Credit Card Fraud Detection Using Machine Learning

Jaafar Ben Khaled

CS100W

Dr. Gregory Tomlinson

12 May 2025

TABLE OF CONTENTS

1. Introduction & Research Questions

Credit card fraud represents a growing threat to financial systems, with attackers continuously developing advanced evasion techniques that outpace traditional rule-based detection methods(Hafez et al., 2025). In this project, I use machine learning to effectively detect fraudulent transactions in a large real-world dataset of nearly 285,000 credit card purchases(*Credit Card Fraud Detection*, 2018). The challenge lies not only in identifying complex fraud patterns but also in dealing with severe class imbalance fraudulent transactions represent only about 0.17% of the data. My analysis explores several key questions: How can machine learning effectively detect fraudulent transactions in a large real-world dataset? How do the "Amount" distributions differ between fraud and normal transactions? What is the best way to handle imbalanced fraud data? Why is accuracy misleading for imbalanced data, and what metrics better evaluate performance? And finally, which classification model performs best in terms of precision and recall for fraud detection?

2. Background Research

Detecting credit card fraud has long been a challenge due to the extreme rarity and evolving tactics of fraudulent transactions. In most real-world datasets, legitimate purchases exceed fraud by ratios of hundreds or even thousands to one. Standard machine learning models trained on such imbalanced data tend to predict nearly every transaction as normal, achieving deceptively high overall accuracy but missing almost all fraud cases. To overcome this, researchers apply data-level sampling methods that rebalance the class distribution before model training. Below, I describe three widely used sampling techniques in detail and explain our choice of SMOTE for this project (Saikat, 2025).

Random Oversampling duplicates existing minority-class instances until they match the size of the majority class. By simply copying rare fraud examples, the training set attains a balanced class ratio, which encourages models to focus on the minority class rather than defaulting to the majority. However, this approach can lead to overfitting because duplicated records do not introduce new information; the model may memorize these repeated points, resulting in poor generalization to unseen fraud patterns. Additionally, large-scale duplication can inflate training time and memory usage, especially when the imbalance ratio is extreme, as it is with fraud detection, where fewer than 0.2% of records belong to the fraud class (Chawla et al., 2002).

Random Undersampling addresses imbalance by removing randomly selected instances from the majority class until the two classes are evenly represented. This method reduces training set size and accelerates computational speed, making it appealing when resources are limited. However, it carries a significant risk of discarding informative normal transactions that capture legitimate spending variability. Removing these vital patterns can degrade model performance, introducing underfitting for genuine transactions, and potentially harming the model's ability to distinguish borderline fraud cases (Saikat, 2025).

SMOTE (Synthetic Minority Oversampling Technique) generates synthetic minority examples by interpolating between existing fraud observations and their k-nearest minority neighbors. For each selected fraud sample, SMOTE picks one or more neighbors and creates new points along the line segments that connect them in feature space. This process effectively enlarges the minority class with new, plausible examples that maintain the underlying data distribution without mere duplication. By enriching the diversity of minority instances, SMOTE

fosters the formation of more robust decision boundaries and reduces the chance of overfitting (Chawla et al., 2002).

Overall, while random oversampling and undersampling each have merits in simplicity and speed, they pose trade-offs in terms of overfitting or information loss. SMOTE provides a middle ground that augments minority-class variability and leads to better generalization, making it the preferred choice for detecting credit card fraud in highly imbalanced data. In my project, I configured SMOTE with k=5 nearest neighbors and targeted a 1:1 ratio between fraud and normal classes.

3. Technical Approach

My technical approach consists of several key stages: dataset loading and preprocessing, exploratory data analysis (EDA), handling class imbalance, model training and tuning, and model evaluation. Each stage is critical to ensure that our machine learning models can accurately detect fraud. The project was developed on Google Colab, leveraging Python libraries such as Pandas and NumPy for data manipulation, Matplotlib and Seaborn for visualization, Scikit-learn for preprocessing and model training, XGBoost for gradient-boosted trees, and Imbalanced-learn for SMOTE resampling. I used the publicly available Kaggle Credit Card Fraud dataset, which includes 284,807 transactions recorded by a European bank over a two-day period, with only 492 transactions (0.17%) labeled as fraudulent. The dataset provides 28 anonymized principal component analysis (PCA) features (V1–V28) derived from the original transaction attributes, a transaction amount, and a timestamp measuring seconds elapsed from an arbitrary reference. To prepare the data, I first removed duplicate entries and handled missing values. The Time feature showed no significant predictive value and was removed to streamline

the model. I then applied z-score normalization to the transaction amount to align its scale with the PCA components, reducing bias during model training. Finally, the data was split into a stratified 70% training set and 30% test set, preserving the original fraud-to-normal ratio in both subsets.

