balancing from the contribution of the algorithm, which a single-model experiment cannot do. The system is delivered as a Django web application in which a user registers, logs in, loads and preprocesses the dataset, applies SMOTE, trains and compares the models, and finally uploads new transaction records for classification.

II. LITERATURE SURVEY

Fraud detection under class imbalance has a substantial research literature, and the recurring theme is that the imbalance must be handled explicitly rather than left to the classifier. Dal Pozzolo et al. addressed the problem through cost-sensitive learning and ensemble methods, arguing that conventional accuracy is misleading on skewed data and recommending precision-recall and ROC-AUC curves in its place; they combined random under-sampling with ensembling to improve detection of rare fraudulent transactions. Carcillo et al. later introduced a realistic credit-card dataset and a learning strategy built on balanced random forests, adding consideration of temporal dependence and data drift so that the detector adapts as fraud patterns move, and showing that oversampling combined with ensembling improves performance materially on imbalanced data.

Bhattacharyya et al. provided a comparative study of Logistic Regression, Decision Trees, and Support Vector Machines for fraud detection, emphasising preprocessing and class rebalancing and concluding that decision trees and ensemble models perform best when paired with a rebalancing step. Sahin and Duman took a different route to the same end, building a cost-sensitive decision tree that assigns a higher misclassification penalty to fraudulent transactions, and demonstrating that cost-based learning outperforms conventional training on imbalanced data. Fiore et al. proposed an explicitly hybrid method, using SMOTE for oversampling and AdaBoost for classification, and reported that the combination raises sensitivity to the minority class while keeping the overall error rate low.

Deep learning has also been applied to the problem. Jurgovsky et al. used Recurrent Neural Networks to capture the temporal structure of transaction sequences, training on a large corpus of sequential data

and achieving strong results, but noted that the computational cost and the requirement for large labelled datasets limit their use in real-time systems. Taken together, this literature motivates the approach adopted here. The consistent finding across cost-sensitive, ensemble, and hybrid studies is that a tree ensemble combined with a synthetic oversampling step is both effective and cheap enough to deploy, which is why Random Forest with SMOTE is the proposed model, and why the experiment is designed to measure the balancing step separately from the classifier.

TABLE I. SUMMARY OF SURVEYED LITERATURE

S.No	Author(s)	Approach	Key Finding / Limitation
1	Dal Pozzolo et al. (2015)	Cost-sensitive and ensemble learning	Accuracy misleading on skewed data; use PR and ROC-AUC
2	Carcillo et al. (2018)	Balanced random forests, realistic modelling	Handles temporal drift; oversampling plus ensembling helps
3	Bhattacharyya et al. (2011)	LogReg, Decision Tree, SVM comparison	Trees and ensembles win when paired with rebalancing
4	Jurgovsky et al. (2018)	Recurrent Neural Network on sequences	Captures temporal patterns but costly and data-hungry
5	Fiore et al. (2019)	SMOTE oversampling with AdaBoost	Hybrid raises minority sensitivity at low error cost
6	Sahin and Duman (2011)	Cost-sensitive decision tree	Weighted misclassification cost beats standard training

III. EXISTING SYSTEM AND PROPOSED SYSTEM

A. Existing System

Existing fraud detection in online payment platforms relies primarily on rule-based engines and conventional machine-learning models trained on raw, unbalanced data. Rule-based systems flag transactions against predefined conditions such as a location mismatch, an unusually large amount, or repeated failed login attempts; they are simple and interpretable, but they are rigid, and a fraudster who infers the thresholds can modify behaviour slightly to slip beneath them. Conventional classifiers such as Logistic Regression, Decision Trees, and Naive Bayes learn from historical data instead of fixed rules, but when that data is heavily skewed they inherit the skew: the model becomes biased towards predicting the majority class and fails on exactly the rare events it was built to catch. Where rebalancing is attempted at all, it is often naive — random under-sampling, which discards useful majority data, or duplicate oversampling, which invites overfitting. Many deployed systems also operate in batch rather than in real time, so a fraudulent transfer may complete before it is scored.

