University of Essex
Department of Mathematical Sciences

MA981: DISSERTATION

Behaviour-Aware Credit Card Fraud Detection: A Comparative Analysis of Machine Learning Approaches with Engineered Spending Pattern Features

SASIKUMAR VENKATESAN
2411729

Supervisor: Dr Na You

December 12, 2025
Colchester

Abstract

Credit card fraud remains a critical concern for financial institutions and consumers, particularly with increasing online and card-not-present transactions. Effective detection is challenged by severe class imbalance and the need to minimise both missed fraud and false alarms.This study develops a behaviour-aware fraud detection framework integrating transactional attributes with customer-specific spending pattern features. A dataset of 1,296,675 transactions including 7,506 fraud cases is analysed. The dataset is enriched with 42 engineered features across seven categories including temporal, geographic, behavioural, and spending patterns. Class imbalance is addressed using SMOTE.Three models—Logistic Regression, Random Forest, and XGBoost—are evaluated using precision, recall, F1 score, and ROC AUC. A four-stage experimental framework assesses hyperparameter tuning and feature engineering contributions across twelve configurations.The champion model, XGBoost with spending features, achieves F1 of 0.9556, precision of 96.87%, recall of 94.29%, and ROC AUC of 0.9999. Spending features contribute 95.42% of total performance improvement versus 4.70% from hyperparameter tuning. Deviation from lifetime spending average emerges as the most predictive feature with a correlation of 0.515 with fraud.

These findings confirm behavioural feature engineering as the dominant driver of fraud detection performance, offering practical value for financial institutions adopting machine learning-based fraud prevention systems.

Keywords: Credit card fraud detection, Machine learning, XGBoost, Random Forest, Feature engineering, Spending patterns, Behavioural analysis, SMOTE, Class imbalance

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Background and Context

Credit card fraud constitutes one of the most significant financial crimes affecting the global economy, with losses continuing to escalate despite substantial investments in prevention technologies. The proliferation of electronic commerce and digital payment channels has fundamentally transformed the fraud landscape, creating significant vulnerabilities exploited by increasingly sophisticated criminal networks [1]. Studies indicate that fraudulent transactions cause billions of dollars in annual losses to financial institutions and cardholders worldwide, necessitating the development of robust automated detection systems [3].The detection of fraudulent transactions presents considerable technical challenges, primarily due to severe class imbalance wherein legitimate transactions vastly outnumber fraudulent ones [2]. Traditional rule based systems have proven inadequate in addressing the evolving nature of fraud patterns, prompting researchers to explore machine learning approaches capable of identifying complex non linear relationships within transaction data [4]. Furthermore, the high dimensionality of transaction datasets and the requirement for real time processing impose additional constraints on detection methodologies [1].Recent advances in machine learning and deep learning have demonstrated promising results in fraud detection applications. Ensemble methods such as Random Forest and gradient boosting algorithms, combined with resampling techniques including the Synthetic Minority Over sampling Technique (SMOTE), have shown improved performance in handling imbalanced datasets [2, 3].

These developments underscore the imperative for continued research into behaviour aware detection frameworks that integrate customer spending patterns with advanced classification algorithms.

1.2 Research Gap and Motivation

Despite extensive research in credit card fraud detection, several significant gaps persist in the existing literature [1, 18]. Firstly, systematic comparative evaluations under standardised experimental conditions remain scarce, with methodological heterogeneity limiting direct comparisons between approaches [9, 7]. Secondly, many studies treat transactions as independent observations, neglecting temporal spending patterns and behavioural anomalies that characterise fraudulent activity [12, 11, 3].The extreme class imbalance inherent in fraud datasets continues to pose substantial challenges, with fraudulent transactions typically comprising less than one percent of observations [2, 15, 16]. Furthermore, a tension exists between model performance and interpretability, as regulatory requirements increasingly mandate transparent decision-making processes [4, 5]. Finally, the integration of gradient boosting methods with behaviour-aware feature engineering remains underexplored [8, 19].This research addresses these gaps by developing a comprehensive framework integrating engineered behavioural features with systematic comparison of Logistic Regression, Random Forest, and XGBoost classifiers under standardised conditions.

1.3 Research Questions

To address the identified research gaps, this dissertation seeks to answer the following research questions:

1. **RQ1:** What is the baseline fraud detection performance of Logistic Regression, Random Forest, and XGBoost classifiers using default hyperparameters on the original feature set?