Since the dataset is highly imbalanced, my EDA began with a bar plot comparing fraud versus non-fraud transactions (Figure A1 in Appendix A), visually showing the rarity of fraud cases (0.17%). I then focused on the "Amount" feature, one of the few non-anonymized attributes, by plotting stacked histograms (Figure A2 in Appendix A) to compare transaction amounts across fraud and non-fraud cases. This revealed subtle patterns: fraudulent transactions often involved smaller amounts, likely to evade detection. The PCA features (V1–V28) were not explored in depth due to their anonymized nature and lack of direct interpretability. To address the extreme class imbalance, I implemented SMOTE (Synthetic Minority Oversampling Technique) with k=5 nearest neighbors, oversampling the minority class to a 1:1 ratio in the training set.

I started with Logistic Regression because it is a simple and interpretable model that serves as a good baseline to evaluate performance. Then, I selected Random Forest, which is a tree-based ensemble model known for its robustness to noise and irrelevant features, which is useful here because the dataset has 28 anonymized features (V1–V28) that may contain hidden or less meaningful information. Random Forest is also efficient on large tabular datasets like this. Next, I used XGBoost, a gradient boosting model that typically outperforms Random Forest in terms of accuracy and efficiency on structured data. XGBoost handles imbalances, irrelevant

features, and nonlinear patterns effectively, making it a top performer for many classification tasks.

To properly evaluate our models on this highly imbalanced fraud detection dataset, I employed specialized metrics that focus on the minority class rather than relying on misleading measures like overall accuracy. The F1-score served as our primary metric because it balances both precision (ensuring most flagged transactions are truly fraudulent) and recall (capturing as many actual fraud cases as possible). I complemented this with detailed confusion matrix analysis to examine the exact distribution of false negatives (dangerous missed fraud cases) versus false positives (costly false alarms). For XGBoost, my best-performing model, this revealed an excellent 80% recall rate, detecting 113 of 142 fraud cases, while maintaining 82% precision (Figure A5 in Appendix A). This comprehensive evaluation approach ensured we didn't just create a model that appeared accurate by always predicting "non-fraud," but one that actually identified real fraudulent transactions with meaningful reliability, striking the optimal balance between catching fraud and minimizing business disruption from false positives. The model's strong 0.81 F1-score demonstrated its superior ability to handle this imbalance compared to alternatives like logistic regression (7% recall) or random forest (~50% recall) (Figure A6 in Appendix A).

4. Broader Applications

The methods I develop for fraud detection extend easily to other areas where rare events hide in large datasets. In insurance, companies can use the same resampling techniques and models to spot fraudulent claims among millions of legitimate filings, speeding up payouts and cutting losses. In cybersecurity, analysts can apply our workflow to detect one malicious login or

network intrusion within massive traffic logs, improving breach response. Healthcare organizations might adopt these approaches to find unusual billing or prescription patterns that indicate fraud or abuse, supporting compliance efforts. Even in manufacturing and IoT settings, teams can use anomaly detection to predict rare equipment failures from sensor data, avoiding downtime and costly repairs. By mastering steps like data balancing, model tuning for high recall and precision, and clear evaluation metrics, students build skills that translate directly to real-world jobs in data science, analytics, and engineering (Blagus & Lusa, 2013).

5. Team members

I am working on this project alone as a student. I will handle all tasks including data cleaning, exploratory analysis, applying SMOTE for balancing, model training, and evaluation.

6. Timeline & Deliverables

The project schedule aligns with the CS 100W Week 12 timeline. On April 9, I submitted the finalized research topic and a detailed outline on Canvas by 11:59 pm. By April 14, I uploaded the faculty advising report with input from my advisor. On April 16, at the start of class, I turned in a half-draft that included the Introduction, Background Research, and Technical Approach sections. A short elevator pitch slide deck and script were submitted to Canvas by 5 pm on April 21. I presented the project in class on May 5 during Week 15, sharing the key insights and results in a 5-minute summary. I will submit the polished project proposal on Canvas by May 12. By the end of May, I plan to publish the full project, including the code, visuals, and documentation, on GitHub to make it accessible and shareable.

