Explainable AI for Extreme Class Imbalance: Interpreting Latent Variables in European Credit Card Transactions

Rohan Srikanth[1]

Received June 23, 2026
Accepted September 30, 2026
Electronic access November 30, 2026

Credit card fraud presents major challenges to our financial ecosystem, and recent studies have been using supervised learning to detect it. Extreme class imbalance, which is common in real-world data, usually hides the true performance of detection models. Techniques like Principal Component Analysis can even turn these models into "black boxes". In this paper, we look at the tradeoff between accuracy and interpretability by comparing XGBoost and Logistic Regression on a dataset of European credit card fraud transactions, which exhibits this extreme class imbalance. We used SMOTE to mitigate this imbalance, as well as leakage, inside the training folds during each iteration of stratified 5-fold cross-validation. XGBoost outperformed Logistic Regression when ranking positive cases, with a PR-AUC of 0.8504 ± 0.0332 compared to 0.7413 ± 0.0283, and a Nadeau-Bengio/Holm-adjusted p-value of 0.0022. To add to this, at the fixed F1 threshold, holdout F1, precision, and recall were 0.8364, 0.9857, and 0.7263 respectively for XGBoost, compared to 0.8136, 0.8780, and 0.7579 for Logistic Regression. These differences, however, were not statistically significant. Using the Interventional TreeExplainer in probability space, we were able to identify a few PCA components (V14, V12, and V4) that are primarily responsible for the model's decision.

Keywords: Explainable AI, Credit Card Fraud, XGBoost, SHAP, Extreme Class Imbalance

Introduction

Credit card fraud – especially card-not-present fraud – is a rare and organized crime that follows a sequence of steps (a crime script)[1]. However, when a legitimate transaction is flagged or a fraudulent transaction is missed, its consequences can get really expensive[2]. This means that machine learning models would need to rank risky transactions and choose a cutoff. Accuracy alone would not suffice because of the extreme class imbalance notable in real financial transactions.

Because of this issue, researchers started building cost-sensitive models and choosing thresholds. For example, Almhaithawi et al. treat cost as example-dependent. They combine this with SMOTE, which addresses extreme class imbalance during learning, and a Bayes minimum-risk decision, which then chooses a threshold based on cost[3]. Sang (2026) argues that with extreme class imbalance, it is not enough to just train a cost-sensitive model; you have to optimize the threshold[4]. However, Islam et al. show how when fraud is rare and both classes overlap, there is always a tradeoff between false positives and false negatives[5]. Many papers already treat cost and optimizing the threshold as important, but locking an F1 threshold inside nested cross-validation is less established.

The default model in credit card fraud detection – besides costs and thresholds – are tree ensembles or benchmark comparisons. Randhawa et al., Zhao et al., and Suryanarayana et al. all use boosting methods: AdaBoost[6], LightGBM[7], and Catboost[8], respectively. Other researchers employ several different models and examine their performance in credit card fraud detection. Alarfaj et al. and Gamal et al. compare standard machine learning models (logistic regression, trees, and SVMS) to deep learning models[9,10].

Al-Bulushi et al. and Farabi et al. compared supervised learning models and ensemble models, and reported their performance metrics[11,12]. Tiwari et al. conduct a survey of credit-card fraud methods and compare the pros and cons of earlier studies[13]. Lastly, Praba et al. survey standard machine learning models across their general applications[14]. These papers usually use accuracy or F1 score as their main metric. They rarely lock the F1 threshold inside nested CV or use a Nadeau-Bengio corrected t-test to actually treat the overlapping CV folds as overlapping.

Other studies use graphs, transaction sequences, and deep neural networks to detect credit card fraud. Motie & Raahemi and Cheng et al. survey graph neural networks (GNNs) for fraud detection. They treat accounts, merchants, devices, and cards as a network where everything is linked[15,16]. Cherif et al. and Wei & Lee put this into action by creating graph neural networks with payments, cards, or merchants as the nodes. Cherif et al. created an encoder-decoder GNN for credit card

[1] *Denmark High School, Georgia, US*

© The National High School Journal of Science 2026

fraud[17] while Wei & Lee created a temporal heterogeneous GNN for transaction fraud[18]. While these are the most specific examples, other studies have created GNNs with other variables – like user accounts[19], supply-chain partners[20], or colluding accounts[21] – as the nodes. Other studies by Benchaji et al.[22,23] and El Hachimi et al.[24], have used LSTMs (Long Short Term Memory) so that the model can follow payment histories. Yu et al. and Deng et al. use transformers for these, although in different sectors: Yu et al. use it for cards[25] while Deng et al. use it for streaming financial fraud[26]. Deep learning models are also used without graphs or long histories. For example, Fanai et al.[27] and Bao et al.[28] use autoencoders and anomaly detection respectively. These approaches require a network or an ordered payment history however, which this paper does not study, so they are not missing baselines.

However, whether the model is a decision tree or a neural network, it is still important to understand why the model flagged the transaction. Rudin argues that for important, high-risk decisions, you should use naturally interpretable, not a black-box model plus an explainer like SHAP[29]. Despite this warning, researchers and banks still use Explainable AI to explain tree models, just not for detecting fraud. For example, Bussmann et al. use it for predicting loan defaults[30]. Bahlool et al. review this literature more broadly and illustrate how work on explainability and performance is mostly concentration in credit scoring, not fraud detection[31]. Even so, there is still a trade-off between making a model more secure and making it more accurate. If you optimize the model to be more secure, the accuracy often drops, and vice versa[32]. So, while studies have focused on explainability and fairness, they have usually focused on them in the context on credit scoring, not credit card fraud

To evaluate the model, we used a leak-free nested cross validation protocol with a fixed F1 threshold. This study asks three research questions:

- RQ1: After using nested cross-validation, inner-out-of-fold F1 thresholding, and the same SMOTE pipeline, does XGBoost outperform L2 Logistic Regression on threshold-free ranking, specifically PR-AUC, on the holdout set?
- RQ2: Do F1/precision/recall/FPR/balanced accuracy – at that locked F1 threshold – pass a Nadeau-Bengio corrected paired t-test and a Holm's procedure at α=0.05?
- RQ3: Which PCA features dominate mean absolute SHAP in the probability space (real-prevalence background), and do true positive, false positive, and false negative cases share these patterns?

We came up with the following hypotheses to answer these research questions:

- H1: After the same nested cross-validation protocol, XGBoost will outperform L2 Logistic Regression on threshold-free ranking. Nested-CV and holdout PR-AUC should both be higher for XGBoost, and this gap should remain significant after a Nadeau-Bengio corrected paired t-test and Holm adjustment.
- H2: XGBoost will outperform L2 Logistic Regression on F1, precision, recall, FPR, and balanced accuracy at the locked F1 threshold. These differences should also remain significant after the t-test and Holm adjustment.
- H3: A small number of PCA variables will be responsible for most mean absolute SHAP in probability space against a real-prevalence background. True-positive and false-positive cases should share this pattern, while a missed fraud should not.

This paper presents the following contributions on to the current body of literature:

1. A nested cross-validation protocol that uses a stratified split before scaling, SMOTE only in the training folds, a matched hyperparameter-search budget (`n_iter`=10), and a locked F1 threshold that is locked on inner out-of-fold scores and fixed for the outer fold and the holdout set.
2. A comparison of XGBoost and L2 Logistic Regression that used Nadeau-Bengio corrected paired t-test, Holm adjustment across six metrics, and the separation of threshold-free metrics like PR-AUC from threshold-dependent metrics like F1, precision, recall
3. Interventional TreeExplainer in probability space, using a real-prevalence background, with global mean absolute SHAP values and TP, FP, and FN waterfall plots so the strongest features can be checked on errors as well.

Methods

Dataset

This study uses the European Credit Card Fraud dataset, originally created by the Machine Learning Group at Université Libre de Bruxelles (ULB)[33] to study how to calibrate class probabilities when dealing with extreme class imbalance. The dataset contains 284,807 real-world credit card transactions made by European cardholders in September 2013. Accounting for the extreme class imbalance in real-world credit card fraud, only 492 (0.172%) of the transactions are labeled as fraudulent. The dataset contains 28 numerical features (V1-V28) that are masked by PCA to protect consumer privacy. These represent the latent variables. Time and Amount are the

© The National High School Journal of Science 2026

only two variables not anonymized by PCA. Finally, the Class feature is the target variable, indicating whether the transaction was fraudulent (1) or legitimate (0).

Data Preprocessing

To address the extreme class imbalance, we used the Synthetic Minority Oversampling Technique (SMOTE)[34]. SMOTE was configured with a default sampling strategy of 'auto' to achieve a perfectly balanced training distribution. We set the number of nearest neighbors (k) to 5 for synthetic interpolation and used a random state seed of 42 to ensure reproducibility.

We used SMOTE over alternative sampling and cost-sensitive methods due to the geometric distribution of the PCA-transformed data. Cost-sensitive methods like class weighting penalize misidentifying the minority class, but they do not provide tree-based algorithms like XGBoost with the diverse data that they need to form non-linear decision boundaries. To add to this, ADASYN adaptively creates synthetic data near complex decision boundaries, but it risks exaggerating noise and outliers. Hybrid methods like SMOTEEN were also not used in order to keep the full scale and behavioral complexity of the majority class. SMOTE provided the optimal balance that was needed to address the extreme imbalance.

To mitigate data leakage, SMOTE was applied only inside the training folds during each iteration of cross-validation. The original data kept its extreme class imbalance to more accurately model reality. We also applied standard scaling to the Time and Amount variables before model training to ensure that subsequent SHAP values were not skewed by variances and to ensure uniformity across all features.

Because the dataset contained no missing values, data imputation was not required. However, we did identify and drop 1081 duplicate rows before the train-test split to mitigate leakage across cross-validation folds. This left 283,726 total transactions and 473 (0.167%) fraudulent ones. We also winsorized the Amount values at the 0.5th and 99.5th percentiles and standardized the Time and Amount features with a scaler. Both were done only on the training split. PCA-transformed variables were left unscaled. Even though roughly 11.17% of the Amount values were considered outliers, these were not deleted. Any extreme values were capped so that they didn't skew the scaler.

Data Analysis

We performed an exploratory data analysis using the pandas library in Python in order to understand the distribution of Time and Amount. We did this to locate potential similarities and differences in the distributions of the two classes. We also used SHAP (SHapley Additive exPlanations)[35] to identify which variables most influenced the model's result. This is especially necessary due to the 'black box' problem we are investigating in this paper, because SHAP will be able to explain the model's decisions even and especially when the features lose their original meaning.

Model Architecture and Implementation

We compared XGBoost and Logistic Regression, using Logistic Regression as the baseline. While both models had their hyperparameters optimized via RandomizedSearchCV with 3 inner folds, `n_iter` = 10, and scoring = PR-AUC, the models used different hyperparameters. XGBoost's hyperparameters `n_estimators` = 182, `max_depth` = 7, `learning_rate` = 0.1203, subsample = 0.7837, `colsample_bytree` = 0.7498, `min_child_weight` = 5, gamma = 1.4261, `reg_alpha` = 0.00607, and `reg_lambda` = 0.1479. XGBoost also had a few hyperparameters that were fixed. These were: objective = binary:logistic, `eval_metric` = aucpr, and `tree_method` = hist. For Logistic Regression only one hyperparameter, which was C = 3.470, was optimized. The others (penalty = L2, solver = liblinear, `max_iter` = 2000) were fixed. Both XGBoost and Logistic Regression also had different F1 thresholds, with XGBoost at 0.9773 and Logistic Regression at 0.9999994. Both models were run on an ARM processor (macOS-15.7.5-arm64), with Logistic Regression having an inference time and holdout training time at 0.257 $\frac{\mu s}{\text{transaction}}$ and 1.23 s respectively. This is much lower than XGBoost's 1.71 $\frac{\mu s}{\text{transaction}}$ and 5.01 s.

Splitting (Testing and Training)

We split the dataset with a stratified 80/20 split (`random_state` = 42). The 20% holdout data was left untouched until final testing and SHAP analysis while the remaining 80% was trained and tuned using nested cross-validation. The outer loop used five stratified folds with `n_splits` = 5, shuffle = True, and `random_state` = 42. We selected hyperparameters for XGBoost and Logistic Regression within each outer training fold by using 3-fold RandomizedSearchCV with the same search budget. We optimized for PR-AUC and did not use early stopping. After the hyperparameters were selected, we chose a single decision threshold to maximize the F1 score for fraudulent transactions on inner out-of-fold predictions and fixed that threshold. Stratification preserved class proportions within every split and SMOTE was applied only in the outer training folds.

Metrics

Due to the extreme class imbalance of the dataset, accuracy was discarded as a primary evaluation metric, as a classifier

© The National High School Journal of Science 2026

could achieve near-perfect accuracy simply by predicting the majority class. Instead, model performance was evaluated using PR-AUC as the predeclared ranking metric and inner-search objective. Precision, recall, F1, FPR, and balanced accuracy were calculated at the F1-maximizing threshold locked on the inner out-of-fold scores. Calibration was summarized with the Brier score. Recall was not optimized on the holdout set.

Results

We performed an exploratory data analysis on the cleaned ULB dataset to determine its characteristics. Figure 1(a) shows the distribution of both classes in the cleaned dataset, with 283,253 legitimate transactions and 473 fraudulent transactions (0.167%), showing the dataset's extreme class imbalance. A logarithmic scale was used so that the fraudulent bar would be visible. Due to the dataset's extreme class imbalance, accuracy and ROC-AUC would be impressive just by guessing every transaction as legitimate. The model would achieve a 99.833% accuracy. This is why we chose to use PR-AUC and F1-score, among other metrics, because they account for the extreme class imbalance.

In addition to plotting the distribution of both classes, we also illustrated the distribution of the Time and Amount. As illustrated in Figure 1(b), legitimate transactions showed a higher median transaction amount at \$22.00 vs. \$9.82 for fraud, showing that fraudulent transactions are typically of lower value than legitimate transactions. However, the mean for fraudulent transactions is higher than the mean for legitimate transactions, at \$124.00 vs. \$88.00. While both distributions are right-skewed and show significantly higher mean transaction amounts than their medians, the small minority of large fraudulent transactions gives fraudulent transactions the higher mean. Even so, extremely large charges usually indicate a legitimate transaction, and most transactions – both legitimate and fraudulent – usually stay below \$100. The kernel density estimate in Figure 1(c) illustrates the distribution of transactions across time. Both classes showed bimodal distributions, albeit with different peaks. Fraudulent transactions usually occurred during night-time or early morning hours, when volume and monitoring of transactions is usually low. Both variables show patterns, but not significant enough to show causation, meaning V1-V28 still have to be analyzed.

We evaluated both XGBoost and Logistic Regression with PR-AUC, F1 score, precision, recall, balanced accuracy, and FPR. Both were evaluated using nested 5-fold cross validation and on the holdout set. Table 1 shows the performance of each model on each evaluation. XGBoost achieved a better threshold-free ranking with a nested CV PR-AUC of 0.8504 $\pm$ 0.0332 vs. 0.7413 $\pm$ 0.0283 for Logistic Regression. On the holdout set, XGBoost achieved a better PR-AUC (0.811 vs. 0.693), precision (0.986 vs. 0.878), and F1-score (0.836 vs. 0.814) than Logistic Regression, but achieved a slightly worse recall (0.726 vs. 0.758). The difference in PR-AUC is also shown in the precision-recall curves in Figure 2(c). These results show that while XGBoost identified fewer true positive transactions than Logistic Regression, it also identified fewer false positive transactions and more false negative transactions.

This is confirmed in the confusion matrices in Figures 2(a) and 2(b). XGBoost has 69 true positives compared to Logistic Regression's 72 true positives. To add to this, XGBoost also had only 1 false positive and 26 false negatives, while Logistic Regression had 10 false positives and 23 false negatives.

We then compared XGBoost and Logistic Regression to see if the differences were statistically significant. Due to overlapping folds, a regular paired t-test is invalid. Using a Nadeau-Bengio corrected t-test and a Holm's procedure at α=0.05, we found that only the difference in PR-AUC was statistically significant, with a difference of 0.1091, a p-value of 0.0022, and a 95% confidence interval of [0.0894, 0.1282]. While some of the other differences – like in F1 Score, Precision, and FPR – might seem statistically significant at first glance, these did not survive significance testing.

We used SHAP in the probability space to determine each feature's influence on XGBoost's decisions. The top 20 features, ordered by decreasing mean SHAP, is shown in Table 3. V14 had the strongest influence, with a mean SHAP value of 0.019819, with V12, V4, and V10 following at 0.008230, 0.007438, and 0.006822 respectively. Amount and Time, as shown back in Figure 1, have relatively little influence on XGBoost's decisions, with mean SHAP values of 0.001169 and 0.001009 respectively. All of this is also visually illustrated in the beeswarm plot in Figure 3. Lower values in V14, V12, and V10 typically indicate higher probabilities of fraud. Conversely, higher values in V4 also indicate higher fraud probabilities.

Figure 4 shows a SHAP dependence plot, compared V14 against its SHAP values. The color of each dot depends on the value of V7. A bluer dot indicates a more negative value of V7, while a redder dot indicates a more positive value. V14 values greater than -5 generally have SHAP values near 0, meaning they have minimal effect on the fraud probability. However, around -5, V14 SHAP values start to rapidly increase before flattening out into a "tail" towards the left (extremely negative V14 values). In this "tail", most values of V7 are negative (blue).

Methodological Note: Explanations were computed using SHAP (v0.52.0), with TreeExplainer configured for `model_output`='probability' and `feature_perturbation`='interventional'. The baseline expected value ($E[f(X)]$ = 0.005) was anchored against a background dataset of 256 instances, sampled uniformly

© The National High School Journal of Science 2026

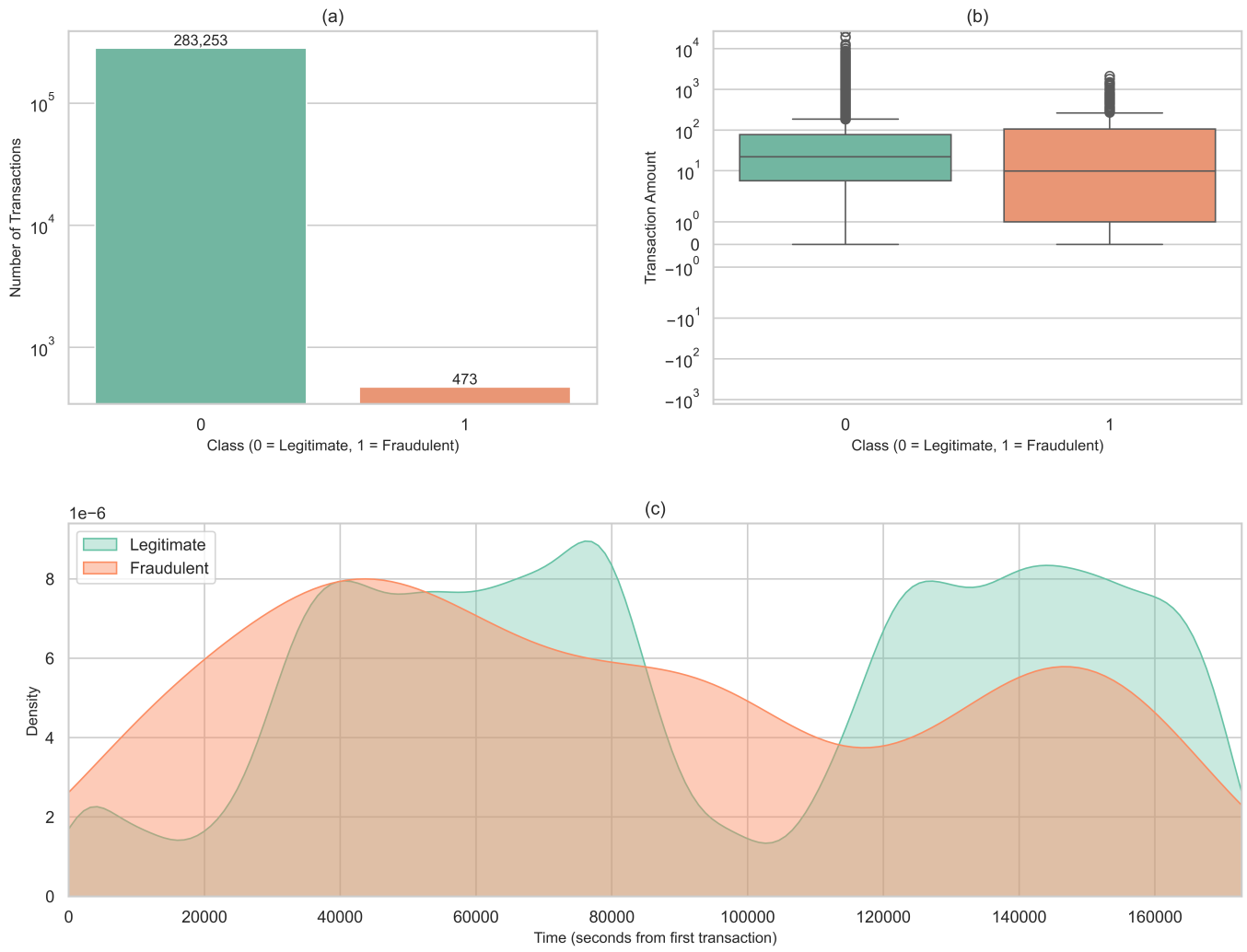

Figure. 1 Exploratory Data Analysis of the ULB Dataset. (a) The severe native class imbalance. (b) Transaction amount distributions (log scale). (c) Temporal density of legitimate versus fraudulent transactions over 48 hours.

Table 1 Predictive Performance of Baseline vs. Gradient-Boosted Architectures

Model	Evaluation	PR-AUC	F1 Score	Precision	Recall	FPR	Balanced Accuracy
Logistic Regression	Nested 5-Fold CV	0.7413 ± 0.0283	0.8072 ± 0.0427	0.8591 ± 0.0406	0.7671 ± 0.0860	0.000216 ± 0.000092	0.8834 ± 0.0430
Logistic Regression	Holdout	0.6929	0.8136	0.8780	0.7579	0.000177	0.8789
XGBoost	Nested 5-Fold CV	0.8504 ± 0.0332	0.8564 ± 0.0306	0.9225 ± 0.0300	0.8015 ± 0.0578	0.000115 ± 0.000050	0.9007 ± 0.0289
XGBoost	Holdout	0.8107	0.8364	0.9857	0.7263	0.000018	0.8631

Note: *At that locked threshold, holdout confusion counts were 69 true positives, 1 false positive, and 26 false negatives for XGBoost, versus 72, 10, and 23 for Logistic Regression. Holdout Brier scores were 0.00076 and 0.02383, respectively.*

without replacement from the original imbalance training to reflect operational fraud prevalence rather than the SMOTE-balanced training. Predictions reflect the final XGBoost architecture utilizing a locked classification threshold of 0.9773.

Figure 5 shows a waterfall plot for a true positive instance in the holdout set, starting from the baseline probability of $E[f(X)] = 0.005$ and ending at the final predicted probability of 1. V14 (-13.695) contributes the most to the model's final decision, contributing +0.34 to the final probability. This is

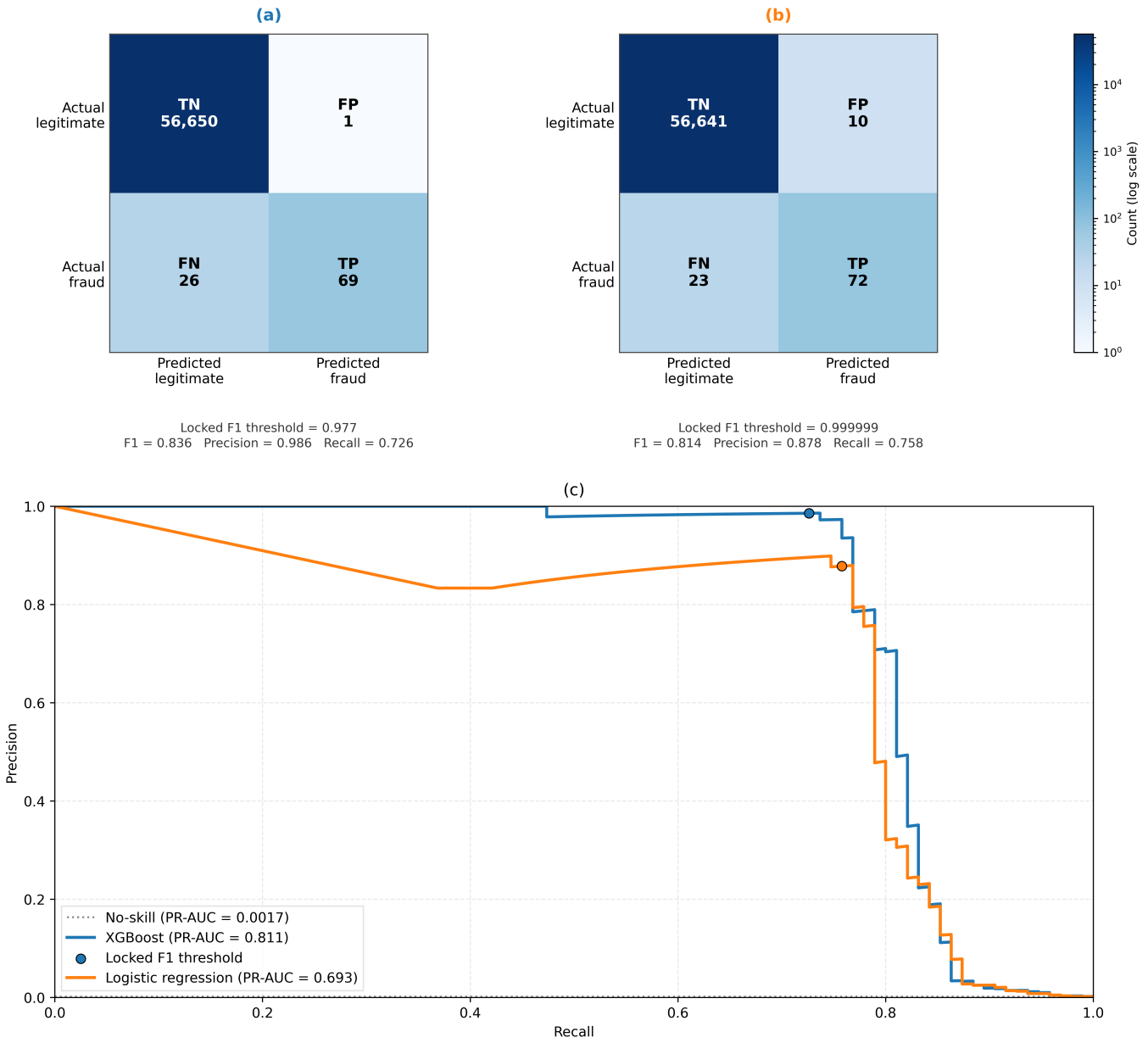

Figure. 2 Model Evaluation on the 20% Holdout Set. (a) and (b) Holdout confusion matrices detailing the absolute classifications counts for the XGBoost and Logistic Regression architectures at their respective locked F1-maximizing thresholds. (c) Precision-Recall curves demonstrating the threshold-free ranking superiority of the XGBoost model (PR-AUC = 0.811) over the baseline

followed by V12 (−13.064) with +0.16, V10 (−12.806) with +0.1, and V4 (8.925) with +0.1. This matches the dependence plot in Figure 4, where extremely low values of V14 were typically correlated with the highest SHAP values.

Figures 6 and 7 both show waterfall plots for instances where the model incorrectly identified a legitimate or fraudulent transaction. Figure 6 illustrates a false-positive instance while Figure 7 illustrates a false-negative instance. Both start with a baseline probability of $E[f(X)] = 0.005$. The false positive instance in Figure 6 ends with a final fraud probability of 1. Again, like the true positive instance in Figure 5, V14 (−7.11), V12 (−7.105), V10 (−5.622) most influenced the model's final decision, contributing +0.43, +0.16, and +0.14 to the final fraud probability. This false positive instance showed similar, extremely low V14, V12, and V10 values as the true positive instance in Figure 5, causing the model to incorrectly

© The National High School Journal of Science 2026

Table 2 Statistical Significance of XGBoost vs. Logistic Regression.

Metric	Mean Difference	t	p-score (raw)	Cohen's D	95% CI	Holm Threshold	Significant?
PR-AUC	0.1091	7.01	0.0022	4.70	[0.0894, 0.1282]	0.0083	Yes
Precision	0.0634	3.40	0.0272	2.28	[0.0472, 0.0883]	0.0100	No
F1 Score	0.0492	2.51	0.0662	1.68	[0.0267, 0.0718]	0.0125	No
FPR	−0.000101	−2.19	0.0936	−1.47	[−0.000163, −0.000062]	0.0167	No
Balanced Accuracy	0.0173	0.84	0.4468	0.57	[−0.0065, 0.0411]	0.0250	No
Recall	0.0345	0.84	0.4484	0.56	[−0.0132, 0.0821]	0.0500	No

Note: *Table sorted by raw p-value to reflect the Holm step-down procedure sequence. Statistical significance was evaluated using a Nadeau-Bengio corrected resampled t-test (df = 4). All cross-validation splits and subsequent statistical tests were executed using a fixed random seed of 42 to ensure exact reproducibility.*

Table 3 Global Feature Importance Rankings (Top 20)

Rank	Feature	Mean |SHAP| Value	Rank	Feature	Mean |SHAP| Value
1	V14	0.019819	**11**	V28	0.001368
2	V12	0.008230	**12**	V22	0.001234
3	V4	0.007438	**13**	V27	0.001171
4	V10	0.006822	**14**	Amount	0.001169
5	V3	0.005754	**15**	V6	0.001166
6	V11	0.003132	**16**	V2	0.001131
7	V17	0.002596	**17**	V5	0.001125
8	V7	0.002366	**18**	V9	0.001070
9	V8	0.002164	**19**	Time	0.001009
10	V1	0.002135	**20**	V18	0.000933

classify it as fraudulent.

On the contrary, the false-negative instance in Figure 7 ends with a final fraud probability of 0. V14 (0.984), V11 (−1.475), and V10 (2.689) contribute the most to bringing the final probability down to zero, although even they contribute very little. This false negative instance does not have an extremely low V14 value, so unlike Figures 5 and 6, the model classifies this as a legitimate transaction even though it is fraudulent.

Discussion

XGBoost had a statistically significant improvement in threshold-free ranking when compared to Logistic Regression. It achieved a nested-CV PR-AUC of 0.8504 ± 0.0332 and a holdout PR-AUC of 0.8107. Logistic Regression only achieved a nested-CV PR-AUC of 0.7413 ± 0.0283 and a holdout PR-AUC of 0.6929. This was the only difference that passed the Nadeau-Bengio corrected t-test and Holm adjustment. XGBoost ordered the rare fraudulent transactions more effectively than Logistic Regression, making it easier to correctly predict fraudulent cases when ranked by predicted fraud probability.

While PR-AUC passed significance testing, other metrics, which were evaluated on the fixed F1 threshold, did not. XGBoost achieved a precision of 0.986, a recall of 0.726, and an F1 score of 0.836. Logistic Regression, on the other hand, achieved a slightly lower precision and F1 score at 0.878 and 0.814, but also achieved a slightly higher recall at 0.758. XGBoost also had 69 true positives and 1 false positive vs. Logistic Regression's 72 true positives and 10 false positives. This means that while Logistic Regression identified 3 more fraudulent transactions than XGBoost, it also falsely identified 9 more legitimate transactions as fraudulent. However, the differences in these metrics did not pass Holm adjustment, so they cannot be considered statistically significant. While a bank may still choose to use XGBoost due to preferences, this was not confirmed by the differences in the threshold-dependent metrics.

A few PCA features – V14 (0.019819), V12 (0.008230), V4 (0.007438), and V10 (0.006822) – contributed most to the