2. **RQ2:** To what extent does hyperparameter optimisation improve the performance of each classifier compared to their baseline configurations?

3. **RQ3:** Do engineered spending behavioural features significantly improve fraud detection performance across all three classifiers?

4. **RQ4:** Is hyperparameter re-optimisation necessary when expanding the feature space, or can optimal parameters be transferred from the original feature set?

These research questions guide a systematic four-stage experimental design, enabling rigorous evaluation of the interplay between feature engineering and hyperparameter optimisation in credit card fraud detection.

1.4 Aim and Objectives

This research aims to develop and evaluate a behaviour-aware credit card fraud detection framework by systematically analysing the effects of engineered behavioural features and hyperparameter optimisation on model performance.

To achieve this aim and address the stated research questions, the following objectives are pursued:

1. Establish the baseline classification performance of Logistic Regression, Random Forest, and XGBoost using default hyperparameters. **(RQ1)**

2. Apply hyperparameter optimisation via GridSearchCV with cross-validation, and quantify the resultant performance improvements. **(RQ2)**

3. Engineer behavioural features based on customer spending patterns and assess their contribution to fraud detection effectiveness. **(RQ3)**

4. Evaluate the necessity of re-optimising hyperparameters when the feature space is expanded with engineered variables. **(RQ4)**

5. Analyse feature importances to identify the most influential predictors, and derive practical recommendations for real-world fraud detection systems.

1.5 Methodology Overview

This research adopts a quantitative experimental methodology to evaluate the effectiveness of behaviour-aware features in credit card fraud detection. The investigation utilises a publicly

available dataset containing over 1.2 million credit card transactions, offering a realistic foundation for assessing detection algorithms under authentic class imbalance conditions.

The methodological framework is structured into four experimental stages:

Stage 1 — Baseline Evaluation: Three supervised learning algorithms are implemented with default hyperparameters: Logistic Regression as an interpretable baseline, Random Forest as an ensemble learner, and XGBoost as a high-performance gradient boosting model [1, 8, 6]. This establishes reference performance metrics for subsequent comparison.

Stage 2 — Hyperparameter Optimisation: GridSearchCV with 5-fold stratified cross-validation is applied to identify optimal configurations for each algorithm, quantifying the contribution of systematic tuning to fraud detection performance [9, 23].

Stage 3 — Behavioural Feature Engineering: The dataset is enriched with 42 engineered features across seven categories, including spending pattern features capturing customer-specific deviations from historical behaviour [27, 26, 5]. To address the extreme class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) is applied during model training [2, 15].

Stage 4 — Re-optimisation: Hyperparameters are re-tuned on the expanded feature set to determine whether optimal parameters transfer across feature spaces or require adjustment.

Model performance is assessed using metrics critical for fraud detection: precision, recall, F1-score, and ROC-AUC [9, 14]. This four-stage framework enables rigorous quantification of the relative contributions from algorithm selection, hyperparameter tuning, and feature engineering to fraud detection effectiveness.

1.6 Structure of the Dissertation

This dissertation is organised into six main chapters:

- **Chapter 1: Introduction** — Provides background, outlines the research problem, defines the objectives, and presents an overview of the methodology.
- **Chapter 2: Literature Review** — Critically examines existing research on credit card fraud detection, covering traditional machine learning approaches, ensemble methods, deep learning techniques, and feature engineering strategies.

- **Chapter 3: Methodology** — Details the dataset, preprocessing steps, feature engineering techniques, model architectures, training strategies, and evaluation metrics.
- **Chapter 4: Data Analysis and Exploration** — Presents exploratory data analysis, risk factor identification, and comprehensive feature engineering including spending pattern features.
- **Chapter 5: Results and Discussion** — Evaluates twelve model configurations across four experimental stages, analyses feature importance, addresses research questions, and interprets findings in relation to existing literature.
- **Chapter 6: Conclusion** — Summarises the contributions of this research and outlines directions for future work in behaviour-aware fraud detection.

Together, these chapters build a comprehensive study that not only advances technical performance but also addresses interpretability and class imbalance—key prerequisites for practical deployment of machine learning-based credit card fraud detection systems.