7. Possible Roadblocks & Remedies

One of the biggest challenges I expected and later confirmed was the issue of class imbalance. This is the first time I have worked on a highly imbalanced dataset, and I was curious to see how it would affect the outcome. As I progressed through the project, I started noticing unexpected results, especially during model evaluation. At first, I assumed that the dataset from Kaggle had already been cleaned or preprocessed, which seemed to explain why my early results looked unusually strong, particularly when reviewing the correlation matrices. After spending a lot of time reviewing and debugging, I discovered that the real issue was in my code: I had mistakenly applied SMOTE after splitting the dataset, which meant that the sampling affected both the training and the test sets. This was a major error, as it led to testing the models on artificially balanced data, an unrealistic scenario in real-world applications. Once I realized this, I corrected the pipeline by splitting the data first, then applying SMOTE only to the training set while keeping the test set untouched. This change gave me more realistic evaluation metrics and made the model's performance much more reflective of what would happen with real, imbalanced data. This experience taught me how critical it is to handle resampling properly and double-check every step in the machine learning workflow. In future projects, especially those involving imbalanced datasets, I will be more careful about when and where I apply data transformation.

8. Appendix A : Code & Visualizations

Full code & Notebooks : https://github.com/JaafarBK02/CreditCard-Fraud-Detection

Figure A1: Bar plot showing the class imbalance between fraud and non-fraud transactions.

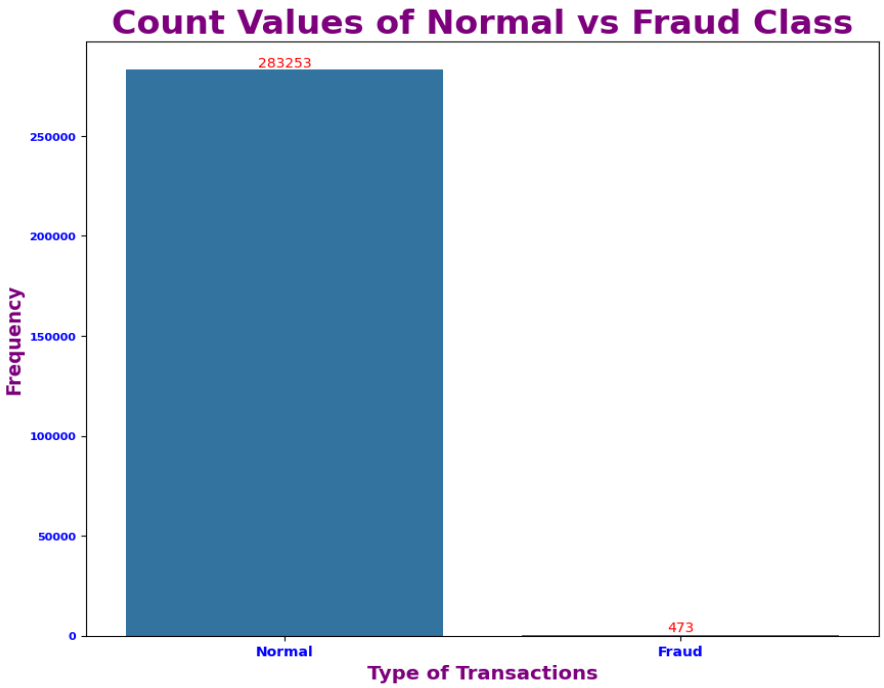


Figure A2: Stacked histograms showing the distribution of transaction amounts by class.

