Fraud Detection in Financial Transactions Using Machine Learning Models

Darell Isaac Sam

Produced in Professor Hongfei Xue's Spring 2025 ITCS 3156.

Introduction

The primary objective of this project is to explore and analyze the use of machine learning methods used for fraud detection by applying them to a large real-world dataset. As usage of online and digital transactions continue to grow, the demand for accurate fraud detection has never been higher. Financial fraud causes consumers financial losses and has an unhealthy effect on confidence in digital infrastructures, thereby underscoring the need for advancing the technologies of fraud detection.

For this project I built and compared two different machine learning models for predicting whether a transaction is fraudulent or legitimate. The complete machine learning pipeline is developed including data exploration, preprocessing, model training, evaluation, and analysis of results. This work not only showcases the practical application of machine learning algorithms but also provides insights into the challenges and strategies involved in handling highly imbalanced datasets, which is a common issue in fraud detection tasks. My code and additional materials are publicly accessible on GitHub at [https://github.com/darellsam/Fraud-Detection-with-Machine-Learning.git].

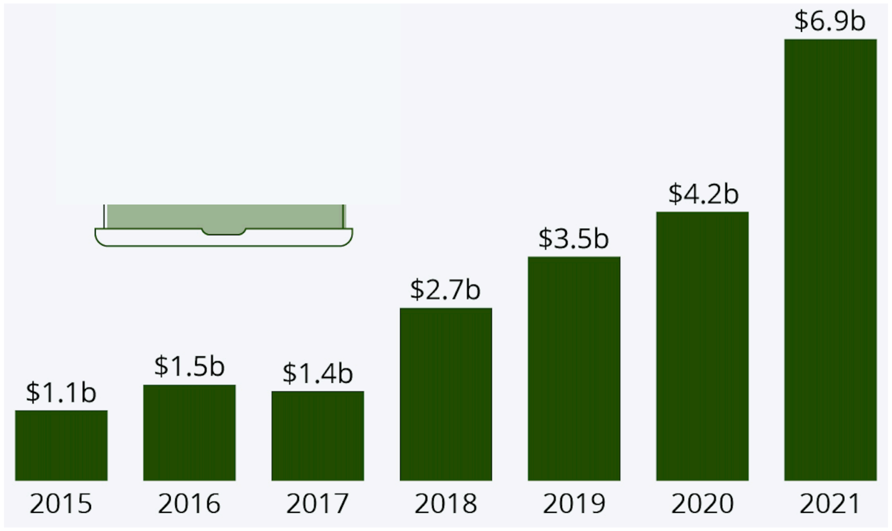

Figure 1: This graph from statista.com shows financial losses suffered by victims of internet crimes reported to the FBI from 2015 to 2021.

Data

The dataset used in training and testing the fraud detection models consists of approximately 6.3 million financial transaction records, each labeled to indicate whether the transaction was fraudulent (isFraud) or not. Each row represents a single transaction and includes a variety of features. The step column refers to the passage of time during the transaction. The type feature tells us about the nature of the transaction and can be one of the following: CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER. The amount column specifies the exact amount of the transaction. The nameOrig and nameDest fields represent the anonymized ID's of the sender and receiver accounts respectively. oldbalanceOrg and newbalanceOrig capture the originating account's balance before and after the transaction, while oldbalanceDest and newbalanceDest do the same for the destination account. The isFraud column is the target variable, where a value of 1 indicates a fraudulent transaction and 0 indicates a legitimate one.

Initial Insights and Visualizations

Out of over 6.3 million transactions in the dataset, only 8,213 were labeled as fraudulent (0.121%). This extreme imbalance really shows the challenge of fraud detection and the need for models that can accurately identify events that are statistically significant. To gain insight into the correlation between fraud and different types of transactions, two visual analyses were performed.

The first visualization shows the percentage of fraudulent transactions by transaction type, revealing the categories in which we see a disproportionate amount of fraud. From the graph of our first visualization, we see that TRANSFER, and CASH_OUT exhibit relatively high rates of fraud in comparison to other types such as PAYMENT, DEBIT, and CASH_IN, which have no instances of fraud. This suggests that those who intend on taking advantage of people through digital transactions primarily exploit the type of transactions that allow money to be moved quickly and directly.

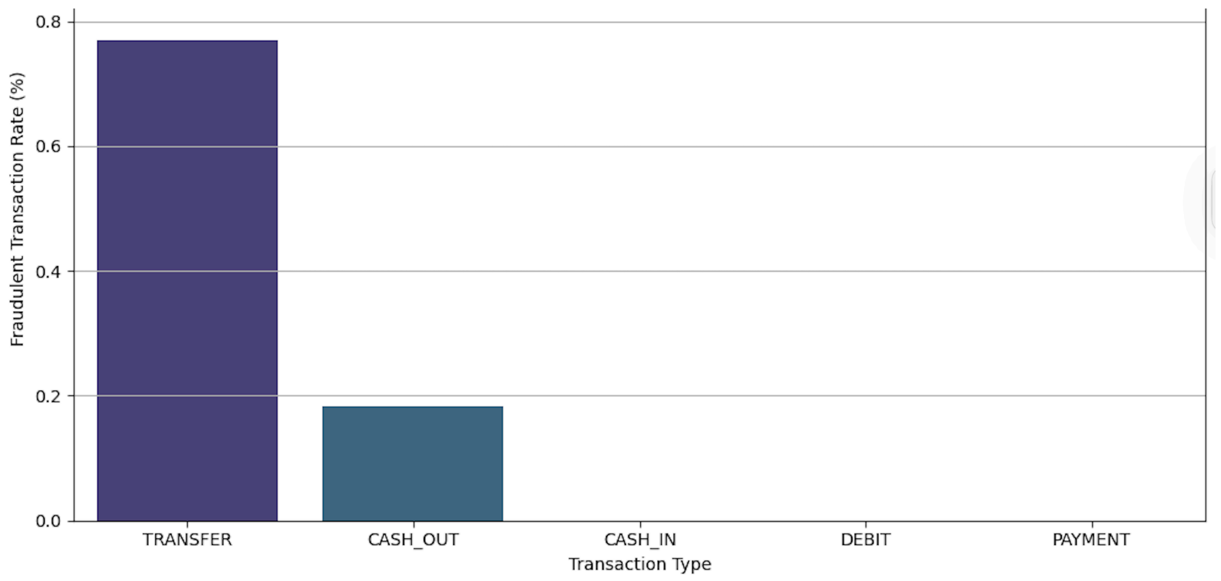

Figure 2: Visualization #1 showing percentage of fraudulent transactions by type.

The second visualization separates non-fraudulent and fraudulent transactions counts into two side-by-side bar plots. While PAYMENT dominates overall transaction volume in non fraudulent cases (in the millions), the fraudulent transactions are far fewer and heavily skewed toward TRANSFER and CASH_OUT. These plots collectively highlight that while some transaction types are more common, others are far more likely to be fraudulent—an important distinction that can guide feature engineering and model emphasis later in the pipeline.