CHAPTER 2

Literature Review

2.1 Introduction

Credit card fraud costs billions of pounds every year, and as more transactions move online, the problem keeps growing. Machine learning offers a solution teaching computers to spot the subtle patterns that distinguish fraudulent transactions from legitimate ones [22]. But there is a catch: fraud is rare. For every thousand transactions, perhaps only one or two are fraudulent. Imagine finding a needle in a haystack, except the haystack keeps growing and the needle keeps changing shape [9].This chapter reviews how the field has evolved. It begins with traditional methods like logistic regression and support vector machines [21], then explores how ensemble methods changed the game [6]. It examines deep learning approaches [18] and discusses two critical challenges: severe class imbalance [2] and the importance of behavioural feature engineering [27]. By the end, the research gaps that motivate this study will become clear.

2.2 Traditional Machine Learning Approaches

When researchers first tackled credit card fraud detection, they reached for the classic tools of machine learning. Logistic regression was a natural starting point it is simple, interpretable, and had been successfully applied to similar binary classification problems for decades [21]. The idea is straightforward: calculate the probability that a transaction is fraudulent based on

its features, and flag anything above a certain threshold. Support vector machines offered a different approach. Instead of calculating probabilities, they try to find the optimal boundary that separates fraudulent transactions from legitimate ones [23]. In theory, this works well when the two classes are clearly distinguishable. In practice, fraud detection rarely offers such clean separation.Several comparative studies have tested these traditional methods head-to-head. [21] found that while logistic regression achieved reasonable accuracy, it struggled to catch the minority class the actual frauds. Vimala Devi and Kavitha [25] reached similar conclusions, noting that traditional classifiers tend to be biased towards the majority class.The fundamental problem is that these algorithms treat all misclassifications equally. Missing a fraud costs the same as falsely flagging a legitimate transaction. In reality, these errors have vastly different consequences, and traditional methods were not designed to handle such asymmetric costs [7].

2.3 Ensemble Methods

If one model struggles to detect fraud, why not use many? That is the core idea behind ensemble methods, and it turned out to be a game-changer for fraud detection [6].

Random Forest was among the first ensemble methods to gain traction in this domain. It works by building hundreds of decision trees, each trained on a slightly different subset of the data, and then combining their votes to make a final prediction [30]. The beauty of this approach is that individual trees might make mistakes, but the collective wisdom of the forest tends to be remarkably accurate.[30] demonstrated that Random Forest significantly outperformed single decision trees on fraud detection tasks.Then came gradient boosting, and things got even better. Unlike Random Forest, which builds trees independently, gradient boosting builds them sequentially each new tree specifically focuses on correcting the mistakes of previous ones [8]. XGBoost and LightGBM emerged as particularly powerful implementations. Taha and Malebary [8] showed that an optimised LightGBM model achieved superior performance compared to traditional methods, with the added benefit of faster training times.What makes ensemble methods particularly suited for fraud detection? They handle imbalanced data better than single classifiers, they are less prone to overfitting, and they can capture complex non-linear relationships in the data [1].[1] confirmed these advantages in a comprehensive comparison, finding that ensemble approaches consistently ranked among

the top performers. However, ensemble methods are not without trade-offs. They sacrifice some interpretability explaining why a forest of 500 trees flagged a particular transaction is harder than explaining a logistic regression coefficient [29]. For many applications, though, the performance gains justify this cost.

2.4 Deep Learning Approaches

Deep learning has revolutionised fields from image recognition to natural language processing, so it was inevitable that researchers would try applying it to fraud detection. The results have been mixed impressive in some cases, disappointing in others [18]. The appeal of deep learning is clear. Neural networks can automatically learn complex patterns without requiring manual feature engineering. Feed them enough data, and they figure out what matters on their own [24].[24] demonstrated that deep neural networks could achieve competitive performance on fraud detection, particularly when dealing with large-scale datasets.Recurrent neural networks and LSTMs brought another advantage: the ability to model sequences. Fraud often follows patterns over time a stolen card might be used for several small transactions before a large one.[28] showed that sequence-based models could capture these temporal dependencies better than traditional approaches. Similarly, [12] developed attention-based networks that learned which past transactions were most relevant for predicting fraud.More recent work has pushed the boundaries further.[10] introduced deep representation learning that maps transactions into a space where fraudulent ones cluster together. [13] combined CatBoost with deep neural networks to leverage the strengths of both approaches.Yet deep learning is not a silver bullet. These models require substantially more data and computational resources than traditional methods [18]. They are also notoriously difficult to interpret a significant drawback when banks need to explain to customers why their transaction was declined. Mienye and Jere [18] note that for many practical fraud detection scenarios, well-tuned ensemble methods still offer a better balance of performance and usability.

2.5 Class Imbalance Problem

Here is the uncomfortable truth about fraud detection: most machine learning algorithms were not designed for it. They assume roughly balanced classes, and fraud datasets are anything but balanced. Ratios of 100:1 or even 1000:1 between legitimate and fraudulent transactions are common [9].Why does this matter? When an algorithm sees 99 legitimate transactions for every fraudulent one, it learns a simple shortcut: predict everything as legitimate and achieve 99% accuracy. Technically impressive, practically useless [2].Researchers have attacked this problem from multiple angles. The most popular solution is SMOTE Synthetic Minority Over-sampling Technique. Rather than simply duplicating fraud cases, SMOTE creates new synthetic examples by interpolating between existing ones [2].[2] demonstrated that combining SMOTE with machine learning classifiers significantly improved fraud detection rates.Variations on this theme have emerged. [15] combined SMOTE with generative adversarial networks to create even more realistic synthetic fraud samples. Ileberi and Sun [19] explored ADASYN, which focuses synthetic sample generation on the hardest-to-learn cases.The opposite approach undersampling the majority class has also shown promise.[16] developed a sophisticated undersampling scheme that removes noisy samples rather than randomly discarding data. This preserves more information while still achieving better class balance.Cost-sensitive learning offers yet another perspective. Instead of rebalancing the data, these methods assign higher misclassification costs to the minority class [7].[7] showed that realistic cost modelling could substantially improve detection performance without artificial data manipulation.

The consensus is clear: addressing class imbalance is not optional. Any serious fraud detection system must incorporate one or more of these techniques [9].

2.6 Feature Engineering and Behavioural Analysis

Sometimes the secret to better fraud detection is not a fancier algorithm it is better data. Feature engineering, the art of creating meaningful variables from raw transaction data, has proven to be one of the most effective ways to improve detection rates [27].The insight is simple but powerful: a transaction does not exist in isolation. It happens at a particular time, in a particular place, by a particular customer with a particular spending history.

Capturing this context transforms how algorithms see the data [26].[27] pioneered this approach by engineering features based on transaction aggregation. Instead of just looking at whether a single transaction seems suspicious, they calculated averages and patterns over time windows. How does this transaction amount compare to the customer's typical spending? How many transactions have they made in the last hour? These contextual features dramatically improved detection performance.Behavioural profiling takes this further. The idea is to build a model of what normal looks like for each customer, then flag deviations [26], demonstrated that transaction behaviour features could capture fraud patterns that raw transaction data missed entirely.More recent work has built on these foundations.[5] combined neural networks with carefully engineered features, finding that feature engineering remained valuable even with deep learning.[17] developed a fraud feature boosting mechanism that automatically identifies the most discriminative behavioural patterns.

The message from the literature is consistent: algorithms matter, but features matter more. A simple model with excellent features often outperforms a complex model with poor features [27]. This insight directly influences the approach taken in this study.

2.7 Research Gaps

Despite significant progress, several gaps remain in the fraud detection literature that this study aims to address.First, most studies focus on a single aspect of the problem. Some papers optimise algorithms, others engineer features, and others address class imbalance but few combine all three systematically [1]. In practice, a production fraud detection system needs all of these elements working together. There is surprisingly little research examining how these components interact and whether their benefits compound.Second, the comparison between traditional machine learning and more complex approaches remains unclear. Deep learning studies often claim superior performance, but the comparisons are not always fair sometimes the baseline models lack proper tuning, or the feature engineering is inconsistent across methods [18]. A rigorous comparison using the same features and optimisation procedures across all algorithms would provide clearer guidance for practitioners.Third, behavioural feature engineering, while promising, has not been fully integrated with modern ensemble methods [5].[27] showed the value of spending pattern features, but their work predates the widespread adoption of XGBoost and other gradient boosting frameworks. How much

do behavioural features improve state-of-the-art ensemble methods? This question remains underexplored.Finally,reproducibility is a persistent issue. Many studies use proprietary datasets or fail to report implementation details, making it difficult to build on previous work [4].

Here it addresses these gaps by implementing a systematic four-stage framework that combines baseline comparison, hyperparameter optimisation, behavioural feature engineering, and re-optimisation. By applying consistent methodology across logistic regression, Random Forest, and XGBoost, and by documenting all implementation details, this research aims to provide clear, reproducible insights into what actually works for credit card fraud detection.

2.8 Comparison of Methods

Having reviewed the various approaches to fraud detection, it is worth stepping back and comparing them directly. Table 2.1 summarises the key studies discussed in this chapter, while Table 2.3 compares the strengths and weaknesses of each methodological approach.

Table 2.1: Summary of Key Studies in Credit Card Fraud Detection (Part 1)

Study	**Method(s)**	**Best Performance**	**Key Contribution**
Traditional Machine Learning			
Awoyemi et al. [21]	LR, NB, KNN	KNN: 97.9% accuracy	Comparative analysis of classifiers
Dhankhad et al. [23]	SVM, RF, LR	SVM: best precision	Supervised learning comparison
Vimala & Kavitha [25]	NB, SVM, RF	RF outperformed	Classification algorithm comparison
Ensemble Methods			
Xuan et al. [30]	Random Forest	High accuracy	RF for fraud detection
Taha & Malebary [8]	LightGBM	AUC: 0.99	Optimised gradient boosting
Alarfaj et al. [1]	XGBoost, RF, DNN	XGBoost: best overall	State-of-the-art comparison
Seera et al. [29]	Ensemble fusion	High recall	Intelligent payment system
Deep Learning			
Roy et al. [24]	DNN	Competitive accuracy	Deep learning application
Jurgovsky et al. [28]	LSTM	Improved F1-score	Sequence classification
Li et al. [10]	Deep representation	AUC: 0.98	Centre loss learning
Xie et al. [12]	Attention-based GRU	High recall	Time-aware detection
Nguyen et al. [13]	CatBoost + DNN	AUC: 0.99	Hybrid approach

Table 2.2: Summary of Key Studies in Credit Card Fraud Detection (Part 2)

Study	**Method(s)**	**Best Performance**	**Key Contribution**
Class Imbalance Handling			
Ileberi et al. [2]	SMOTE + AdaBoost	F1: 0.86	SMOTE effectiveness
Ghaleb et al. [15]	SMOTE + GAN + RF	High F1-score	Synthetic data generation
Zhu et al. [16]	NUS undersampling	Improved precision	Noisy sample removal
Dal Pozzolo et al. [7]	Cost-sensitive	Realistic modelling	Learning strategy
Feature Engineering			
Bahnsen et al. [27]	Aggregated features	13% improvement	Transaction aggregation
Kho & Vea [26]	Behavioural features	Improved detection	Transaction behaviour
Esenogho et al. [5]	NN + engineered features	AUC: 0.99	Feature engineering + DL

Table 2.3: Comparison of Methodological Approaches

Approach	**Strengths**	**Weaknesses**	**Best Use Case**
Logistic Regression	Interpretable, fast training, probabilistic outputs	Limited to linear boundaries, struggles with imbalance	Baseline model, explainability required
Support Vector Machine	Effective in high dimensions, robust to outliers	Slow on large datasets, less interpretable	Small to medium datasets
Random Forest	Handles non-linearity, resistant to overfitting, feature importance	Memory intensive, less interpretable than LR	General-purpose fraud detection
XGBoost / Gradient Boosting	State-of-the-art performance, handles imbalance well	Requires tuning, risk of overfitting	Production systems requiring high accuracy
Deep Learning	Automatic feature learning, captures complex patterns	Data hungry, computationally expensive, black box	Large-scale systems with abundant data
SMOTE	Simple to implement, proven effectiveness	Can create noisy samples, may not help all algorithms	When minority class is too small
Behavioural Features	Captures customer patterns, improves all methods	Requires domain knowledge, computational overhead	When transaction context is available

Several patterns emerge from this comparison. Ensemble methods, particularly gradient boosting variants, consistently achieve top performance across studies. Deep learning shows

promise but requires more resources and data than traditional methods. Most importantly, the studies that combine multiple strategies-class balancing, feature engineering, and algorithm optimisation-tend to report the best results [1, 5].

This observation directly motivates the integrated approach adopted in this study.

2.9 Summary

Traced the evolution of credit card fraud detection from simple statistical classifiers to sophisticated deep learning systems. Several key themes emerged.Traditional methods like logistic regression and support vector machines provided a foundation, but their inability to handle class imbalance limited their effectiveness [21]. Ensemble methods, particularly Random Forest and gradient boosting, marked a significant improvement by combining multiple models to achieve more robust predictions [1]. Deep learning brought automatic feature learning and the ability to model sequential patterns, though at the cost of interpretability and increased computational demands [18].Two cross-cutting challenges affect all approaches. Class imbalance remains a fundamental obstacle that must be addressed through techniques like SMOTE or cost-sensitive learning [2]. Feature engineering, especially behavioural features that capture customer spending patterns, has proven to be one of the most reliable ways to improve detection performance regardless of the underlying algorithm [27].The literature reveals a clear gap: while individual components have been studied extensively, few works systematically combine algorithm comparison, hyperparameter optimisation, and behavioural feature engineering within a unified framework. This study addresses that gap by implementing a four-stage approach that integrates all three elements, using logistic regression, Random Forest, and XGBoost as representative algorithms. The goal is not merely to achieve high performance, but to understand how these components interact and which combinations work best for practical fraud detection.

CHAPTER 3

Methodology

3.1 Introduction

This chapter presents the methodological framework employed to develop and evaluate machine learning models for credit card fraud detection. The research follows a four-stage experimental pipeline designed to systematically address each research question: baseline model comparison, hyperparameter optimisation, behavioural feature engineering, and re-optimisation with enhanced features.The analysis utilises a large-scale transaction dataset comprising 1,296,675 credit card transactions with 24 attributes [31]. To enhance predictive capacity and capture customer spending patterns, additional behavioural features were engineered based on temporal, geographic, and transactional aggregation principles established in the fraud detection literature [27, 26].Three classification algorithms were selected for comparative evaluation: Logistic Regression, Random Forest, and XGBoost. These algorithms represent distinct methodological approaches linear modelling, bagging-based ensemble learning, and gradient boosting enabling comprehensive assessment of algorithmic suitability for fraud detection tasks [1]. All modelling was implemented in Python using the scikit-learn library, with XGBoost utilised for gradient boosting implementation.To address the severe class imbalance inherent in fraud detection datasets, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to training data [2]. Stratified sampling ensured representative train-test partitions, whilst StandardScaler normalisation prepared features for algorithm consumption.To ensure methodological rigour and reproducibility, all randomised processes

were seeded using `random_state=42`, and precautions were implemented throughout the pipeline to prevent data leakage. The remainder of this chapter elaborates each component of the methodological framework in detail.

3.2 Research Design and Approach

Supervised machine learning techniques were employed to develop predictive models for credit card fraud detection, leveraging a large-scale dataset of labelled transaction records. Supervised learning is particularly well-suited to the objectives of this research, given the availability of ground-truth fraud labels and the binary classification nature of the task [23].

A binary classification approach was adopted, distinguishing between legitimate transactions (class 0) and fraudulent transactions (class 1). This formulation serves as the foundation for real-world fraud detection systems, enabling automated screening of transactions for potential fraud [21].

3.2.1 Four-Stage Experimental Framework

To systematically address the research questions, a four-stage experimental framework was designed. Each stage builds upon the previous, enabling isolation of the effects of hyperparameter tuning and feature engineering on model performance.

Stage 1: Baseline Evaluation. The first stage establishes baseline performance using default hyperparameters across all three algorithms-Logistic Regression, Random Forest, and XGBoost. This stage utilises the 41 base features comprising temporal, geographic, amount-based, and behavioural variables. The baseline results provide reference metrics against which all subsequent improvements are measured, directly addressing **RQ1** regarding algorithm comparison under standardised conditions.

Stage 2: Hyperparameter Optimisation. The second stage applies GridSearchCV with 5-fold stratified cross-validation to identify optimal hyperparameter configurations for each algorithm. The feature set remains unchanged at 41 variables, isolating the contribution of systematic tuning from other factors. This stage addresses **RQ2** by quantifying the performance improvement attributable to hyperparameter optimisation alone.

Stage 3: Spending Feature Integration. The third stage introduces 11 additional spending pattern features to the optimised models from Stage 2, expanding the feature set from 41 to 51

variables. These features capture customer-specific spending deviations across multiple time horizons. By retaining the Stage 2 hyperparameters, this stage isolates the contribution of behavioural feature engineering, addressing **RQ3** regarding the impact of spending pattern features on fraud detection performance.

Stage 4: Re-optimisation. The final stage re-optimises all hyperparameters using the expanded 51-feature set. This determines whether optimal parameters identified in Stage 2 transfer effectively to the new feature space or require adjustment. The results address **RQ4** concerning the necessity of re-tuning when feature spaces are modified.

