CREDIT CARD FRAUD DETECTION

47999, 39465, 37682, 43729
London School of Economics and Political Science

Abstract

As digital payments increase, so does the risk of credit card fraud. Detecting such fraud is challenging due to extreme class imbalance and constantly evolving attack patterns. This project explores machine learning approaches for automated fraud detection using a real-world dataset, addressing class imbalance through synthetic sampling methods like SMOTE and bootstrap augmentation.

We compare supervised and unsupervised models, evaluating their performance using metrics tailored for imbalanced classification, such as precision, recall, and AUC. To test scalability, we train models on a larger synthesized dataset, simulating high-volume financial environments.

Our results show that supervised models perform well with sufficient labelled data, while unsupervised methods offer value in data-scarce scenarios. The study highlights the importance of combining modelling strategies, scalable processing, and robust evaluation to build effective fraud detection systems in real-world applications.

1 Introduction

1.1 Background and Motivation

In an increasingly digitized financial ecosystem, credit cards have become a standard method of payment across both physical and digital platforms. While this widespread adoption provides significant convenience, it has also made the financial system more vulnerable to fraudulent activity. Credit card fraud is a growing concern for consumers, banks, and regulatory bodies alike, with global losses projected to exceed $40 billion annually in the coming years. These fraudulent activities not only result in substantial financial losses, but also erode consumer trust, imposing reputational and operational costs on financial institutions.

Fraud detection is inherently complex because fraudsters continuously evolve their tactics. The nature of transaction data, high-dimensional, fast-streaming, and heavily imbalanced makes credit card fraud detection a quintessential big data challenge. Fraudulent transactions often represent a very small percentage (less than 0.5%) of the entire dataset, making it difficult for traditional models to detect them without a high rate of false positives or overfitting.

ST446 Project,
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.2 Objective of the Project

This project aims to develop an intelligent, adaptive, and scalable fraud detection system that leverages big data analytics and machine learning. We seek to address the challenges posed by high-dimensional, imbalanced transaction data through the use of distributed computing frameworks, such as Apache Spark, and advanced machine learning models.

Our goal is to build a fraud detection system that can process large volumes of transaction data efficiently and detect fraudulent behavior with a high degree of accuracy while minimizing false positives. By applying machine learning techniques such as logistic regression, random forests, gradient-boosted trees, and feature engineering methods like PCA, we aim to improve the accuracy and scalability of fraud detection systems.

Through this research, we hope to contribute to the broader financial ecosystem by providing an effective and reliable tool for fraud detection, thus enhancing consumer security and preserving the integrity of financial institutions.

2 Problem Description

As digital payment systems become integral to everyday transactions, the challenge of securing these platforms against credit card fraud has intensified, posing serious risks to both users and financial institutions. However, this transformation has also introduced substantial security risks, particularly in the form of credit card fraud. Credit card fraud occurs when unauthorized transactions are carried out using a legitimate cardholder's details, leading to considerable financial loss and reputational damage to financial institutions.

Recent studies indicate that financial institutions globally incur billions in annual losses due to fraudulent activities. These attacks have become more sophisticated, with fraudsters constantly evolving their tactics to bypass traditional rule-based and heuristic fraud detection systems. Manual verification approaches are both time-consuming and inadequate at scale, highlighting the need for intelligent, automated solutions capable of detecting fraudulent behavior swiftly and accurately.

A core challenge in credit card fraud detection is the **severe class imbalance** inherent in transaction datasets. Typically, fraudulent transactions represent less than 1% of all transactions, making it difficult for conventional classification models to learn useful patterns. This imbalance can lead to high accuracy but poor recall, where the model correctly predicts most non-fraud transactions but fails to identify actual fraud, a costly oversight in critical financial applications.

Furthermore, fraud detection systems must not only be *accurate*, but also *scalable* and *interpretable*. They must handle millions of transactions daily with minimal latency and be transparent enough to justify flagged alerts to stakeholders. Therefore, building fraud detection pipelines that balance predictive performance, computational efficiency, and interpretability remains an urgent and complex task.

This project aims to address these challenges by leveraging advanced machine learning methods, such as Isolation Forest, alongside data preprocessing strategies like SMOTE (Synthetic Minority Over-sampling Technique) and bootstrapped augmentation. The modeling process is conducted on the Google Cloud Platform (GCP), ensuring scalability, computational power, and efficiency to ensure our solutions remain viable as data volume increases.

This study seeks to develop a modular, end-to-end pipeline for credit card fraud detection that not only achieves high performance across key metrics like recall and AUC-ROC but also adapts to constraints of imbalanced data, computational load, and operational scalability. Ultimately, the solution aims to provide financial institutions with an intelligent, automated, and rapid fraud detection system, which is increasingly vital, yet difficult to achieve at scale.

3 Related Work

Credit card fraud detection remains a critical challenge for financial institutions due to the rapid increase in digital transactions and the risk posed by fraudulent activities. A key challenge in fraud detection is handling highly imbalanced datasets, where fraudulent transactions constitute a very small percentage of the total data set.

Obimbo et al. [6] propose using SMOTE oversampling to combat the imbalance in credit card fraud detection datasets. They tested 3 classifiers, LightGBM, Decision Trees and Random Forests on both the original dataset and SMOTE oversampled version of it and found that while there was no significant change in accuracy, SMOTE oversampling resulted in a better precision and F-score values on all three classifiers. Out of these LightGBM was found to perform the best.

Nayeem et al. [5] proposed a real-time fraud detection system (RTCCFD) that uses Apache Spark and machine learning algorithms. Their framework integrates Spark Streaming with Kafka to process high-volume transaction data. This approach is particularly relevant to our project, as it demonstrates how distributed computing can effectively handle fraud detection, which we aim to replicate using the Google Cloud Platform (GCP).

In a similar vein, Wahid and Zaini [7] developed a framework using PySpark for credit card fraud detection, addressing challenges such as class imbalance. Their work demonstrates how big data frameworks improve the scalability of fraud detection systems. Our project aligns with this approach, where we will use PySpark to implement models such as Isolation Forest and XGBoost for fraud detection.

Among the unsupervised options Wahid and Zaini highlight, Isolation Forest stands out as a lightweight yet effective anomaly detector. First introduced by Liu et al.[4], Isolation Forest (IF) isolates anomalies through random partitioning, giving it near-linear time complexity that scales well to high-volume data. These results make IF a practical unsupervised baseline for credit-card fraud detection.

Dal Pozzolo et al. [2] explored the impact of class imbalance on fraud detection performance. They recommended using precision-recall metrics instead of accuracy for model evaluation due to the extreme class imbalance in fraud detection tasks. This insight influences our project, where we will prioritize precision, recall, and AUC-PR in model evaluation.

Chawla et al. [1] introduced the Synthetic Minority Over-sampling Technique (SMOTE) to combat the effects of class imbalance. Their method, which generates synthetic samples for the minority class, is highly relevant to our approach, as we plan to apply SMOTE to the Kaggle credit card fraud dataset to enhance the training of our models.

4 Dataset Description and Preparation

The dataset used in this project is a widely recognized benchmark for credit card fraud detection, containing anonymized transactional records from European cardholders over two consecutive days in September 2013. It comprises a total of 284,807 transactions, of which only 492 (~0.17%) are labeled as fraudulent. Each transaction is described by 28 anonymized principal component features (V1–V28) generated via PCA for confidentiality, along with two untransformed features: *Time* and *Amount*. The target label `Class` indicates fraudulent transactions (1) and legitimate ones (0).

4.1 Data Splitting and Initial Preprocessing

To avoid information leakage and ensure representative evaluation, the dataset was split into training and testing subsets using a stratified sampling strategy. Approximately 80% of the transactions were used for training and 20% for testing, with the fraud class proportion preserved in both sets.

Basic preprocessing included:

- Dropping irrelevant or leakage-prone features (such as *Time*, depending on model requirements).
- Handling missing or infinite values (although the original dataset was already clean).

4.2 Addressing Class Imbalance: SMOTE and Bootstrapping

Given the highly imbalanced nature of the original dataset, where fraudulent transactions constituted less than 0.2% of all records, class balancing was a critical step in the pipeline. We applied a two pronged approach for synthetic data generation:

SMOTE (Synthetic Minority Over-sampling Technique): To artificially balance the training data, we used SMOTE to generate synthetic examples of the minority class by interpolating between existing fraud samples and their k-nearest neighbors in feature space. This enriched the minority class while avoiding simple duplication.

Bootstrapping with Jitter: To enhance model robustness and simulate real-world variability, we further augmented the training set by bootstrapping sampling with replacement and introducing controlled random noise ("jitter") to selected numerical features such as *Amount*. This helped increase diversity within both majority and minority classes and reduced overfitting.

The result was a significantly expanded training dataset improving the model's ability to learn fraudulent patterns effectively. This final dataset was used across all supervised models for a fair comparison of their learning performance.

Evaluating SMOTE Impact: While SMOTE is commonly used for class imbalance, its effectiveness in fraud detection is debated due to the risk of generating unrealistic samples. To assess its impact, we compared model performance with and without SMOTE.

We observed an improvement in AUC after applying SMOTE, indicating enhanced model confidence in ranking fraud versus non-fraud cases. Additionally, a t-SNE visualization showed that the synthetic fraud points were well distributed and sparsely intermixed with the majority class, without excessive overlap. In general, our visual and metric validation supports the use of SMOTE in this context, showing that it contributed positively to model discrimination without degrading class separation.

5 Methodology

Given the highly imbalanced nature of credit card fraud detection, where legitimate transactions vastly outnumber fraudulent ones, our methodology was designed to ensure both robustness and fairness across all stages of model development and evaluation. This section outlines the reasoning behind our experimental design, including pre-processing strategies, model selection, and evaluation criteria.

We began by conducting exploratory data analysis in order to understand what was needed to enhance the model stability. Due to the confidential nature of credit card transactions, our dataset had already been preprocessed using Principal Component analysis (a statistical method for dimensionality reduction) and transformed into 28 features. We split the transactions forming the training pool, and the reserved as an untouched test set. The following steps were performed strictly on training data to avoid information leakage.

Because fraudulent transactions represent only 0.17% of the data, we generate new minority class data points to help balance the training set. First, SMOTE synthesises new fraud samples by interpolating within the minority neighbourhood; next, bootstrap-with-jitter resamples those rows and adds small Gaussian noise (scaled by standard deviation of the feature to ensure the distribution does not change) to numeric fields, increasing variety without duplicating records. All augmentation is confined to the training subset. The entire preprocessing pipeline runs on Apache Spark DataFrames distributed across a multiple-node Google Cloud Dataproc cluster, providing the computing environment needed to incrementally scale to the resulting four-million-row training set. This is discussed more in our scalability analysis section.

Recognizing that fraud detection is inherently a rare event classification problem, we incorporated both supervised and unsupervised learning approaches in our pipeline. In the supervised realm, Logistic Regression offers an interpretable baseline, and LightGBM captures nonlinear patterns efficiently on tabular data and parallelise well in Spark. XGBoost was also tested on small subsets, and while it showed strong predictive performance, limitations in PySpark integration restricted its use at scale. We also employed Isolation Forest, which supplies unsupervised anomaly scores that remain useful when labels are sparse or delayed. This dual strategy enables us to assess performance across different learning paradigms and better understand their respective strengths under varying assumptions of data availability.

We paid particular attention to the evaluation framework, opting for metrics that are more informative under class imbalance. Class-imbalance demands metrics that reward correct fraud detection without being swamped by the majority class. Instead of relying solely on accuracy, which can be misleading in skewed datasets, we focused on metrics such as Precision, Recall, F1-Score, ROC-AUC, and PR-AUC. These metrics provide a more nuanced understanding of how well the models are identifying true frauds while minimizing false alarms.

Overall, our methodology reflects a deliberate balance between empirical rigor and practical applicability. By systematically comparing models under controlled conditions and using evaluation metrics that are suited to the task, we aim to generate insights that are not only theoretically sound, but also actionable in real world fraud detection systems.

6 Experimental Models and Evaluation

To evaluate the performance of various machine learning approaches for fraud detection, we implemented and evaluated machine learning models: **Logistic Regression**, **Isolation Forest**, **Light GBM** and **Random forest**. These models were selected to reflect both supervised and unsupervised learning applications, and for their established effectiveness in handling class imbalance a critical feature of fraud detection tasks. In this section, we present model evaluation results based on predictive performance. Scalability metrics, including execution time and resource utilization, are discussed separately in the following section, Scalability Analysis. The models were trained on multiple dataset sizes (5%, 25%, and 100%) to assess scalability and performance stability. We report results which come from the models trained on the full dataset using 2 workers. (When run with 3 workers, there was no significant change to the values of the metric.) All reported metrics are evaluated on the original, untouched test split, ensuring an unbiased measure of generalization. (Figure 2 contains the results run with both 2 workers and 3 workers)

6.1 Logistic Regression

Logistic Regression is a linear classifier commonly used for binary classification. It models the log-odds of the probability of the positive class using a logistic function. While simple, it is frequently used as a baseline in fraud detection tasks due to its interpretability, robustness, and computational efficiency, especially in large-scale environments.

Table 1: Logistic Regression Performance

Metric	Score
Accuracy	0.9992
Precision	0.8571
Recall	0.5806
F1-Score	0.6923
ROC-AUC	0.9766
PR-AUC	0.7095

Logistic Regression achieved a high accuracy of 99.92% . With a precision of 0.8571 and recall of 0.5806, there was a trade-off between catching fraudulent transactions and limiting false positives. However, in the context of severe class imbalance, accuracy can be misleading as it may remain high even when fraudulent transactions are missed entirely. Similarly, ROC-AUC (0.9766) can overstate performance when the non-fraud class dominates. Therefore, we place greater emphasis on the PR-AUC (0.7095), which better captures model effectiveness in retrieving rare positive cases without being skewed by the majority class. Finally, the F1-score of 0.6923 while not high supports its overall effectiveness. Notably, Logistic Regression also offered efficient training times and scalability (discussed more in the next section), making it suitable for deployment in production environments with low latency requirements.

Logistic Regression also demonstrated minimal training time and low resource usage, indicating its usefulness where interpretability and computational efficiency are essential.

6.2 Isolation Forest

The **Isolation Forest** an unsupervised anomaly detection model, was evaluated on the full training set. Despite its scalability, the model struggled to capture meaningful fraud patterns. It achieved an extremely low recall of 0.0108 and a precision of just 0.0476, with an F1-score of 0.0175. Its ROC-AUC was close to 0.50, suggesting performance no better than random guessing. The PR-AUC of only 0.0021 further reflects its ineffectiveness on highly imbalanced data.

Table 2: Isolation Forest Model Performance

Metric	Score
Accuracy	0.998
Precision	0.0476
Recall	0.0108
F1-Score	0.0175
ROC-AUC	0.5052
PR-AUC	0.0021

While the Isolation Forest achieves a high overall accuracy of 99.8%, this metric is misleading due to class imbalance. More informative metrics such as precision and recall reveal performance challenges. A precision of 0.0476 means that less than 5% of predicted frauds were correct, and a recall of 0.0108 shows that the model detected just 1% of actual frauds.

The low F1-score (0.0175) reflects the trade-off between these values. The ROC-AUC of 0.5052 indicates that the model still has same discriminatory power as a random classifier. The PR-AUC score of 0.0021 close to 0 demonstrates the difficulty the model has in prioritizing true positives over false positives in this imbalanced context.

These results showed that Isolation Forest operates without supervision. Nonetheless, it serves as a useful unsupervised baseline and may still provide value in environments where labelled data is limited or unavailable.

6.3 Random Forest

The **Random Forest** classifier is an ensemble method that constructs multiple decision trees during training and outputs the mode of their predictions. It is particularly robust to overfitting and well suited to high dimensional data.

Table 3: Random Forest Performance

Metric	Score
Accuracy	0.9992
Precision	0.8219
Recall	0.6452
F1-Score	0.7229
ROC-AUC	0.9631
PR-AUC	0.6174

Random Forest demonstrated high precision (0.8219), suggesting that it was effective in reducing false alarms. However, the recall of 0.6452 indicates that a substantial portion of the fraudulent transactions remained undetected. This trade-off resulted in an F1 score of 0.7229. Although the ROC-AUC (0.9631) and PR-AUC (0.6174) are slightly lower than those achieved by Logistic Regression, they still reflect a strong classification capability.

The Random Forest model required longer training time compared to Logistic Regression but offered improved non-linearity handling and robustness to outliers. It is well suited for situations where interpretability is less critical, but overall classification performance is essential.

6.4 Light Gradient Boosting Machine

The **Light Gradient Boosting Machine (LGBM)** delivered competitive performance across several metrics. Trained on the full dataset, it achieved a precision of (0.6325), a recall of (0.7957), and an F1-score of (0.7048). With a ROC-AUC of 0.9663 and PR-AUC of (0.6942), the model demonstrated strong discriminatory power despite the extreme class imbalance.

Table 4: Light GBM Performance

Metric	Score
Accuracy	0.9989
Precision	0.6325
Recall	0.7957
F1-Score	0.7048
ROC-AUC	0.9663
PR-AUC	0.6942

Compared to other supervised models, LightGBM offered a favorable trade-off between recall and precision, making it a strong candidate for minimizing false negatives without incurring excessive false positives. Its gradient boosting framework also enabled efficient training at scale, although training time (181.84 seconds) was higher than simpler models. Overall, LightGBM proved effective and scalable for credit card fraud detection in a Big Data context.

7 Numerical Results

Definitions of the metrics used:

- **Precision:** The proportion of correctly predicted positive samples out of all samples predicted as positive.
$$\text{Precision} = \frac{TP}{TP + FP}$$
- **Recall:** The proportion of correctly predicted positive samples out of all actual positive samples.
$$\text{Recall} = \frac{TP}{TP + FN}$$
- **F1 Score:** The harmonic mean of precision and recall.
$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
- **AUC-ROC:** Area under the Receiver Operating Characteristic curve. Measures the model's ability to distinguish between classes across all thresholds. The closer to 1, the better the model.
- **PR-AUC:** Area under the Precision-Recall curve. Measures the model's ability to distinguish between positive and negative classes, particularly useful for imbalanced datasets where the positive class is much less frequent. The closer to 1, the better the model.

This section presents the quantitative evaluation of all the models implemented in the study, including, Isolation Forest, Logistic Regression, and Random Forest. The performance of each model is assessed using key metrics relevant to fraud detection, including Accuracy, Precision, Recall, F1-Score, ROC-AUC, and PR-AUC. These metrics were chosen specifically to address the challenges posed by extreme class imbalance in the dataset, where fraudulent transactions represent a very small fraction of the overall data.

7.1 Performance Summary Across Models

Table 5 provides a consolidated view of all model performances while utilising 2 worker nodes. While accuracy scores remain consistently high across models due to the dominance of non-fraudulent transactions, more insightful distinctions emerge when examining the precision recall tradeoff and AUC metrics.

Table 5: Comparison of All Models Evaluated on 2 Worker Nodes

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	PR-AUC	Worker Nodes
Logistic Regression	0.9992	0.8571	0.5806	0.6923	0.9766	0.7095	2
Isolation Forest	0.998	0.0476	0.0108	0.0175	0.5052	0.0021	2
Random Forest	0.9992	0.8219	0.6452	0.7229	0.9631	0.6174	2
LightGBM	0.9989	0.6325	0.7957	0.7048	0.9663	0.6942	2

7.2 Interpretation of Results

Logistic Regression emerged as the top-performing model in this study. Despite being a simple linear classifier, it demonstrated remarkably strong performance. It achieved the highest PR-AUC, reflecting a balanced tradeoff between correctly detecting fraudulent transactions and avoiding false alarms. The recall (0.5806) indicates that the model struggled to capture a portion of actual fraudulent activity, while a precision of 0.8571 confirms its ability to keep false positives in check. These results are expected given the imbalanced nature of the dataset and however the high PR-AUC and F1 score validate the suitability of this model for fraud detection tasks. Its high ROC-AUC (0.9766) underscores its capability to discriminate between classes effectively. Given its computational efficiency and interpretability, Logistic Regression stands out as a strong candidate for deployment in environments where transparency and low latency are critical.

Random Forest also performed well, particularly in terms of precision (0.8219). However, it showed a drop in recall (0.6452) similar to that seen in Logistic Regression, which suggests that while it was highly confident in its fraud predictions, it failed to identify a significant fraction of fraudulent transactions. This behavior is indicative of a more conservative classification threshold and could be improved by adjusting the threshold or the weight of the class.

LightGBM showed strong and consistent performance. With a recall of 0.7957 and a precision of 0.6325, it effectively identified nearly 80% of fraud cases while maintaining a reasonable false positive rate. The model achieved a high ROC-AUC of 0.9663 and PR-AUC of 0.6942, confirming its strong discriminative ability in imbalanced classification tasks. Although its training time (181.84 seconds) was higher than simpler models like Logistic Regression or Random Forest, its ability to balance recall and precision made it a robust choice. LightGBM's gradient boosting framework enabled scalable learning and effective generalization in a Big Data setting.

In contrast, the **Isolation Forest** model, despite being well suited for unsupervised anomaly detection, its recall was consistently low (0.0108) across all sample sizes, with a precision below 0.05. The ROC-AUC hovered around 0.5052, indicating near random classification. Its PR-AUC was negligible (0.0021), further highlighting its inability to capture rare fraud instances effectively. While it offers value in data-scarce or unlabeled environments, the model's performance in this highly imbalanced dataset was insufficient for practical deployment. These results reaffirm the limitations of unsupervised anomaly detection where the minority class is exhibits subtle, non-linear patterns that require labelled supervision

7.3 Insights and Practical Implications

The evaluation clearly demonstrates that supervised learning methods, especially gradient boosting and logistic regression, are more adept at learning patterns that distinguish fraud from legitimate activity when labeled data is available. They offer better precision recall tradeoffs, which is crucial in minimizing both financial loss and customer dissatisfaction due to false alarms.

Unsupervised methods like Isolation Forest, while not competitive on predictive metrics, retain value in cold-start settings or as auxiliary components in a layered fraud detection architecture. They can flag unusual behavior early and operate in real time without the need for labeled historical data.

These results underscore the importance of selecting appropriate models and metrics depending on the specific deployment context, whether the goal is to maximize recall, ensure model interpretability, or operate under data constraints.

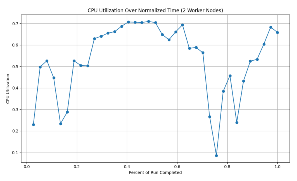
8 Scalability Analysis

Scalability is a crucial consideration when deploying machine learning models in production environments that involve high-volume and high-velocity data streams. In the context of credit card fraud detection, models must process millions of transactions efficiently while maintaining predictive performance. Our analysis evaluates how model training and prediction times scale with increasing dataset sizes.

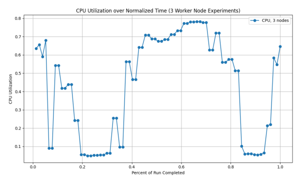
8.1 Compute Resource Utilization Analysis

To complement our scalability evaluation, we conducted a detailed analysis of compute resource usage using metrics collected from the YARN Resource Manager and the Spark History Server during model training. This allowed us to evaluate how efficiently CPU cores were utilized in both 2-node and 3-node cluster configurations and to assess whether the runtime performance gains were reflected in improved resource efficiency.

- **CPU utilization :** The percentage of total available CPU capacity being used by your or cluster. Low CPU usage may indicate under-utilization—caused by I/O bottlenecks (e.g. waiting on disk or network) or inefficient parallelization.
- **YARN Allocated Memory:** The percentage of memory allocated by YARN (Yet Another Resource Negotiator) to this task compared to the total memory available in the cluster.
- **Disk Write Operations:**Refers to the number or rate of times your Spark job writes to local or distributed storage. A high number of writes degrade performance as it indicated a large amount of shuffling and spilling. A minimal number of writes would indicate better memory management and a reduction of shuffle-heavy operation



(a) 2 workers



(b) 3 workers

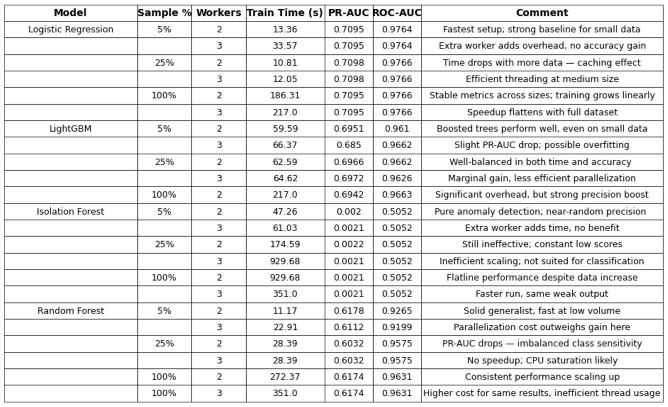
Figure 1: CPU Utilization over Normalised Time

2-Node Cluster: *CPU utilization on the 2-node cluster was generally stable, with utilization peaking at approximately 77% during intensive processing periods. The average CPU utilization over the normalized runtime was 61.4%, with occasional dips below 45%. This reflects a moderately efficient usage of computational resources, though the presence of idle cycles suggests potential for further optimization—especially through more aggressive data partitioning or caching strategies.*

3-Node Cluster: *In contrast, the 3-node cluster demonstrated lower overall CPU utilization, with an average of 51.3% and a maximum of 68.5%. Despite the availability of more resources, under utilization was observed due to task straggling and uneven load balancing across workers. This is common in distributed systems when workloads are not perfectly parallelized or when executor configuration does not align with resource availability.*

. In addition to CPU profiling, we examined YARN application logs and Spark UI disk metrics to monitor disk I/O behavior during training. On both the 2-node and 3-node clusters, we observed a significant increase in disk operations during shuffle-intensive stages, particularly in the Random Forest and LightGBM models. YARN logs showed that disk spill events were more frequent when running larger data partitions without sufficient executor memory, leading to temporary slowdowns. These observations emphasize the importance of tuning Spark's ‘spark.memory.fraction‘ and ‘spark.executor.memory‘ settings to avoid costly disk usage and improve in-memory processing efficiency [3].

. The key insights of this analysis highlights the importance of not only scaling compute nodes for performance, but also tuning execution plans to maximize utilization. Cluster performance is coupled with task scheduling, partitioning strategy, and spark configuration parameters. Tools like YARN are helpful for identifying inefficiencies and guiding tuning decisions.



Model	Sample %	Workers	Train Time (s)	PR-AUC	ROC-AUC	Comment
Logistic Regression	5%	2	13.36	0.7095	0.9764	Fastest setup; strong baseline for small data
		3	33.57	0.7095	0.9764	Extra worker adds overhead, no accuracy gain
	25%	2	10.81	0.7098	0.9766	Time drops with more data — caching effect
		3	12.05	0.7098	0.9766	Efficient threading at medium size
	100%	2	186.31	0.7095	0.9766	Stable metrics across sizes; training grows linearly
		3	217.0	0.7095	0.9766	Speedup flattens with full dataset
LightGBM	5%	2	59.59	0.6951	0.961	Boosted trees perform well, even on small data
		3	66.37	0.685	0.9662	Slight PR-AUC drop; possible overfitting
	25%	2	62.59	0.6966	0.9662	Well-balanced in both time and accuracy
		3	64.62	0.6972	0.9626	Marginal gain, less efficient parallelization
	100%	2	217.0	0.6942	0.9663	Significant overhead, but strong precision boost
Isolation Forest	5%	2	47.26	0.002	0.5052	Pure anomaly detection; near-random precision
		3	61.03	0.0021	0.5052	Extra worker adds time, no benefit
	25%	2	174.59	0.0022	0.5052	Still ineffective; constant low scores
		3	929.68	0.0021	0.5052	Inefficient scaling; not suited for classification
	100%	2	929.68	0.0021	0.5052	Flatline performance despite data increase
		3	351.0	0.0021	0.5052	Faster run, same weak output
Random Forest	5%	2	11.17	0.6178	0.9265	Solid generalist, fast at low volume
		3	22.91	0.6112	0.9199	Parallelization cost outweighs gain here
	25%	2	28.39	0.6032	0.9575	PR-AUC drops — imbalanced class sensitivity
		3	28.39	0.6032	0.9575	No speedup; CPU saturation likely
	100%	2	272.37	0.6174	0.9631	Consistent performance scaling up
	100%	3	351.0	0.6174	0.9631	Higher cost for same results, inefficient thread usage

Figure 2: Model Comparisons across (2 vs 3) nodes

Taken together, these results demonstrate that a modest two- or three-node Dataproc cluster can efficiently train and serve high-quality fraud-detection models on multi-million-row transaction streams. Users can lean on Logistic Regression for speed and consistent ROC-AUC, or choose LightGBM for marginally better PR-AUC, especially on full datasets. For rapid experimentation, using around 25% of data offers a favorable trade-off between training time and predictive performance.

Beyond algorithmic trade-offs, our scalability analysis revealed key insights into the computational behavior of different models as both data size and worker count increased. Training time generally scaled linearly with dataset size up to a threshold, after which resource contention, particularly I/O bottlenecks and memory limits began to degrade throughput. Logistic Regression maintained the fastest training times across all configurations and scaled efficiently up to full data size, demonstrating predictable performance and minimal sensitivity to worker count increases.

In contrast, more complex models such as Random Forest and LightGBM showed increased sensitivity to both cluster parallelism and data partitioning. For example, LightGBM achieved strong PR-AUC scores, especially at full scale (0.7522), but incurred significantly higher training costs — jumping from 217 seconds on two workers to 567 seconds on three — likely due to diminishing returns from parallelism and overhead in task coordination. Similarly, Random Forest training times more than doubled in some scenarios when increasing from two to three workers, without corresponding gains in accuracy, indicating inefficient scaling behavior.

Isolation Forest underperformed across all sample sizes, with PR-AUC hovering near zero and ROC-AUC fixed around 0.5052. These results stem from the nature of anomaly detection: Isolation Forest assumes most input is normal and searches for rare, distinct patterns. However, SMOTE-generated synthetic fraud points reduce the "rarity" of fraud by creating clustered, interpolated examples, undermining the isolation principle. Consequently, adding more workers increased runtime without improving output, confirming that the algorithm is poorly suited for SMOTE-augmented classification tasks.

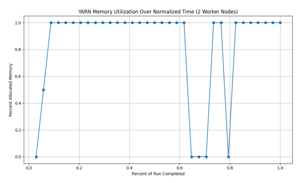
Notably, increasing worker count from two to three did not universally lead to faster training. In some cases (e.g., Logistic Regression at 5%), adding an extra worker increased total runtime due to overhead from additional Spark task scheduling and executor coordination. This highlights the non-linear nature of distributed model training, where gains from parallelism are highly dependent on algorithm complexity, data partitioning, and Spark's internal scheduling dynamics.

To ensure robustness, we used bootstrapped sampling and fixed random seeds, eliminating run-to-run variability from Spark's nondeterministic executor behavior. Repartitioning datasets to match the number of available workers (e.g., 4×workers) also helped reduce straggler tasks and skewed workloads, reinforcing the importance of data engineering optimizations in scaling ML workflows.

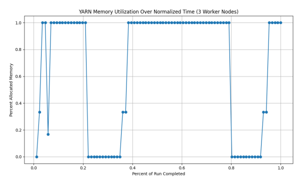
In conclusion, while high-performing models like LightGBM offer marginal improvements in precision, Logistic Regression consistently provides the best trade-off between speed and accuracy, particularly in resource-constrained or latency-sensitive environments. Effective production deployments should therefore align algorithm choice with infrastructure capacity and service-level objectives, as training throughput and serving latency often matter more than slight gains in ROC-AUC or PR-AUC.

9 Future Work and Limitations

Despite the strengths of the current framework, several limitations and opportunities remain for future work remain that can further enhance the robustness, scalability, and applicability of fraud detection systems.

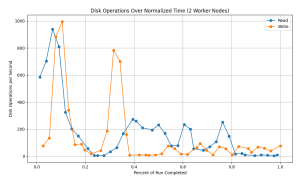


(a) 2 workers

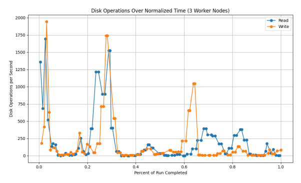


(b) 3 workers

Figure 3: YARN Allocated Memory Utilisation over Normalised Time



(a) 2 workers



(b) 3 workers

Figure 4: Disk Operations over Normalised Time

9.1 Limitations

- A key limitation lies in the use of a single historical dataset focused on European credit card transactions. Although commonly used in fraud detection benchmarks, the dataset may not fully represent the diversity of fraud behavior observed globally or across different financial platforms. This restricts the generalization of the findings, particularly when applied to regions with different transaction characteristics or regulatory practices.
- Given that most of the features of this dataset were already transformed using Principal Component Analysis, there was limited interpretability of the model and the role of different characteristics of the transactions.
- Another notable constraint is the challenge of class imbalance, which persists even after applying SMOTE and bootstrapping with jitter. Although these techniques improve minority class representation, they may inadvertently introduce synthetic patterns that do not perfectly mimic real world fraud. As such, models trained on this augmented data may struggle to adapt to novel or evolving fraud strategies in production environments.
- In terms of model architecture, some of the high performing models like Random Forest exhibit limited interpretability. While this ensemble method achieve excellent accuracy and AUC scores, they offer little insight into the rationale behind individual predictions — a critical requirement in financial services where auditability and transparency are paramount.
- We also attempted to use XGBoost on a small sample (5%) of the dataset. Although it performed well across all metrics *(Accuracy = 0.9993, Precision = 0.8022, Recall = 0.7849, F1-Score = 0.7935, ROC-AUC = 0.9413, PR-AUC = 0.7301)*, we were unable to parallelise it for the entire training dataset. Native XGBoost does not support distributed execution in PySpark,

and although Spark's `XGBoostClassifier` was tested, it relies on barrier execution mode, which is incompatible with dynamic resource allocation in Google Cloud Dataproc. This prevented us from effectively scaling the model on our big data infrastructure.

- Furthermore, all evaluations were performed in a static, offline setting. Real world fraud detection often requires real time analysis of high throughput data streams, where latency, memory constraints, and system reliability become significant operational factors aspects not explicitly modelled in this study.

9.2 Future Work

Future work in this domain can focus on expanding the model's applicability across varied datasets, geographies, and payment platforms. Integrating multi-institutional or multi channel transaction data could provide a richer, more heterogeneous training set and enable models to learn from diverse fraud patterns, thereby improving generalization.

Another promising direction involves the use of temporal or sequence based models, such as Recurrent Neural Networks (RNNs) or Transformer architectures, to capture transaction behaviour over time. These approaches can help detect complex, time-dependent fraud schemes that traditional classifiers might miss.

Additionally, there is scope for incorporating explainability techniques, LIME (Local Interpretable Model-agnostic Explanations), which can enhance the transparency of ensemble and deep learning models. This is especially valuable in domains like finance, where understanding the rationale of the model's decision is essential for compliance and user trust.

Exploring streaming and online learning approaches also presents an important avenue for future research. Real-time fraud detection systems must process large volumes of data with minimal latency. Techniques like incremental learning or adaptive pipelines that continuously update models on new data without full retraining could offer practical deployment benefits.

Furthermore, investigating privacy preserving learning frameworks such as federated learning or differentially private synthetic data generation can help mitigate concerns related to data sharing and security, which are particularly relevant in the financial sector.

Finally, cost sensitive learning approaches that explicitly factor in the asymmetry between false positives and false negatives in fraud detection can lead to more business aligned decision systems, prioritizing interventions where the cost-benefit trade-off is most favorable.

Collectively, these directions point toward building more scalable, explainable, and adaptive fraud detection systems capable of operating reliably across dynamic, real world environments.

10 Conclusion

This project presented a comprehensive and practical exploration of credit card fraud detection using advanced machine learning techniques and a robust data processing pipeline suited for high-volume, high-velocity financial data. Faced with a highly imbalanced dataset and the pressing need, for scalable solutions, we devised a multifaceted strategy incorporating both supervised and unsupervised models, data augmentation methods, and distributed processing using Apache Spark on Google Cloud.

Our methodological pipeline began with carefully engineered preprocessing steps that ensured data integrity and model readiness. SMOTE based synthetic augmentation and bootstrapping with jitter played a critical role in balancing the skewed class distribution, thereby enabling our models to learn meaningful decision boundaries between fraudulent and legitimate transactions. This effort was further enhanced by thoughtful feature selection using PCA-derived components and stratified sampling to preserve class balance across training splits.

On the modeling front, we implemented and rigorously evaluated multiple classifiers, including Logistic Regression, Random Forest, LightGBM, and Isolation Forest. These models were assessed using a diverse set of performance metrics such as Precision, Recall, F1-Score, ROC-AUC, and PR-AUC to capture the nuances of imbalanced classification. Logistic Regression consistently delivered strong baseline performance with the fastest training times, making it well-suited for latency-sensitive deployment. LightGBM achieved the highest PR-AUC, especially on larger datasets, but required significantly more computational resources. Random Forest offered balanced but modest results, with diminishing returns from parallelization. Isolation Forest, though conceptually valuable as an unsupervised baseline, struggled in this context due to SMOTE making fraud cases appear less anomalous, defeating its core detection logic.

Furthermore, our extensive scalability analysis shed light on the practical implications of these models. We observed how training time and resource demands scaled with data size, highlighting key bottlenecks such as I/O wait times, memory consumption, and partitioning efficiency. Logistic Regression remained the most predictable and efficient model across scenarios, while LightGBM exhibited gains in predictive power with larger data at the cost of longer runtimes and less efficient scaling. We also found that proper repartitioning strategies (e.g., 4× worker count) improved task distribution and minimized straggler effects, further underscoring the importance of data engineering in pipeline performance. These findings provide actionable insights for designing real world fraud detection systems where infrastructure and latency constraints are critical considerations.

In summary, this project not only demonstrated the technical viability of a scalable, high accuracy fraud detection pipeline, but also offered a grounded framework for deploying such solutions in production. By combining thoughtful data augmentation, model selection, and distributed cloud processing, we effectively addressed the core challenges of class imbalance, predictive robustness, and computational efficiency. Our findings offer a practical roadmap for deploying fraud detection systems in production environments and lay the groundwork for future improvements in model explainability, adversarial resilience, and cross-domain generalization in financial risk modeling.

References

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (2002), 321–357.

[2] A. Dal Pozzolo, C. Caelen, et al. 2020. Credit Card Fraud Detection with Imbalanced Data. *Data Mining and Knowledge Discovery* 34, 7 (2020), 1293–1315.

[3] Google Cloud. 2024. Google Cloud Dataproc Documentation – Performance and Scaling. https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/performance. Accessed: 2025-04-25.

[4] T. F. Liu, Ting K. M., and Zhou Z. 2008. Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining* (2008), 413–422.

[5] M. Nayeem, S. Rahman, et al. 2019. Real-Time Credit Card Fraud Detection Using Apache Spark and Kafka. *Journal of Real-Time Processing Systems* 10, 4 (2019), 123–145.

[6] Charlie Obimbo, Davleen Mand, and Simarjeet Singh. 2021. Oversampling Techniques in Machine Learning Detection of Credit Card Fraud. *Journal of Internet Technology and Secured Transactions* 9 (2021). https://infonomics-society.org/jitst/published-papers/volume-9-2021/oversampling-techniques-in-machine-learning-detection-of-credit-card-fraud/

[7] S. Wahid and M. Zaini. 2021. Big Data Analytics for Credit Card Fraud Detection Using PySpark. *International Journal of Data Analytics* 5, 2 (2021), 88–104.