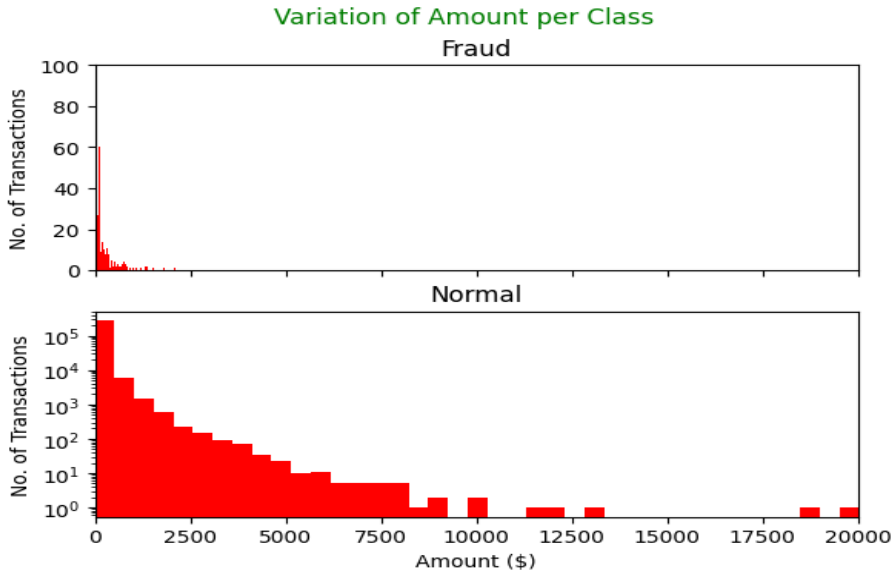


Figure A3: Confusion matrix for Logistic Regression.

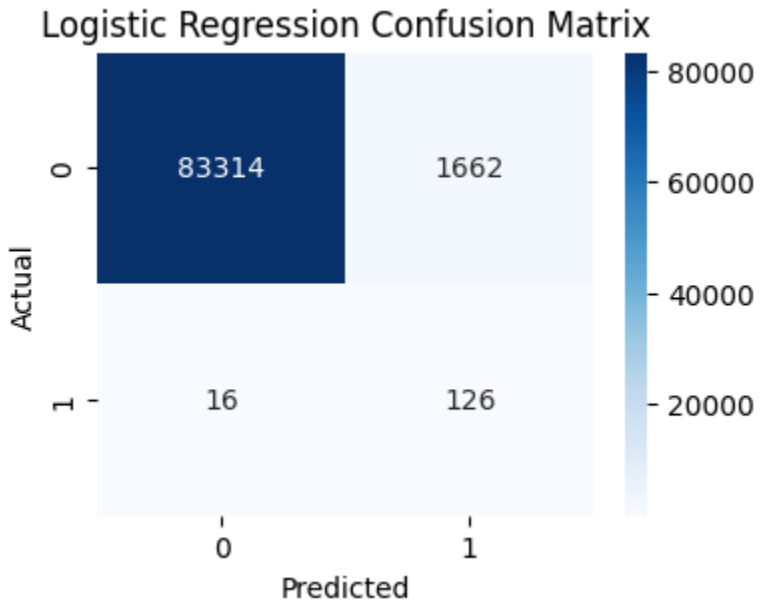


Figure A4: Confusion matrix for Random Forest.

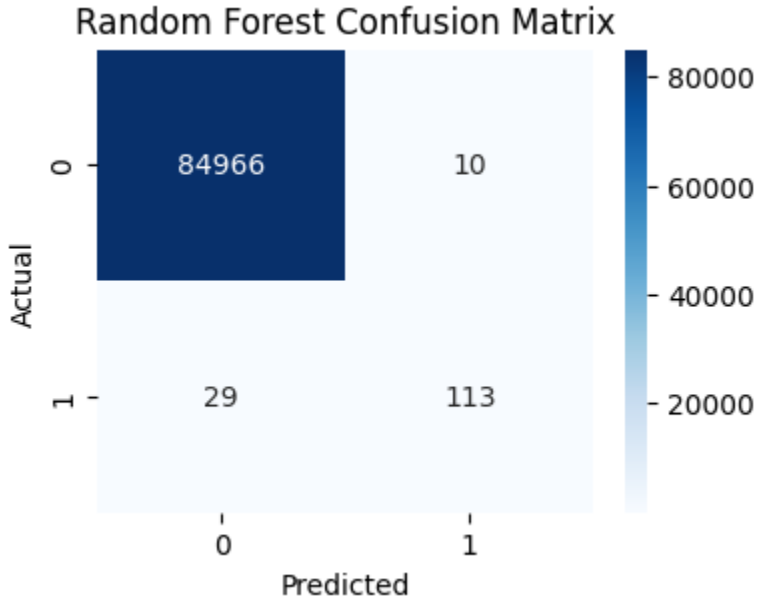


Figure A5: Confusion matrix for XGBoost.