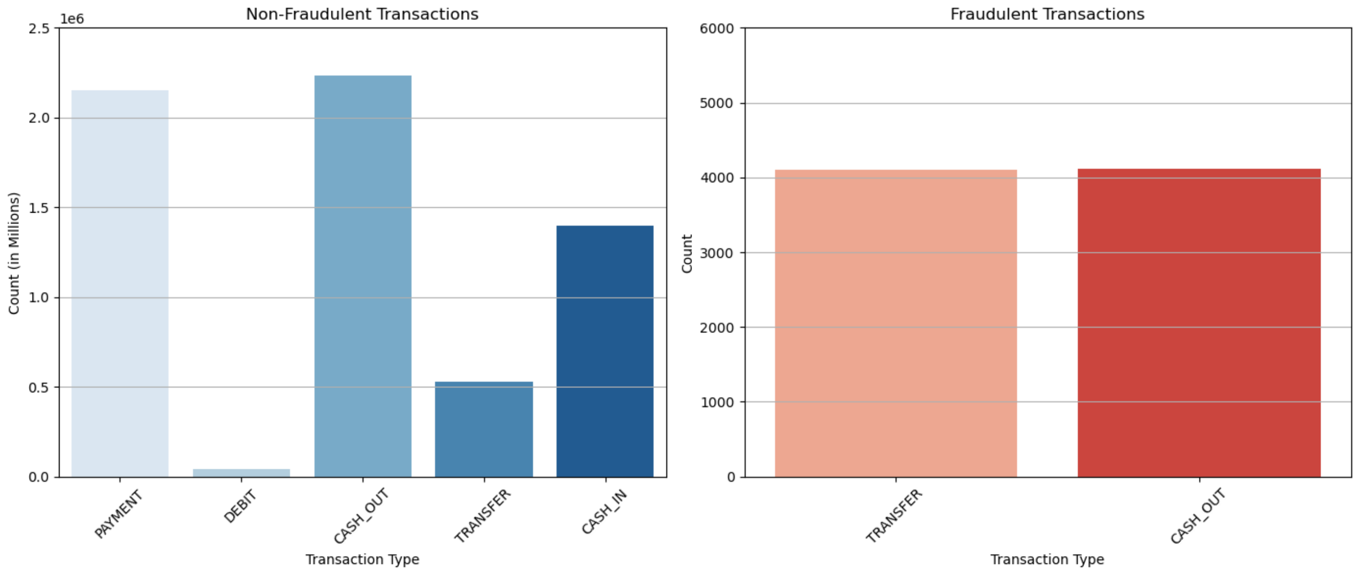

Figure 3: Visualization #2 comparing volume of non-fraud cases and fraudulent cases for each transaction type.

Data Preprocessing and Preparation

Before applying machine learning models, several preprocessing steps were conducted to prepare the dataset. First, two high-cardinality identifier columns, nameOrig and nameDest, were dropped since they contain unique values for nearly every transaction and do not provide meaningful patterns for prediction—especially for distance-based models like k-Nearest Neighbors (kNN). Next, the categorical type feature, which represents the type of transactions as encoded into numerical format using LabelEncoder. This allows the model to interpret categorical values effectively. Then the dataset was split into features (X) and the target variable (y), which tells us whether a transaction is fraudulent or not. Because kNN is sensitive to feature scales, StandardScaler was used to normalize all input features to have a mean of 0 and standard deviation of 1. Without this normalization the model may think that differences in a feature like amount, which may have a range of 1 to 1,000,000, are far more important than step, which ranges from 1 to 743. Finally the data was split into training and testing sets using an 80-20 stratified[1] split to maintain class distribution, which is particularly important given the heavy imbalance of the dataset.

[1] It is important to stratify because, as mentioned before, less than 0.13% of the data is actually labeled as a fraudulent transaction. Without stratification a test set may have almost no fraud cases, if any, or you might end up with skewed ratio leading to biased training.

Methods

In approaching the problem of fraud detection, I chose to implement two fundamentally different machine learning algorithms: k-Nearest Neighbors (kNN) and Logistic Regression. These models were selected to explore both instance-based learning and linear probabilistic classification.

kNN is a non-parametric algorithm that classifies a data point based on the majority label among its nearest neighbors in the feature space. The core idea is that similar transactions (based on numerical features like amount and account balances) will likely share the same class. I aimed to see how well this distance-based approach could identify fraudulent patterns without making strong assumptions about data distribution.

Logistic Regression, in contrast, models the probability of a transaction being fraudulent as a logistic function of its input features. Its interpretability and efficiency makes it a good baseline classifier. It also handles binary classification tasks like fraud detection well when class imbalance is addressed (which I accounted for by using class weights).

Results

Model Implementation Strategy

The dataset used in this study contains over 6.3 million financial transactions, out of which only 8,213 are labeled as fraudulent, accounting for approximately 0.13% of the total. Due to this extreme imbalance, traditional accuracy as a performance metric can be misleading. To combat this, models were evaluated using precision, recall, F1-score, and confusion matrices, with special attention given to recall for the fraud class (i.e., correctly identifying frauds). The dataset was preprocessed to normalize features and encode transaction types.

Logistic Regression was configured with class_weight=‘balanced’ to give more importance to the minority class during training. class_weight=‘balanced’ automatically adjusts the weights assigned for each class inversely proportional to their frequency in the data. Specifically the weight for class *i* is calculated as:

$$weight_i = \frac{n_{\text{samples}}}{n_{\text{classes}} \times n_i}$$

On the other hand, kNN was initialized with n_neighbors=5 after a few other tested k values; this is easy to tune later as well. 5 was a good and balanced value for k because a k too small, like 1 or 2, could lead the model to overfit while the contrary, suppose, a k of 20, would cause the model to underfit. Also, too high of a k would make the training and testing time far too long because of how many distance calculations need to be done, especially because the dataset has over 6.3 million rows.

Model Performance Analysis

Both models achieved overall accuracy, but their effectiveness at fraud detection varies greatly. The kNN model had an overall accuracy of 99.95%, predicting most non-fraud cases correctly and even detected 1,068 out of 1,643 frauds. Despite this, it still missed 575 fraud cases (false negatives), showing that even though kNN is highly accurate overall, it struggles with detecting fraud because of the datasets imbalance and kNN's bias towards the majority class. Its fraud class recall of 0.65 and precision of 0.95 indicate that while the frauds it does catch are usually correct, it misses many.

	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	0.95	0.65	0.77

kNN Accuracy: 0.9995

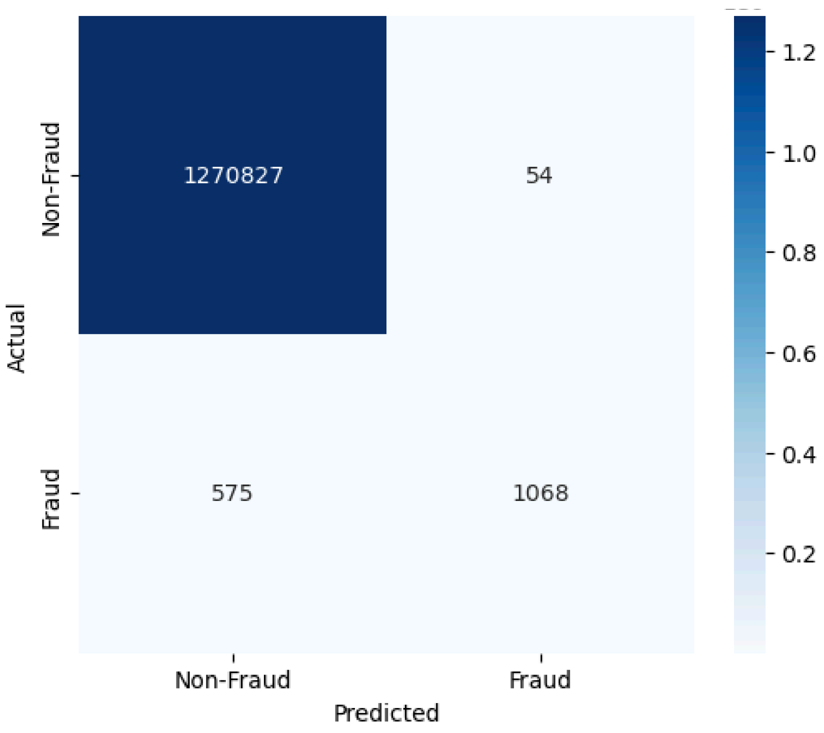

Figure 4: kNN confusion matrix.

In contrast, the Logistical Regression model had a lower overall accuracy, but had a much better recall for fraud. It detected 1,474 out of 1,643 fraudulent transactions, which meant that it had a recall of 0.90 for the fraud class. Despite this, there was a downside to this high recall and that was low precision, as it misclassified 38,527 non-fraud transactions as frauds (false positives). Regardless, its strong ability to catch most fraudulent transactions makes it a lot more useful in situations where catching fraud is critical, even if it means investigating more false alarms.