Limitations of the existing system:

- **Ineffective handling of imbalanced data:** models trained on skewed data favour the majority class and produce high false negatives.

- **Rigid rule-based logic:** manually defined thresholds do not adapt, and can be inferred and evaded.
- **Lack of real-time detection:** batch processing allows fraudulent transactions to complete before being flagged.
- **Overfitting from naive oversampling:** duplicating minority records makes the model memorise rather than generalise.
- **Information loss from under-sampling:** discarding majority records removes data the classifier could have used.
- **Low interpretability:** black-box models offer no justification for a decision, which regulators and analysts require.

B. Proposed System

The proposed system treats class imbalance as the primary design problem rather than an afterthought. Transaction records are cleaned, label-encoded, and normalised with standard scaling, then split 80:20 into training and test partitions. SMOTE is applied to the training partition only, generating synthetic minority examples by interpolating between neighbouring fraudulent records in feature space until the two classes are equally represented. Applying the balancing after the split, and only to the training side, matters methodologically: the test partition retains its original class distribution, so the reported metrics reflect performance on realistically distributed data rather than on an artificially balanced sample.

Two classifiers are then trained on each version of the training data, giving four models in total — Random Forest and Naive Bayes, each with and without SMOTE — all evaluated on the same held-out test set. This paired design is what allows the contribution of the balancing step to be measured directly rather than inferred. The whole workflow is exposed through a Django web application with user registration and login, so that each stage can be run and inspected in sequence, and a trained model can be applied to newly uploaded transaction records to label them Fraud or Normal.

Advantages of the proposed system:

- **Explicit imbalance handling:** SMOTE synthesises plausible minority examples instead of duplicating or discarding records.
- **Measured, not assumed, benefit:** each classifier is evaluated with and without balancing on the same test partition.
- **Honest evaluation:** balancing is applied only to training data, so test metrics reflect the real class distribution.
- **High detection performance:** Random Forest with SMOTE classifies 5,606 of 5,643 test transactions correctly.
- **Interpretability:** tree ensembles expose feature importance, unlike black-box alternatives.
- **Metrics suited to the problem:** precision, recall, and F-score are reported alongside accuracy.
- **Deployable interface:** a Django application with authentication runs the full pipeline end to end.

IV. METHODOLOGY

A. Dataset Description

The system is trained on an online payment transaction dataset of 28,213 labelled records carrying two class labels, Non-Fraud and Fraud. Each record describes a single payment event through attributes including a time step, the transaction type (TRANSFER, CASH_OUT, CASH_IN, PAYMENT), the amount, identifiers for the originating and destination accounts, and the balances on both sides before and after the transfer. The balance fields are particularly informative: a transfer that reduces an originating balance to zero without a matching increase at the destination is a strong structural signature of fraud, and the prediction output in Fig. 9 shows exactly this pattern in the records the model flags. The class distribution is uneven, with Non-Fraud records substantially outnumbering Fraud records, which is the imbalance the system is designed to address. Table II summarises the dataset and experimental configuration.

TABLE II. DATASET AND EXPERIMENTAL SETUP

Parameter	Value
Total records	28,213
Class labels	Non-Fraud, Fraud
Normalisation	Standard scaling
Categorical encoding	Label Encoder
Training partition (80%)	22,570 records
Testing partition (20%)	5,643 records
Training size after SMOTE	31,886 records (classes equalised)
Classifiers evaluated	Random Forest and Naive Bayes, with and without SMOTE
Evaluation metrics	Accuracy, precision, recall, F-score, confusion matrix

B. Data Preprocessing

Preprocessing runs in three steps before any model sees the data. Non-numeric fields — principally the transaction type and the account identifiers — are converted to numeric form with a label encoder, since neither classifier accepts string input. The feature matrix is then normalised using standard scaling, which centres each feature on zero mean and unit variance. This step is not cosmetic: the amount and balance fields span several orders of magnitude, and without scaling they would dominate any distance or variance-sensitive computation, including the neighbour search that SMOTE performs when it interpolates. Finally the dataset is split 80:20 into 22,570 training and 5,643 test records, and the class distribution is plotted so that the imbalance is visible before modelling begins.