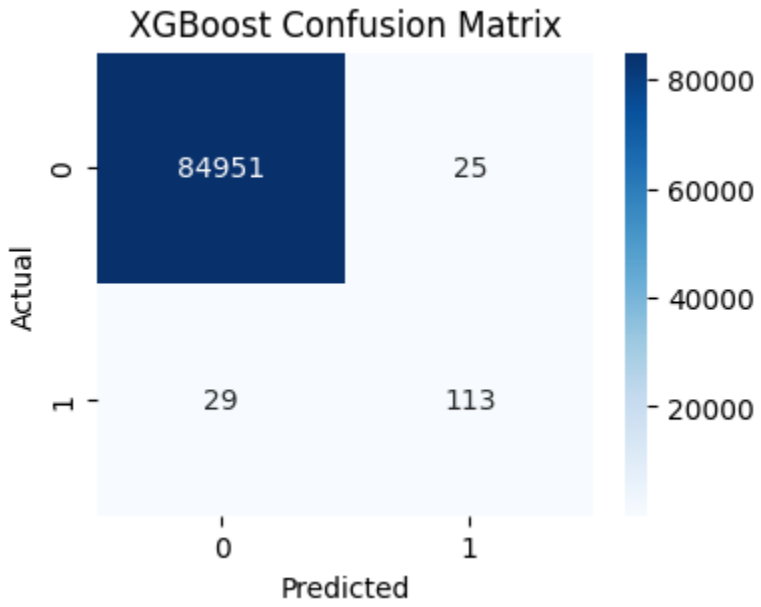
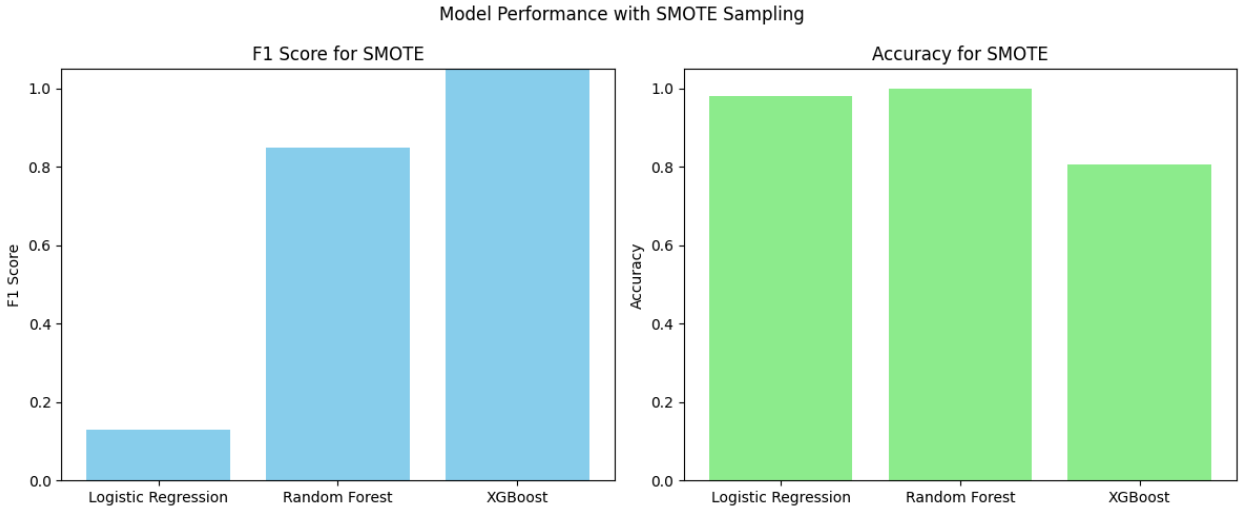


Figure A6: F1-Score & Accuracy comparison Bar plot



9. References

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. https://doi.org/10.1613/jair.953

Saikat. (2025, April 4). 5 Techniques to handle imbalanced data for a classification problem. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/

Blagus, R., & Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics, 14(1). https://doi.org/10.1186/1471-2105-14-106

Hafez, I. Y., Hafez, A. Y., Saleh, A., El-Mageed, A. a. A., & Abohany, A. A. (2025). A systematic review of AI-enhanced techniques in credit card fraud detection. Journal of Big Data, 12(1). https://doi.org/10.1186/s40537-024-01048-8

Credit card fraud Detection. (2018, March 23). Kaggle. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud