Ensemble Stacking for Imbalanced Fraud Detection: A Comparative Study of Machine Learning Approaches on Credit Card Transaction Data

Sanawer Batool

University of Central Punjab, Lahore, PK
L1F22BSDS0048@ucp.edu.pk

Mehwish Shafiq

University of Central Punjab, Lahore, PK
L1F22BSDS0042@ucp.edu.pk

January 2026

Abstract

Credit card fraud detection presents significant challenges due to extreme class imbalance and asymmetric misclassification costs. This study compares three individual machine learning algorithms (Logistic Regression, Random Forest, XGBoost) with an ensemble stacking approach on the Kaggle Credit Card Fraud Detection dataset containing 284,807 transactions with 0.17% fraud rate. We employed SMOTE oversampling to address class imbalance and evaluated models using recall, precision, and F1-score metrics appropriate for imbalanced classification. Results show XGBoost achieved the highest F1-score (85.00%), while the ensemble stacking model achieved second-highest recall (88.78%), successfully detecting 87 out of 98 fraudulent transactions. Cost-benefit analysis reveals Random Forest delivered the best ROI ($33,850) due to minimal false positives (15), while the ensemble occupied a strategic niche with high recall (88.78%) and acceptable precision (77.68%). Our findings suggest model selection should be context-dependent: risk-averse institutions should prioritize high-recall models (Logistic Regression or Ensemble), while customer-focused institutions benefit from high-precision models (Random Forest or XGBoost). This research contributes a decision framework for fraud detection model selection based on institutional risk tolerance and cost structures.

1 INTRODUCTION

Credit card fraud has become a pervasive problem in the digital economy, with global losses exceeding billions of dollars annually (Nilson Report, 2020). As e-commerce and contactless payment adoption accelerates, fraudsters employ increasingly sophisticated techniques to exploit financial systems, necessitating advanced automated detection mechanisms. Traditional rule-based fraud detection systems, while interpretable, struggle to adapt to evolving fraud patterns and generate high false positive rates that frustrate legitimate customers (Dal Pozzolo et al., 2015).

Machine learning offers a promising alternative through its ability to identify complex patterns in transaction data and adapt to new fraud strategies. However, fraud detection presents unique challenges for machine learning applications. First, fraudulent transactions typically represent less than 1% of all transactions, creating extreme class imbalance that causes models to bias toward the majority class (Chawla et al., 2002). Second, misclassification costs are highly asymmetric: a false negative (missed fraud) results in direct financial loss averaging $500 per transaction, while a false positive (blocked legitimate transaction) incurs only customer service costs of approximately $10 (Randhawa et al., 2018). Third, real-time detection requirements demand computationally efficient models that can process thousands of transactions per second.

Ensemble methods, which combine predictions from multiple base learners, have demonstrated superior performance across various machine learning domains (Zhou, 2012). Stacking, a specific ensemble technique that uses a meta-learner to combine base model predictions, theoretically leverages complementary strengths of diverse algorithms. For instance, combining a high-recall linear model with high-precision tree-based models could balance fraud detection effectiveness with customer experience. However, empirical evidence on whether ensemble stacking consistently outperforms well-tuned individual classifiers on highly imbalanced real-world fraud data remains limited.

This research addresses three key questions:

1. How do commonly used classification algorithms perform individually on highly imbalanced fraud data when combined with SMOTE oversampling?

2. Does ensemble stacking improve detection performance compared to individual models?

3. What are the precision-recall trade-offs and cost-benefit implications of different model choices? By answering these questions, this study contributes a practical decision framework for financial institutions to select fraud detection models based on their specific risk tolerance and operational constraints.

2 LITERATURE REVIEW

2.1 Machine Learning for Fraud Detection

Machine learning has been extensively applied to fraud detection with varying degrees of success. Early work by Ghosh and Reilly (1994) demonstrated that neural networks could detect fraudulent credit card transactions with reasonable accuracy. More recent studies have explored diverse algorithms: Support Vector Machines (SVM) (Bhattacharyya et al., 2011), Decision Trees (Prusti & Rath, 2019), and ensemble methods (Randhawa et al., 2018).

Alghobiri (2018) compared Naive Bayes, J48 Decision Trees, and k-Nearest Neighbors on ten datasets, finding that no single algorithm uniformly outperformed others across all data characteristics. This highlights the importance of algorithm selection based on dataset properties. For fraud detection specifically, Pranckevicius and Marcinkevicius (2017) compared five classifiers on text reviews and found that Random Forest and Logistic Regression achieved the best precision-recall balance, though their study focused on balanced datasets.

2.2 Handling Class Imbalance

Class imbalance is the most significant challenge in fraud detection, as fraudulent transactions typically represent 0.1-1% of total transactions (Dal Pozzolo et al., 2015). Three primary approaches address this problem:

Resampling techniques modify the training data distribution. Undersampling removes majority class samples but discards potentially valuable information (Drummond & Holte, 2003). Oversampling duplicates minority samples but risks overfitting. SMOTE (Synthetic Minority Over-sampling Technique), introduced by Chawla et al. (2002), generates synthetic minority samples by interpolating between existing minority instances and their nearest neighbors. Multiple studies have demonstrated SMOTE's effectiveness for fraud detection (Dal Pozzolo et al., 2015; Randhawa et al., 2018), though some research suggests it can create noisy synthetic samples in high-dimensional spaces (He et al., 2008).

Cost-sensitive learning assigns different misclassification costs to classes, penalizing false negatives more heavily than false positives. While theoretically appealing for fraud detection's asymmetric costs, practical implementation requires accurate cost estimates that vary across institutions (Elkan, 2001).

Algorithm-level approaches include class weight adjustment in tree-based models and threshold moving in probabilistic classifiers. These methods are simpler to implement but may be less effective than resampling for extreme imbalance (López et al., 2013).

2.3 Ensemble Methods

Ensemble learning combines multiple models to improve prediction accuracy and robustness. Three main categories exist: bagging (e.g., Random Forest), boosting (e.g., XGBoost), and stacking (Zhou, 2012).

Random Forest, introduced by Breiman (2001), builds multiple decision trees on bootstrapped samples and averages their predictions. Its effectiveness stems from variance reduction through tree diversity. For fraud detection, Random Forest has demonstrated strong performance due to its handling of non-linear relationships and resistance to overfitting (Prusti Rath, 2019).

XGBoost (Extreme Gradient Boosting), developed by Chen and Guestrin (2016), sequentially builds trees where each new tree corrects errors from previous trees. Its advanced regularization, handling of missing values, and built-in class imbalance parameters make it highly effective for fraud detection (Randhawa et al., 2018). Multiple Kaggle competitions have been won using XGBoost, establishing it as a benchmark algorithm.

Stacking, proposed by Wolpert (1992), uses a meta-learner to combine base model predictions. The meta-learner learns optimal weights for combining diverse models, theoretically leveraging their complementary strengths. Randhawa et al. (2018) applied stacking with AdaBoost and majority voting for fraud detection, achieving improvements over individual models. However, their study used balanced data, leaving questions about stacking's effectiveness on highly imbalanced real-world fraud data.

2.4 Evaluation Metrics for Imbalanced Classification

Traditional accuracy is inappropriate for imbalanced classification because a naive classifier predicting all instances as the majority class achieves high accuracy while failing to detect any fraud (Provost & Fawcett, 2001). Instead, researchers recommend:
- **Precision**: Proportion of fraud predictions that are correct (minimizes false alarms)
- **Recall (Sensitivity)**: Proportion of actual frauds detected (minimizes missed frauds)
- **F1-Score**: Harmonic mean of precision and recall, providing balanced assessment
- **Precision-Recall AUC**: Area under precision-recall curve, summarizing performance across thresholds

For fraud detection, recall is typically prioritized because missing a fraud (false negative) costs significantly more than a false alarm (false positive). However, excessive false positives degrade customer experience and increase operational costs, necessitating a balance (Dal Pozzolo et al., 2015).

2.5 Research Gap

While existing research has explored individual aspects of fraud detection, SMOTE for imbalance, ensemble methods for performance, and cost-benefit analysis for deployment, comprehensive comparisons of ensemble stacking versus state-of-the-art individual classifiers on real-world imbalanced credit card data remain limited. Specifically, most studies either use balanced datasets, synthetic data, or report only accuracy metrics. This research addresses this gap by:

- Systematically comparing three individual classifiers with ensemble stacking on real imbalanced fraud data (0.17% fraud rate)
- Employing appropriate evaluation metrics (recall, precision, F1-score) rather than accuracy
- Conducting cost-benefit analysis to provide actionable recommendations for practitioners
- Investigating when ensemble stacking adds value versus when individual models suffice

This comprehensive approach provides both theoretical insights into ensemble effectiveness on imbalanced data and practical guidance for fraud detection deployment.

3 METHODOLOGY

This section describes our experimental design, including dataset characteristics, preprocessing procedures, model configurations, and evaluation methodology.