C. Class Balancing with SMOTE

SMOTE is applied to the training partition after the split. For each minority-class record the algorithm locates its nearest minority neighbours in feature space and generates a new synthetic record at a random

point along the line segment joining them. The effect is to enlarge the minority class with plausible variation rather than exact duplicates, which is what distinguishes it from naive oversampling and why it is less prone to inducing overfitting. No majority records are discarded, so nothing is lost from the legitimate class either. On this dataset SMOTE expands the training partition from 22,570 to 31,886 records with the two classes equally represented, implying that the majority class contributed 15,943 records and the minority class was enlarged from 6,627 to match it. The test partition is deliberately left untouched, so the evaluation still reflects the original class distribution.

D. Classification Algorithms

Two classifiers are compared. Naive Bayes applies Bayes' theorem under the assumption that features are conditionally independent given the class. It is fast and requires little data, but the assumption is badly violated here — the balance fields are arithmetically related to the transaction amount — and this is reflected in its results. Random Forest is an ensemble of decision trees, each grown on a bootstrap sample of the data using a random subset of features at every split, with the final prediction taken by majority vote. It suits this problem for two reasons: fraud is characterised by conjunctions of conditions rather than by any single attribute crossing a threshold, and a tree represents such conjunctions natively; and averaging over many decorrelated trees controls the variance that a single deep tree would exhibit. Random Forest also provides feature importance scores, which give the interpretability that regulated financial applications require.

V. SYSTEM IMPLEMENTATION

A. Model Description & Module Implementation

Four metrics are reported for every model. Accuracy gives the overall proportion of correct decisions. Precision gives the proportion of transactions flagged as fraud that really were fraudulent, and therefore measures the false-alarm burden placed on analysts and genuine customers. Recall gives the proportion of actual fraud that was caught, and therefore measures financial exposure. The F-score is their harmonic mean, and is the single figure that matters most on imbalanced data because it cannot be inflated by performing well on the majority class alone. The confusion matrix presents the same information positionally, with correct predictions on the diagonal and errors off it. In the prediction output the model labels each uploaded record as a Fraud or Normal transaction; as Fig. 9 shows, the flagged records are dominated by TRANSFER and CASH_OUT events that zero the originating balance, while CASH_IN and PAYMENT events with consistent balance movements are passed as normal.

Module	Function
Load & Processed Dataset	Loads the dataset, label-encodes and standard-scales it, splits 80:20, and plots the class imbalance
Balanced Data using ML	Applies SMOTE to the training partition and plots the balanced class distribution
Train ML Models	Trains Random Forest and Naive Bayes with and without SMOTE and reports all metrics

Module	Function
Detect Fraud	Uploads new transaction data and labels each record Fraud or Normal using the best model

VI. RESULTS

All four models were evaluated on the same 5,643-record test partition, which retains the original class distribution. Table V reports the results. Random Forest with SMOTE performs best on every metric, reaching 99.344% accuracy, 99.016% precision, 99.371% recall, and 99.192% F-score. Random Forest without balancing follows at 97.572% accuracy, then Naive Bayes with SMOTE at 78.611%, and Naive Bayes alone at 77.884%.

TABLE V. COMPARATIVE PERFORMANCE ON THE 5,643-RECORD TEST SET

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)
Random Forest + SMOTE	99.344	99.016	99.371	99.192
Random Forest + WITHOUT SMOTE	97.572	96.856	97.159	97.006
Naive Bayes + SMOTE	78.611	75.252	67.306	69.193
Naive Bayes + WITHOUT SMOTE + WITHOUT SMOTE	77.884	74.628	65.533	67.294

Because the test partition is fixed, these percentages correspond to whole numbers of transactions, and expressing them that way makes the practical size of each difference easier to judge. Table VI gives this view.

TABLE VI. CORRECT PREDICTIONS OUT OF 5,643 TEST TRANSACTIONS

Algorithm	Correct	Incorrect	Accuracy (%)
Random Forest + SMOTE	5,606	37	99.344
Random Forest + WITHOUT SMOTE	5,506	137	97.572
Naive Bayes + SMOTE	4,436	1,207	78.611
Naive Bayes + WITHOUT SMOTE	4,395	1,248	77.884

A. Effect of Class Balancing

SMOTE improves both classifiers, but by very different amounts, and the difference is informative. For Random Forest, balancing raises accuracy from 97.572% to 99.344% and F-score from 97.006% to 99.192%; in absolute terms it reduces the error count from 137 transactions to 37, cutting misclassifications by roughly three quarters. Recall rises from 97.159% to 99.371%, which is the change

that matters operationally, since recall is the measure of fraud that escapes detection. For Naive Bayes the improvement is real but marginal: accuracy moves from 77.884% to 78.611% and F-score from 67.294% to 69.193%, recovering only 41 additional transactions. Balancing the data therefore helps a model that is capable of using the extra minority examples, and helps very little when the model's own assumptions are the binding constraint.

B. Comparison of the Classifiers

The gap between the two algorithms is far larger than the gap created by balancing. Random Forest outperforms Naive Bayes by roughly twenty percentage points of accuracy and thirty of F-score under both conditions, and the reason is the independence assumption. Naive Bayes treats every feature as conditionally independent given the class, but in this dataset the balance fields are arithmetically tied to the transaction amount, so the assumption is violated by construction and the model double-counts correlated evidence. Its recall of 65.533% without balancing means roughly a third of fraudulent transactions pass undetected. Random Forest makes no such assumption; each tree can condition one attribute on another, which is exactly what is needed to represent the conjunction of a transfer type with a balance movement that fails to reconcile. The practical implication is that attention is better spent on model class than on preprocessing when the two compete for effort.

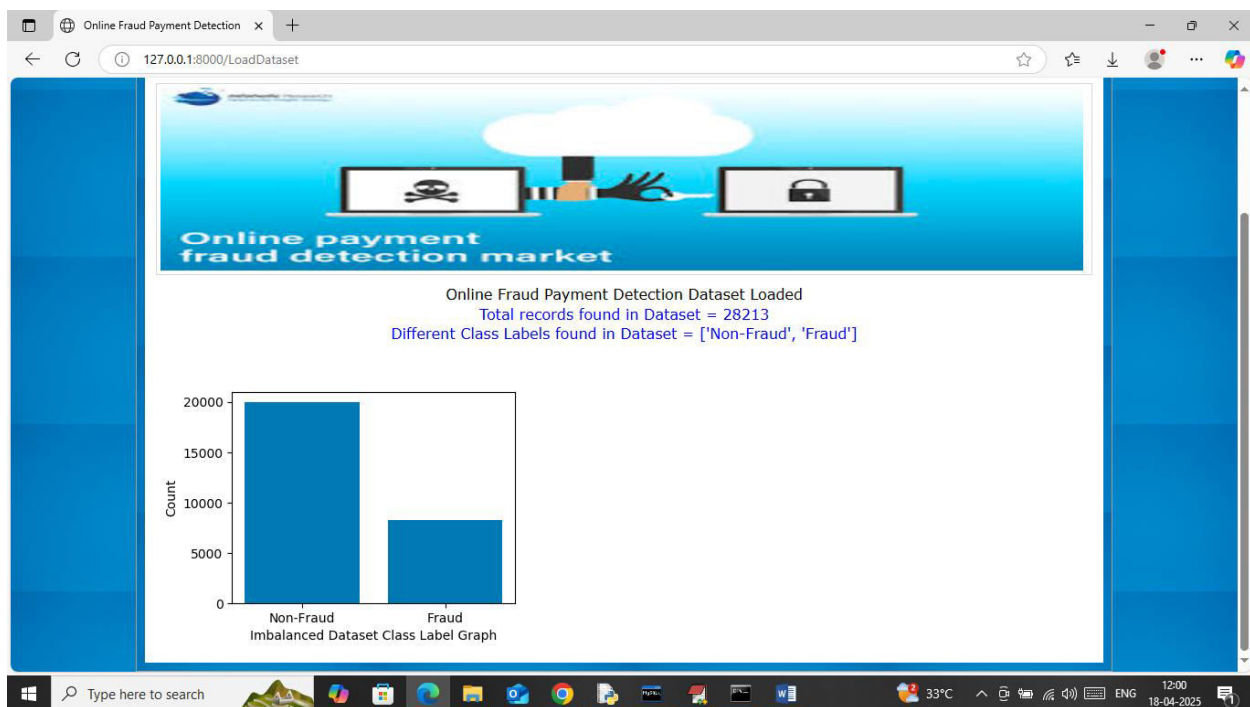


Fig. 1. Dataset loaded, with the imbalanced class distribution.

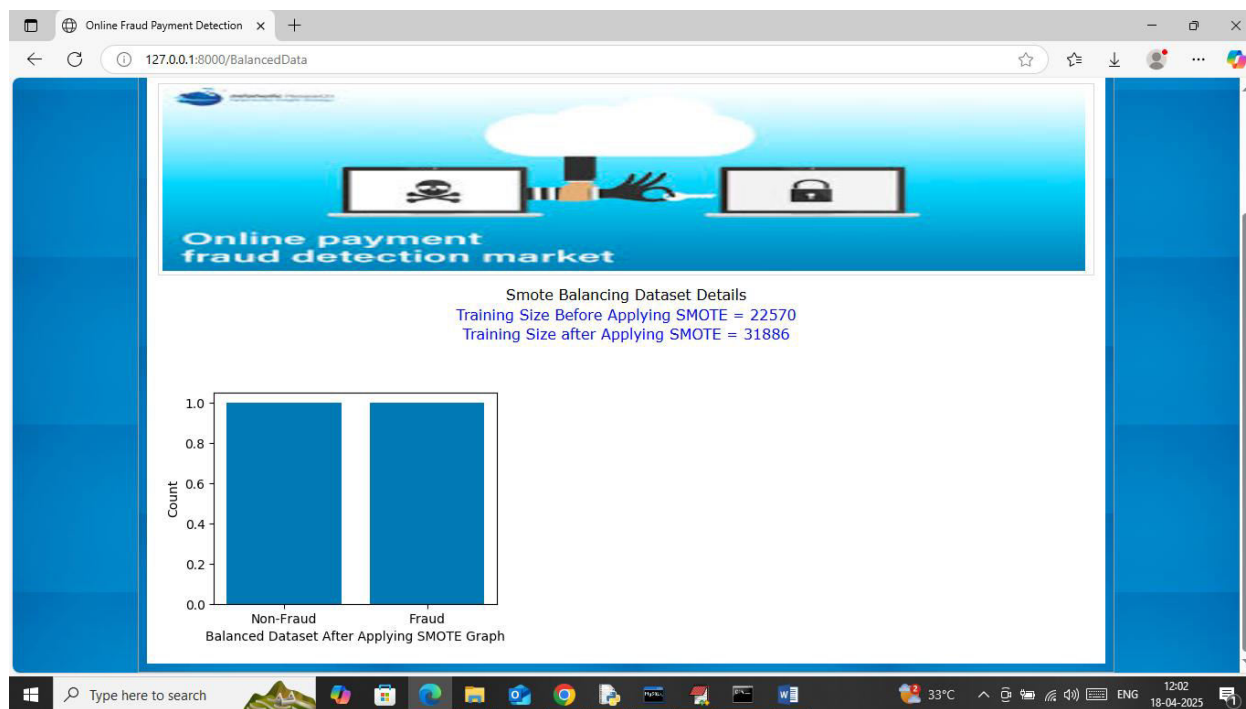


Fig. 2. SMOTE balancing details and the balanced class distribution.

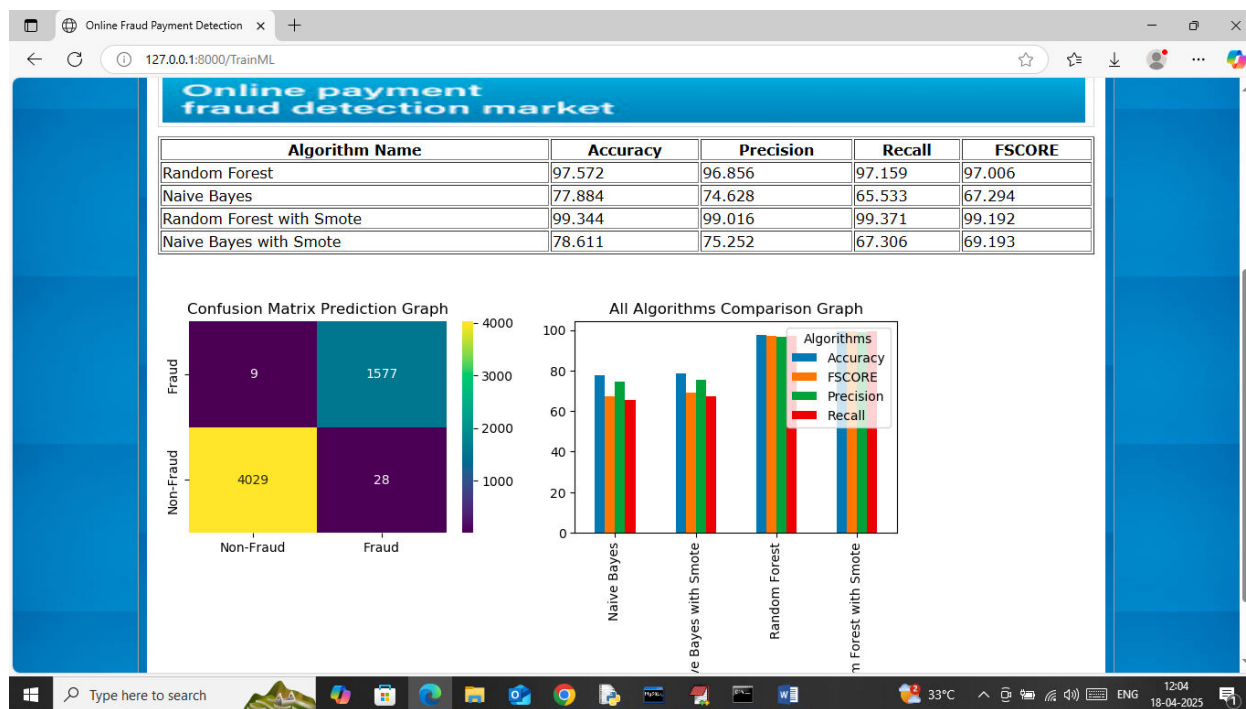


Fig. 3. Model metrics, confusion matrix, and comparison graph.



Test Data	Detection Status
[60 'TRANSFER' 45807.51 'C958487787' 45807.51 0.0 'C218393296' 0.0 0.0]	Fraud Transaction
[60 'CASH_OUT' 45807.51 'C981996349' 45807.51 0.0 'C1099066639' 267824.05 313631.56]	Fraud Transaction
[3 'CASH_IN' 399133.53 'C1509715205' 3111318.61 3510452.14 'C2063763486' 450075.95 50942.42]	Normal Transaction
[3 'CASH_IN' 99252.4 'C2016442966' 3510452.14 3609704.54 'C1850180796' 430778.28 331525.89]	Normal Transaction
[3 'PAYMENT' 2317.06 'C1770453310' 46139.96 43822.9 'M113042195' 0.0 0.0]	Normal Transaction
[3 'CASH_IN' 254872.55 'C958817283' 7408.0 262280.55 'C569022831' 786.0 0.0]	Normal Transaction
[60 'TRANSFER' 742973.53 'C153820103' 742973.53 0.0 'C1530694083' 0.0 0.0]	Fraud Transaction
[60 'CASH_OUT' 742973.53 'C261473899' 742973.53 0.0 'C897961171' 0.0 742973.53]	Fraud Transaction
[60 'TRANSFER' 581421.85 'C1021314422' 581421.85 0.0 'C2062695733' 0.0 0.0]	Fraud Transaction
[60 'CASH_OUT' 581421.85 'C1175189548' 581421.85 0.0 'C340833561' 407848.96 989270.81]	Fraud Transaction
[3 'CASH_OUT' 226424.35 'C436126678' 0.0 0.0 'C977993101' 1748436.14 1615874.57]	Normal Transaction

Fig.4. Prediction output labelling each transaction Fraud or Normal.

VII. CONCLUSION AND FUTURE WORK

This project demonstrates that class balancing is a decisive step in building an effective online fraud-detection system, and that its benefit can be measured rather than assumed. By applying SMOTE to the training partition only and evaluating every model on an untouched test set, the system isolates the contribution of balancing from the contribution of the classifier. On a dataset of 28,213 transactions split 80:20 and expanded from 22,570 to 31,886 balanced training records, Random Forest with SMOTE classified 5,606 of 5,643 held-out transactions correctly, achieving 99.344% accuracy, 99.016% precision, 99.371% recall, and 99.192% F-score, and outperformed the unbalanced Random Forest and both Naive Bayes variants on every metric. Two findings emerge from the comparison. Balancing helps a model that can exploit the additional minority examples — it cut Random Forest's errors from 137 to 37 — and helps far less when the model's own assumptions are the limiting factor, as with Naive Bayes. And the choice of model class matters more than the preprocessing: the gap between the two algorithms is roughly twenty points of accuracy under either condition. Delivered as a Django application with authentication, the system runs the whole pipeline end to end and labels newly uploaded transactions as Fraud or Normal.

The clearest priority for future work follows from the limitations above: evaluation on a realistically skewed dataset in which fraud accounts for well under 1% of transactions, with k-fold cross-validation in place of a single split and per-class metrics reported for the fraud class specifically. Beyond that, several directions are worth pursuing. Comparing SMOTE against ADASYN, borderline-SMOTE, and cost-sensitive learning would establish which balancing strategy suits this data best, rather than assuming the first one tried. Gradient-boosted ensembles such as XGBoost would test whether further gains remain available beyond Random Forest. Recurrent architectures would exploit the sequential structure of a

cardholder's transaction history, which the present per-transaction feature representation discards. Explainability tools such as SHAP or LIME would let an analyst see which attributes drove a particular flag, which matters for both analyst trust and regulatory compliance. Finally, deployment behind a streaming pipeline with periodic retraining and a feedback loop from confirmed fraud cases would let the model track tactics as they evolve rather than as they stood on the day it was trained.

REFERENCES

- [1] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit Card Fraud Detection through Cost-Sensitive and Ensemble Learning," 2015.
- [2] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," 2018.
- [3] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data Mining for Credit Card Fraud: A Comparative Study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.
- [4] J. Jurgovsky et al., "Sequence Classification for Credit-Card Fraud Detection," *Expert Systems with Applications*, vol. 100, pp. 234–245, 2018.
- [5] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using Generative Adversarial Networks for Improving Classification Effectiveness in Credit Card Fraud Detection," *Information Sciences*, vol. 479, pp. 448–455, 2019.
- [6] Y. Sahin and E. Duman, "Detecting Credit Card Fraud by Decision Trees and Support Vector Machines," in *Proc. Int. MultiConference of Engineers and Computer Scientists*, 2011.
- [7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [8] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [9] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [10] M. Buda, A. Maki, and M. A. Mazurowski, "A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.
- [11] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," in *Proc. 22nd ACM SIGKDD*, 2016, pp. 1135–1144.
- [12] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [13] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.
- [14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.
- [15] Django Documentation. [Online]. Available: <https://docs.djangoproject.com/>
- [16] Scikit-learn Documentation. [Online]. Available: <https://scikit-learn.org/>