	Precision	Recall	F1-Score
0	1.00	0.97	0.98
1	0.04	0.90	0.07

Log Reg Accuracy: 0.9696

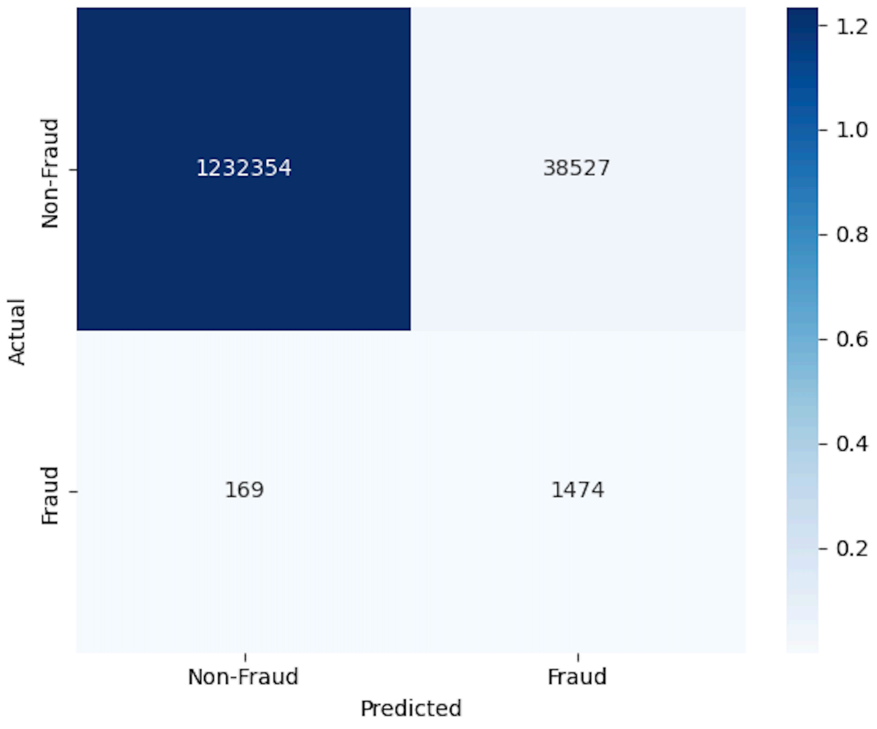

Figure 5: Log Reg confusion matrix.

Comparing the Models

Metric	kNN	Logistic Regression
Accuracy	99.95%	96.96%
Precision (Fraud)	0.95	0.04
Recall (Fraud)	0.65	0.90
F1-Score (Fraud)	0.77	0.07
False Positives	54	38527
False Negatives	575	169

Model	Accuracy	Precision (Fraud)	Recall (Fraud)	F1 Score (Fraud)
k-Nearest Neighbors	0.9995	0.95	0.65	0.77
Logistic Regression	0.9696	0.04	0.9	0.07

Conclusion

This study investigated fraud detection through the application of two supervised machine learning algorithms: k-Nearest Neighbors (kNN) and Logistic Regression, utilizing a significantly imbalanced dataset of financial transactions. The key finding is that relying solely on accuracy can be deceptive when addressing infrequent yet impactful categories such as fraud. While kNN produced nearly perfect accuracy, it failed to identify a substantial portion of fraud cases. Logistic Regression, on the other hand, performed far better in terms of fraud detection due to the use of class weighting, even though its accuracy was slightly lower and precision was much lower.

This work also emphasized the importance of careful data preprocessing, including feature scaling, encoding, and removing irrelevant information. It highlighted the need to choose relevant evaluation metrics that serve the need of the scenario which, in this case, is prioritizing recall over accuracy because of how large of a negative effect fraud can have on victims of it. Addressing class imbalance was very important to this work and measures such as weighting and sampling were implemented to reduce its impact. The key takeaways include a deeper understanding of how imbalanced data skews model behavior, the trade-offs between precision and recall, and the value of using relevant evaluation criteria.

Looking forward, this work can be expanded by testing more robust machine learning algorithms like Neural Networks. Additionally, cost-sensitive learning could directly penalize misclassified fraud cases, and targeted feature engineering could enhance detection capabilities by extracting more relevant signals. Ultimately, this project demonstrated not only how to apply machine learning in fraud detection but also how to critically evaluate and improve models for real-world use.

Works Cited and Acknowledgements

"Infographic: Americans Are Losing Billions due to Internet Crime." *Statista Infographics*, www.statista.com/chart/20845/financial-losses-suffered-by-victims-of-internet-crimes/.

IBM. "What Is the K-Nearest Neighbors (KNN) Algorithm?" *Ibm.com*, 4 Oct. 2021, www.ibm.com/think/topics/knn.

IBM. "Logistic Regression." *Ibm.com*, 16 Aug. 2021, www.ibm.com/think/topics/logistic-regression.

* AI was used to generate ideas for model enhancement, preprocessing strategies, and evaluation approaches. All analysis, implementation, and interpretation of results were conducted by the author.