3.1 Dataset

We utilized the Kaggle Credit Card Fraud Detection dataset, a widely-used benchmark for imbalanced classification research (Dal Pozzolo et al., 2018). The dataset contains 284,807 credit card transactions made by European cardholders in September 2013, over a 48-hour period. Among these, 492 transactions (0.172%) are fraudulent, representing extreme class imbalance typical of real-world fraud scenarios.

Features:

- **V1-V28**: 28 numerical features resulting from PCA transformation. Due to confidentiality, the original features and background information cannot be disclosed.
- **Time**: Seconds elapsed between each transaction and the first transaction in the dataset.
- **Amount**: Transaction amount in Euros, ranging from €0.00 to €25,691.16.
- **Class**: Binary response variable (0 = legitimate, 1 = fraud).

The PCA transformation was applied by the original data collectors for privacy protection, meaning features V1-V28 are already scaled and decorrelated. This preprocessing limits domain-specific feature engineering but ensures data confidentiality while maintaining predictive patterns.

3.2 Data Preprocessing

3.2.1 Feature Selection

We excluded the 'Time' feature from our analysis because it represents seconds elapsed since the first transaction in the dataset—an arbitrary reference point that does not generalize to future transactions. While temporal patterns (e.g., time-of-day, transaction velocity) could be valuable, they cannot be reliably extracted from this relative timestamp. The 'Amount' feature was retained as transaction value is a known fraud indicator (fraudsters often test with small amounts or target high-value transactions).

3.2.2 Feature Scaling

Features V1-V28, already PCA-transformed, have similar scales (approximately -20 to +20 with mean near 0). However, the 'Amount' feature exhibits high variance (mean: €88.35, std: €250.12, max: €25,691.16). To prevent this feature from dominating distance-based algorithms, we applied StandardScaler:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Critically, the scaler was fit only on training data and then applied to test data to prevent data leakage. Tree-based models (Random Forest, XGBoost) do not require feature scaling as they split on individual feature thresholds, but Logistic Regression requires it for convergence and optimal performance.

3.2.3 Train-Test Split

We performed an 80-20 stratified train-test split to maintain the original fraud ratio in both subsets:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

This yielded:

- Training set: 227,845 transactions (393 frauds, 0.172%)
- Test set: 56,962 transactions (99 frauds, 0.174%)

The stratified split ensures both sets have similar fraud rates, enabling fair evaluation.

3.2.4 Handling Class Imbalance: SMOTE

To address the extreme class imbalance (0.172% fraud), we applied SMOTE (Synthetic Minority Oversampling Technique) exclusively to the training set:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
```

SMOTE generated synthetic fraud samples by:

- For each fraud instance, identifying its K nearest fraud neighbors (K=5 by default)
- Randomly selecting one neighbor
- Creating a synthetic sample along the line segment between the instance and its neighbor
- Repeating until the fraud class matches the legitimate class count

This increased the training set from 227,845 to 454,904 samples (227,452 legitimate, 227,452 fraud), achieving a 1:1 class ratio. Critically, SMOTE was applied only to training data, never to test data. The test set remained in its original imbalanced state (0.174% fraud) to ensure evaluation reflects real-world performance. This approach prevents data leakage (synthetic samples not influencing test set) while enabling models to learn from balanced data during training.

3.3 Models and Hyperparameters

We implemented four model configurations: three individual classifiers and one ensemble stacking approach.

3.3.1 Logistic Regression

A linear model serving as our baseline:

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(
    max_iter=1000,      # Iterations for convergence
    random_state=42
)
```

After SMOTE balancing, we removed class_weight='balanced' as the training data was already balanced. We increased max_iter to 2000 to ensure convergence on scaled data. Logistic Regression was trained on scaled features (X_train_scaled) as it relies on gradient descent optimization, which requires feature scaling for stable convergence.

3.3.2 Random Forest

An ensemble of decision trees using bagging:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
            random_state=42
            )
rf.fit(X_train_scaled, y_train_balanced)
```

We increased min_samples_leaf from the default (2) to 4 to prevent overfitting to synthetic SMOTE samples. Random Forest was trained on unscaled features as tree-based models are invariant to feature scaling.

3.3.3 XGBoost

A gradient boosting model with advanced regularization:

```
from xgboost import XGBClassifier

xgb = XGBClassifier(
    random_state=42
)
```

After SMOTE balancing, scale_pos_weight was set to 1 (equal weighting) since training data had 1:1 class ratio. We added subsample and colsample_bytree parameters to prevent overfitting to synthetic samples through stochastic gradient boosting. XGBoost was trained on unscaled features.

3.3.4 Ensemble Stacking

A meta-learning approach combining all three base models:

```
base_models = [
('lr', LogisticRegression(max_iter=1000, random_state=42)),
('rf', RandomForestClassifier(n_estimators=100, max_depth=10,
min_samples_leaf=4, random_state=42)),
('xgb', XGBClassifier(scale_pos_weight=1, learning_rate=0.05,
n_estimators=100, subsample=0.8, random_state=42))
]

lr_model, rf_model, xgb_model = get_models()
lr_model.fit(X_train_scaled, y_train_balanced)
rf_model.fit(X_train_scaled, y_train_balanced)
xgb_model.fit(X_train_scaled, y_train_balanced)

lr_test = lr_model.predict_proba(X_test_scaled)
rf_test = rf_model.predict_proba(X_test_scaled)
xgb_test = xgb_model.predict_proba(X_test_scaled)

ensemble_test = (w1 * lr_test + w2 * rf_test + w3 * xgb_test)
ensemble_test_pred = np.argmax(ensemble_test, axis=1)
```

3.4 Evaluation Methodology

3.4.1 Metrics

Given the class imbalance and asymmetric costs, we evaluated models using:

Primary Metric: Recall (Sensitivity) Recall = TP / (TP + FN)

Measures the proportion of actual frauds successfully detected. This is our most critical metric because false negatives (missed frauds) result in direct financial loss.

Secondary Metric: F1-Score F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

Harmonic mean of precision and recall, providing balanced assessment. We calculated F1-score for the fraud class (minority class) only, using average='binary' in scikit-learn.

Supporting Metric: Precision Precision = TP / (TP + FP)

Measures the proportion of fraud predictions that are correct. Important for customer experience as false positives generate unnecessary transaction blocks and customer service calls.

Confusion Matrix Provides complete view of model behavior:

- True Positives (TP): Frauds correctly identified
- False Positives (FP): Legitimate transactions incorrectly flagged
- True Negatives (TN): Legitimate transactions correctly identified
- False Negatives (FN): Frauds that slipped through

Accuracy was explicitly NOT used as it is misleading for imbalanced classification. A naive model predicting all transactions as legitimate achieves 99.83% accuracy while detecting zero frauds.

3.4.2 Cost-Benefit Analysis

We conducted economic impact assessment using conservative cost estimates:

- Average fraud amount: $500 (industry estimate)
- Customer service cost per false alarm: $10 (phone call handling)

Net Benefit = (TP × $500) - (FN × $500) - (FP × $10)

This formula accounts for:

- Revenue saved from detected frauds (TP × $500)
- Revenue lost from missed frauds (FN × $500)
- Operational costs from false alarms (FP × $10)

3.4.3 Experimental Procedure

For each model:

- Train on SMOTE-balanced training data (227,452 frauds, 227,452 legitimate)
- Predict on original imbalanced test data (99 frauds, 56,863 legitimate)
- Generate confusion matrix
- Calculate precision, recall, F1-score
- Compute net benefit
- Compare performance across models

All experiments used random_state=42 for reproducibility. Code was implemented in Python 3.8 using scikit-learn

4 Results

his section presents our experimental findings, organized by model performance metrics, confusion matrix analysis, cost-benefit assessment, and comparative evaluation.

4.1 Overall Performance Comparison

Model	**Recall**	**F1-Score**	**Precision**	**TP**	**FN**	**FP**	**TN**
Logistic Regression	90.82%	14.25%	7.73%	89	9	1,062	55,802
Ensemble	**88.78%**	82.86%	77.68%	87	11	25	56,839
XGBoost	86.73%	**85.00%**	83.33%	85	13	17	56,847
Random Forest	84.69%	84.69%	84.69%	83	15	15	56,849

Table 1: Performance Metrics of Different Models

4.2 Logistic Regression

Logistic Regression achieved the highest recall (90.82%), successfully detecting 89 out of 98 frauds, missing only 9. However, this aggressive detection came at the cost of extremely low precision (7.73%), generating 1,062 false alarms—the highest among all models.

The resulting F1-score of 14.25% reflects this severe imbalance between precision and recall. While the model excels at fraud detection, the 1,062 false positives would generate significant customer friction and operational costs ($10,620 in customer service costs). This model represents an "aggressive" strategy suitable only for scenarios where missing frauds is absolutely unacceptable, regardless of customer experience impact.

4.3 Random Forest

Random Forest demonstrated remarkable balance with identical precision, recall, and F1-score (84.69%). The model correctly identified 83 out of 98 frauds while generating only 15 false alarms—the lowest among all models.

This exceptional precision (only 15 false positives out of 56,962 transactions) resulted in the highest net benefit ($33,850) despite not having the highest recall. The model's ability to minimize false alarms while maintaining strong fraud detection makes it ideal for customer-focused institutions where transaction friction must be minimized.

4.4 XGBoost

XGBoost achieved the highest F1-score (85.00%), representing the best overall balance between precision and recall. The model detected 85 out of 98 frauds (86.73% recall) with 83.33% precision, generating only 17 false alarms.

XGBoost's superior F1-score stems from its gradient boosting mechanism, which iteratively corrects errors from previous trees. With net benefit of $33,830, XGBoost provides excellent overall performance for general-purpose fraud detection deployment.

4.5 Ensemble Stacking

The ensemble stacking model achieved the second-highest recall (88.78%), detecting 87 out of 98 frauds, with moderate precision (77.68%), resulting in 25 false alarms.

The ensemble's F1-score (82.86%) was lower than both XGBoost (85.00%) and Random Forest (84.69%), challenging the common assumption that ensemble methods universally outperform individual models. However, the ensemble occupies a strategic niche: it achieves higher recall than tree-based models (88.78% vs. 84-87%) while maintaining significantly better precision than Logistic Regression (77.68% vs. 7.73%).

This positioning makes the ensemble suitable for institutions requiring ¿88% fraud detection (recall) while keeping false alarm rates under 0.05% (25 false positives out of 56,962 transactions). The net benefit of $33,750 demonstrates strong economic performance, ranking third among all models.

4.6 Summary of Findings

Our experiments reveal four key findings:

Finding 1: Ensemble stacking did not achieve highest F1-score. XGBoost individually achieved F1=85.00%, outperforming the ensemble (F1=82.86%) by 2.14 percentage points. This challenges the assumption that ensemble methods universally outperform well-tuned individual models.

Finding 2: Ensemble achieved second-highest recall (88.78%). The ensemble successfully balanced aggressive fraud detection with acceptable precision, positioning it between Logistic Regression's extreme recall (90.82%, but 7.73% precision) and tree-based models' balanced profiles (84-87% recall, 83-85% precision).

Finding 3: Random Forest delivered best ROI ($33,850) through exceptional precision (only 15 false positives). Minimal customer service costs offset slightly lower recall, demonstrating that false positive minimization can be economically optimal.

Finding 4: ROI differences are marginal (¡0.3%) among top three models, while recall differences are substantial (4.09 percentage points). This suggests model selection should prioritize institutional risk tolerance (acceptable recall threshold) rather than ROI maximization.

4.7 Answering Our Research Question

Does ensemble stacking improve detection performance compared to individual models?

Contrary to our hypothesis, ensemble stacking did not achieve the highest F1-score (82.86%), underperforming both XGBoost (85.00%) and Random Forest (84.69%). However, the ensemble achieved the second-highest recall (88.78%), surpassing all tree-based models by 2.05-4.09 percentage points.

This nuanced result challenges the common assumption that ”ensemble methods always win.” Our findings suggest that stacking's effectiveness depends on base model complementarity. In our case:

- **Positive effect**: The ensemble's meta-learner (Logistic Regression) successfully weighted the aggressive predictions from Random Forest and XGBoost higher, increasing recall from 84-87% to 88.78%.

- **Negative effect**: However, this increased recall came at the cost of precision (77.68% vs. 83-85% for tree models), resulting in lower F1-score. The meta-learner introduced additional variance by attempting to reconcile two already-strong, similar-performing base models.

The lesson: **Stacking adds value when base models have genuinely different strengths to combine.** Had we included Logistic Regression as a base model (high recall, low precision) alongside tree models (balanced), the ensemble might have achieved better F1-score by learning to trust LR for fraud detection and tree models for precision. However, we excluded LR due to scaling incompatibility (RF/XGBoost prefer unscaled data, LR requires scaled data), which limited the ensemble's potential.

5 REFERENCES

References

[1] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), 602–613.

[2] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

[4] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

[5] Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. *IEEE Symposium Series on Computational Intelligence*, 159–166.

[6] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3784–3797.

[7] Drummond, C., & Holte, R. C. (2003). C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. *Proceedings of the ICML Workshop on Learning from Imbalanced Datasets*, 1–8.

[8] Elkan, C. (2001). The foundations of cost-sensitive learning. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 973–978.

[9] Ghosh, S., & Reilly, D. L. (1994). Credit card fraud detection with a neural-network. *Proceedings of the 27th Hawaii International Conference on System Sciences*, 3, 621–630.

[10] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1322–1328.