© The National High School Journal of Science 2026

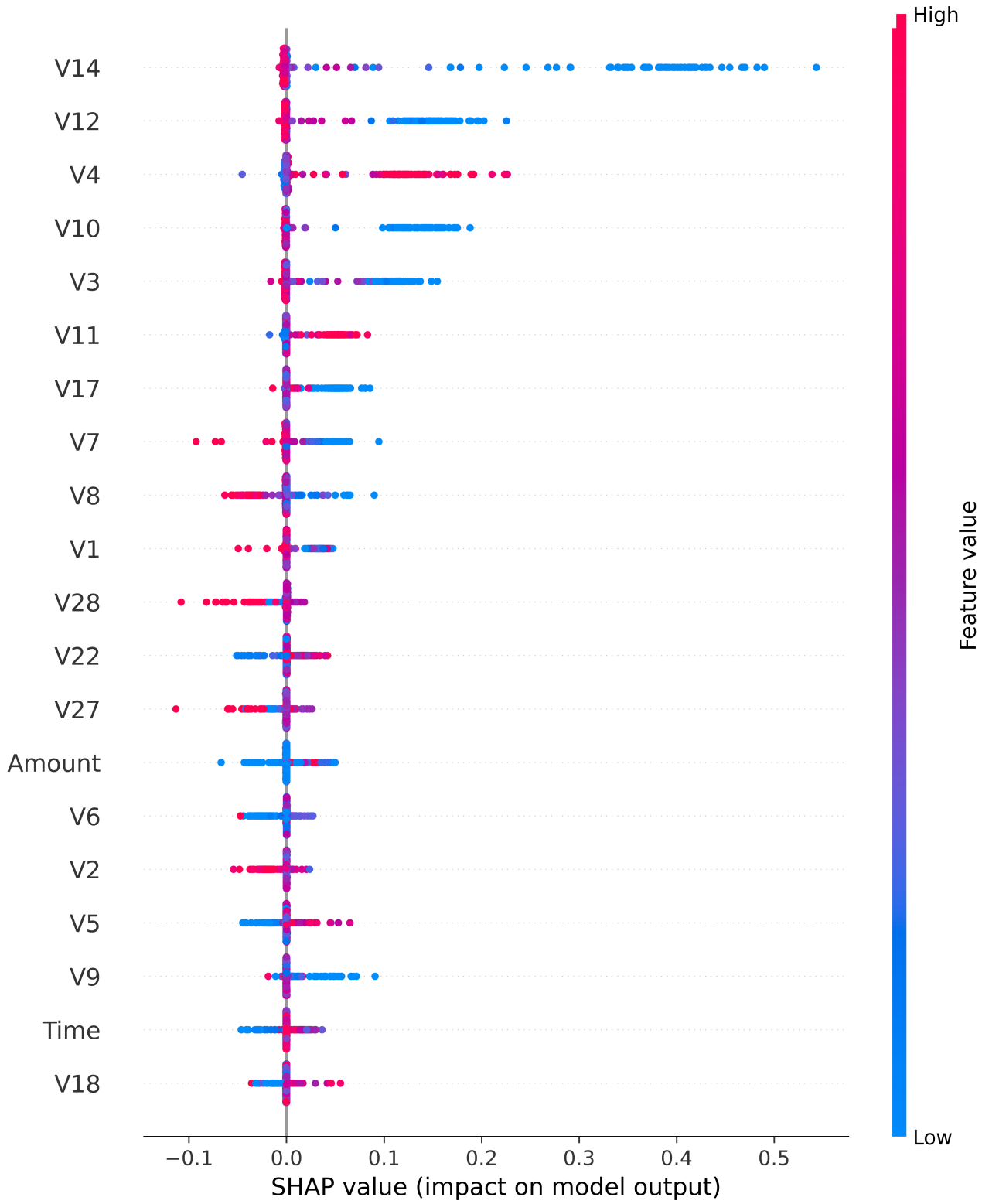

Figure. 3 Global Feature Analysis and Directional Impact via SHAP Summary Analysis for the XGBoost Architecture

© The National High School Journal of Science 2026

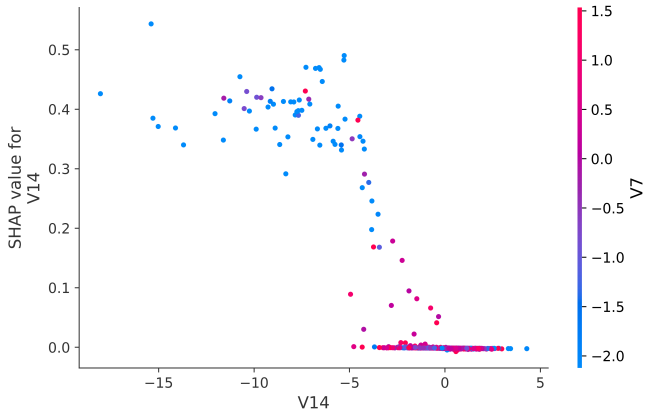

Figure. 4 SHAP Dependence Plot Illustrating the Non-Linear Predictive Threshold of Feature V14 and its Primary Interactive Effect

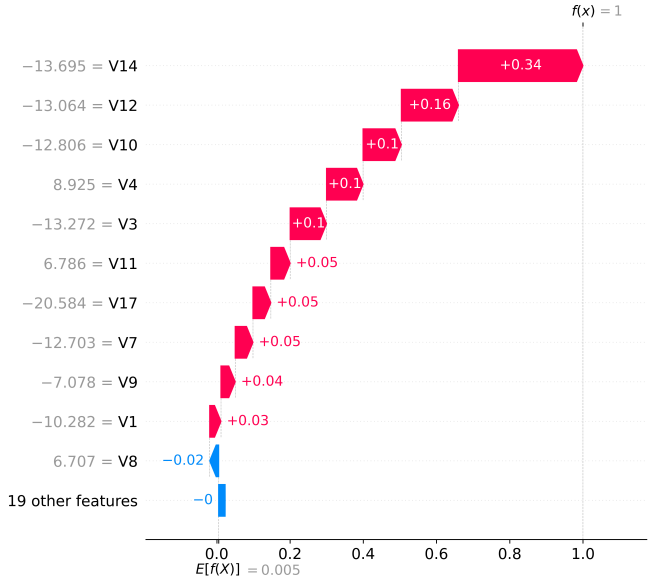

Figure. 5 SHAP Waterfall Plot Illustrate Local Feature Attribution and Step-by-step Probability Progression for a Single True Positive Fraud Instance.

model's final predicted fraud probability. However, the SHAP dependence plot in Figure 4 showed that V14 only truly mattered when it is extremely negative. V14 values greater than −5 contribute almost zero to the final fraud probability. Figures 5 and 6 show this pattern with extreme negative values of V14. Both the true positive instance in Figure 5 and the false positive instance in Figure 6 showed extremely low values of V14, causing the model to incorrectly label the legitimate transaction in Figure 6 as fraudulent because it shares similar patterns to the actually fraudulent one in Figure 5. On the contrary, the false negative instance in Figure 7 didnt have

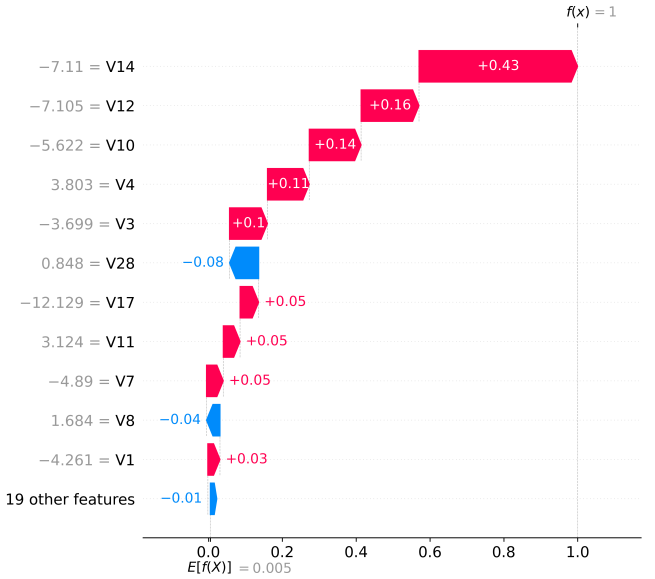

Figure. 6 SHAP waterfalls in probability space for a false positive on the holdout set.
NOTE: *Same explainer, background, and probability space as Figure 4.*

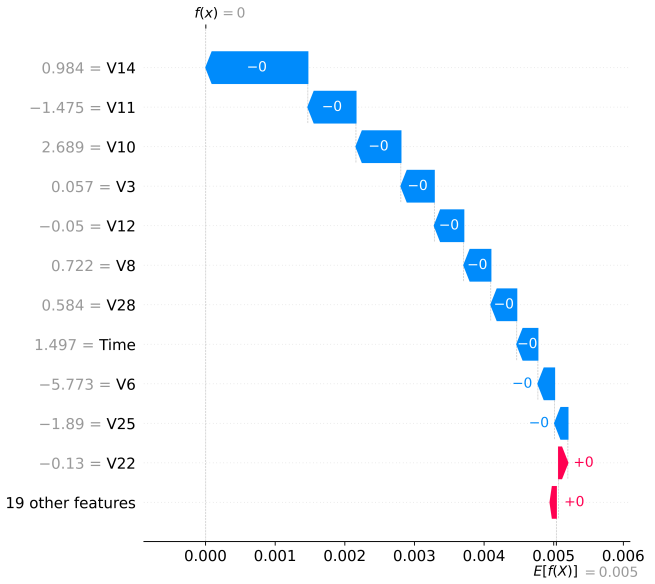

Figure. 7 SHAP waterfalls in probability space for a false negative on the holdout set.
NOTE: *Same explainer, background, and probability space as Figure 4.*

a low value of V14, so even though it was fraudulent, it classified it as legitimate.

However, there are some limitations that must be acknowl-

edged. The dataset was created in 2013, meaning it might not be able to fully represent modern credit card fraud. The dataset also only considered two days of transactions, so seasonal changes cannot be considered. 28 of the features (V1-V28) were also anonymized via PCA, meaning that we cannot accurately interpret each feature. Additionally, while SMOTE was used, it was manually chosen in advance, not picked using an ablation study, so there might be other techniques that could have fared better for this paper. To add to this, Figures 5 and 7 show the "loudest" true positive and the "quietest" false negative, not every scenario. Since there was also only 1 false positive for XGBoost, the SHAP waterfall plot in Figure 6 is the only false positive. In conclusion, XGBoost showed significant increases in PR-AUC, but improvements in F1, precision, recall, and FPR at a fixed threshold were not sufficiently supported.

Acknowledgements

I would like to thank Dr. Ganesh Mani (Carnegie Mellon University) for mentorship guidance during the development of this paper.

Data and Code Availability

The source code can be found in this GitHub Repository: https://github.com/MegaMonster799/creditcard-fraud-xai

References

1 M. T. Whitty, Card-not-present fraud: using crime scripts to inform crime prevention initiatives. Security Journal. 2022., https://doi.org/10.1057/s41284-022-00359-w.

2 S. Höppner, B. Baesens, W. Verbeke, T. Verdonck, Instance-dependent cost-sensitive learning for detecting transfer fraud. European Journal of Operational Research. Vol. 297, pg. 291–300, 2022., https://doi.org/10.1016/j.ejor.2021.05.028.

3 D. Almhaithawi, A. Jafar, M. Aljnidi, Example-dependent cost-sensitive credit cards fraud detection using SMOTE and Bayes minimum risk. SN Applied Sciences. Vol. 2, 2020., https://doi.org/10.1007/s42452-020-03375-w.

4 V. Ngoc Thanh Sang, Enhancing credit card fraud detection under severe class imbalance using cost-sensitive learning and threshold optimization. EAI Endorsed Transactions on Intelligent Systems and Machine Learning Applications. Vol. 3, 2026., https://doi.org/10.4108/eetisml a.12078.

5 M. A. Islam, M. A. Uddin, S. Aryal, G. Stea, An ensemble learning approach for anomaly detection in credit card data with imbalanced and overlapped classes. Journal of Information Security and Applications. Vol. 78, pg. 103618, 2023., https://doi.org/10.1016/j.jisa.2023.103618.

6 K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, A. K. Nandi, Credit card fraud detection using AdaBoost and majority voting. IEEE Access. Vol. 6, pg. 14277–14284, 2018., https://doi.org/10.1109/access.2018.2806420.

7 X. Zhao, Y. Liu, Q. Zhao, Improved LightGBM for extremely imbalanced data and application to credit card fraud detection. IEEE Access. Vol. 12, pg. 159316–159335, 2024., https://doi.org/10.1109/access.2024.3487212.

8 V. Suryanarayana, K. Maddileti, D. Satyanarayana, R. L. Jyothi, K. Sreekanth, P. Mande, R. N. Miriyala, O. Sudhakar, An efficient implementation of credit card fraud detection using CatBoost algorithm. Indonesian Journal of Electrical Engineering and Computer Science. Vol. 38, pg. 1914–1923, 2025., https://doi.org/10.11591/ijeecs.v38.i3.pp1914-1923.

9 F. K. Alarfaj, I. Malik, H. U. Khan, N. Almusallam, M. Ramzan, M. Ahmed, Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. IEEE Access. Vol. 10, pg. 39700–39715, 2022., https://doi.org/10.1109/ACCESS.2022.3166891.

10 N. Gamal, E. M. G. Younis, W. M. Makram, Enhancing credit card fraud detection with a hybrid approach using machine and deep learning. Scientific Reports. Vol. 16, pg. 10944, 2026., https://doi.org/10.1038/s41598-026-42891-4.

11 A. Al-Bulushi, A. K. Shaikh, N. Adhikari, Applying supervised machine learning algorithms and ensemble models to enhance credit card fraud detection. Frontiers in Artificial Intelligence. Vol. 9, 2026., https://doi.org/10.3389/frai.2026.1813728.

12 S. F. Farabi, M. Prabha, M. Alam, M. Z. Hossan, M. Arif, M. R. Islam, A. Uddin, M. Bhuiyan, M. Z. A. Biswas, Enhancing credit card fraud detection: A comprehensive study of machine learning algorithms and performance evaluation. Journal of Business and Management Studies. Vol. 6, pg. 252–259, 2024., https://doi.org/10.32996/jbms.2024.6.13.21.

13 P. Tiwari, S. Mehta, N. Sakhuja, J. Kumar, A. K. Singh, Credit card fraud detection using machine learning: A study. arXiv. 2021., https://doi.org/10.48550/arXiv.2108.10005.

14 R. Praba, G. Darshan, K. T. Roshanraj, B. Surya Prakash, Study on machine learning algorithms. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. Vol. 7, pg. 67–72, 2021., https://doi.org/10.32628/CSEIT2173105.

15 S. Motie, B. Raahemi, Financial fraud detection using graph neural networks: A systematic review. Expert Systems with Applications. Vol. 240, pg. 122156, 2023., https://doi.org/10.1016/j.eswa.2023.122156.

16 D. Cheng, Y. Zou, S. Xiang, C. Jiang, Graph neural networks for financial fraud detection: A review. Frontiers of Computer Science. Vol. 19, 2025., https://doi.org/10.1007/s11704-024-40474-y.

17 A. Cherif, H. Ammar, M. Kalkatawi, S. Alshehri, A. Imine, Encoder-decoder graph neural network for credit card fraud detection. Journal of King Saud University - Computer and Information Sciences. Vol. 36, pg. 102003, 2024., https://doi.org/10.1016/j.jksuci.2024.102003.

18 S. Wei, S. Lee, Internet fraud transaction detection based on temporal-aware heterogeneous graph oversampling and attention fusion network. PLOS One. Vol. 20, pg. e0337208, 2025., https://doi.org/10.1371/journal.pone.0337208.

19 D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, Y. Qi, A semi-supervised graph attentive network for financial fraud detection. IEEE Xplore. 2019., https://ieeexplore.ieee.org/document/8970829.

20 B. Wu, K. M. Chao, Y. Li, Heterogeneous graph neural networks for fraud detection and explanation in supply chain finance. Information Systems. Vol. 121, pg. 102335, 2024., https://doi.org/10.1016/j.is.2023.102335.

21 S. Ghosh, R. L. Anand, T. Bhowmik, S. Chandrashekhar, GoSage: Heterogeneous graph neural network using hierarchical attention for collusion fraud detection. 2023., https://doi.org/10.1145/3604237.3626856.

© The National High School Journal of Science 2026

22 I. Benchaji, S. Douzi, B. El Ouahidi, Credit card fraud detection model based on LSTM recurrent neural networks. Journal of Advances in Information Technology. Vol. 12, pg. 113–118, 2021., https://doi.org/10.12720/jait.12.2.113-118.

23 I. Benchaji, S. Douzi, B. El Ouahidi, J. Jaafari, Enhanced credit card fraud detection based on attention mechanism and LSTM deep model. Journal of Big Data. Vol. 8, 2021., https://doi.org/10.1186/s40537-021-00541-8.

24 K. EL Hachimi, G. Orhanou, L. Ballihi, Credit card fraud detection model based on optimized long short-term memory: Imbalanced class solution and overfitting reduction. Expert Systems with Applications. Vol. 305, pg. 130871, 2026., https://doi.org/10.1016/j.eswa.2025.130871.

25 C. Yu, Y. Xu, J. Cao, Y. Zhang, Y. Jin, M. Zhu, Credit card fraud detection using advanced transformer model. 2024 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom). pg. 343–350, 2024., https://doi.org/10.1109/metacom62920.2024.00064.

26 T. Deng, S. Bi, J. Xiao, Transformer-based financial fraud detection with cloud-optimized real-time streaming. pg. 702–707, 2024., https://doi.org/10.1145/3724154.3724271.

27 H. Fanai, H. Abbasimehr, A novel combined approach based on deep autoencoder and deep classifiers for credit card fraud detection. Expert Systems with Applications. Vol. 217, pg. 119562, 2023., https://doi.org/10.1016/j.eswa.2023.119562.

28 Q. Bao, J. Wang, H. Gong, Y. Zhang, X. Guo, H. Feng, A deep learning approach to anomaly detection in high-frequency trading data. 2025 4th International Symposium on Computer Applications and Information Technology (ISCAIT). pg. 287–291, 2025., https://doi.org/10.1109/iscait64916.2025.11010680.

29 C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence. Vol. 1, pg. 206–215, 2019., https://doi.org/10.1038/s42256-019-0048-x.

30 N. Bussmann, P. Giudici, D. Marinelli, J. Papenbrock, Explainable machine learning in credit risk management. Computational Economics. Vol. 57, 2020.

31 R. Bahlool, N. Hewahi, W. Elmedany, Performance, fairness, and explainability in AI-based credit scoring: A systematic literature review. Journal of Risk and Financial Management. Vol. 19, pg. 104, 2026., https://doi.org/10.3390/jrfm19020104.

32 A. Kumiega, R. Xu, Governing the tradeoff between predictive accuracy and adversarial robustness: An enterprise model risk framework. Risks. Vol. 14, pg. 184, 2026., https://doi.org/10.3390/risks14080184.

33 A. Dal Pozzolo, O. Caelen, R. A. Johnson, G. Bontempi, Calibrating probability with undersampling for unbalanced classification. 2015 IEEE Symposium Series on Computational Intelligence. pg. 159–166, 2015., https://doi.org/10.1109/SSCI.2015.33.

34 N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research. Vol. 16, pg. 321–357, 2002., https://doi.org/10.1613/jair.953.

35 S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems 30 (NeurIPS 2017), 2017.

© The National High School Journal of Science 2026

Appendix A

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
PR-AUC (XGBoost)	0.8265	0.8715	0.8037	0.8768	0.8733
PR-AUC (Logistic)	0.7508	0.7305	0.6969	0.7637	0.7644
PR-AUC (XGB − LR)	0.0757	0.1410	0.1068	0.1131	0.1089
F1 Score (XGBoost)	0.8421	0.8873	0.8116	0.8611	0.8800
F1-Score (Logistic)	0.8261	0.8029	0.7368	0.8205	0.8497
F1-Score (XGB − LR)	0.0160	0.0844	0.0748	0.0406	0.0303
Precision (XGBoost)	0.9655	0.9403	0.9032	0.9118	0.8919
Precision (Logistic)	0.9048	0.8871	0.8596	0.8000	0.8442
Precision (XGB − LR)	0.0608	0.0532	0.0436	0.1118	0.0477
Recall (XGBoost)	0.7467	0.8400	0.7368	0.8158	0.8684
Recall (Logistic)	0.7600	0.7333	0.6447	0.8421	0.8553
Recall (XGB − LR)	−0.0133	0.1067	0.0921	−0.0263	0.0132
FPR (XGBoost)	0.000044	0.000088	0.000132	0.000132	0.000177
FPR (Logistic)	0.000132	0.000154	0.000177	0.000353	0.000265
FPR (XGB − LR)	−0.000088	−0.000066	−0.000044	−0.000221	−0.000088
Balanced Acc. (XGBoost)	0.8733	0.9200	0.8684	0.9078	0.9341
Balanced Acc. (Logistic)	0.8799	0.8666	0.8223	0.9209	0.9275
Balanced Acc. (XGB − LR)	−0.0066	0.0534	0.0461	−0.0130	0.0066

© The National High School Journal of Science 2026