This systematic approach enables direct comparison of 12 distinct model configurations (3 algorithms $\times$ 4 stages), providing comprehensive insights into the relative contributions of algorithm selection, hyperparameter optimisation, and feature engineering to fraud detection performance. The sequential design ensures that each improvement source-tuning versus features-can be quantified independently before examining their combined effect.

3.3 Data Source and Acquisition

Data were obtained from a publicly available credit card transactions dataset, sourced via Kaggle [31] in December 2024. The dataset comprises 1,296,675 individual transaction records, providing sufficient statistical power for robust model development and comparative evaluation across multiple algorithmic approaches.

3.3.1 Dataset Structure

Each transaction record contains 24 attributes spanning multiple domains: transaction details, customer demographics, merchant information, and geographic coordinates. The target variable `is_fraud` indicates whether a transaction was fraudulent (1) or legitimate (0).

The class imbalance ratio was calculated using:

$$IR = \frac{N_{legitimate}}{N_{fraud}} \tag{3.3.1}$$

where $N_{legitimate}$ and N_{fraud} represent the number of legitimate and fraudulent transactions, respectively. High imbalance ratios are characteristic of fraud detection datasets and necessitate specialised handling techniques [9].

3.3.2 Data Quality Verification

Initial quality checks were performed to assess data completeness and validity. All records were examined for missing values, duplicate entries, and out-of-range measurements. Transaction amounts were verified to be positive, geographic coordinates were validated against expected ranges, and temporal attributes were checked for consistency. These verification steps confirmed dataset integrity for subsequent analysis.

3.4 Data Preprocessing

A systematic preprocessing pipeline was implemented to prepare data for model training whilst preventing data leakage. The pipeline sequence ensured that all transformations were fitted exclusively on training data before application to test data.

3.4.1 Train-Test Split

The dataset was partitioned into training and testing sets using stratified sampling to preserve class distribution across both partitions. A 65%-35% split ratio was employed, balancing sufficient training data for model learning with adequate test data for robust performance evaluation.

Stratified sampling ensures that the proportion of each class remains consistent across partitions:

$$\frac{n_c^{train}}{N^{train}} = \frac{n_c^{test}}{N^{test}} = \frac{N_c}{N} \tag{3.4.1}$$

where n_c^{train} and n_c^{test} denote the count of class c in the training and test sets respectively, N^{train} and N^{test} are the total sizes of each partition, and N_c is the total count of class c in the full dataset.

3.4.2 Feature Scaling

StandardScaler normalisation was applied to continuous features to ensure comparable scales across all predictors. This transformation is particularly important for distance-based algorithms and gradient descent optimisation [23].

The z-score transformation was computed as:

$$z = \frac{x - \mu_{train}}{\sigma_{train}} \tag{3.4.2}$$

where x is the original feature value, μ_{train} is the mean computed from training data, and σ_{train} is the standard deviation computed from training data.

Critically, the scaler was fitted exclusively on training data and subsequently applied to both training and test sets. This approach prevents data leakage by ensuring that test set statistics do not influence the scaling parameters [7].

3.5 Feature Engineering

Domain knowledge-driven feature engineering was conducted to enhance predictive capacity and capture behavioural patterns indicative of fraudulent activity. Features were derived following established principles in the fraud detection literature [27, 26].

3.5.1 Feature Engineering Approach

Feature engineering was motivated by three objectives: capturing temporal patterns in transaction behaviour, quantifying geographic anomalies, and deriving behavioural profiles that reflect normal customer spending patterns. Unlike raw transaction attributes, engineered features enable detection of contextual anomalies transactions that appear normal in isolation but deviate from established patterns [27].

A critical consideration throughout the feature engineering process was the prevention of data leakage. All aggregation-based features were computed exclusively from training data, with statistics subsequently applied to test data through lookup operations. This approach ensures that no information from the test set influences feature values during model training [7].

3.5.2 Temporal Features

Nine temporal features were extracted from transaction timestamps to capture time-based fraud patterns. These features encode the hour, day, month, and year of each transaction,

along with derived indicators for weekend transactions, night-time transactions, and high-risk hours. Temporal features enable models to learn that fraudulent activity often exhibits distinct timing patterns compared to legitimate transactions [12].

3.5.3 Geographic Features

Four geographic features were engineered to quantify the spatial relationship between customer and merchant locations. The primary feature calculates the Euclidean distance between customer coordinates and merchant coordinates:

$$d = \sqrt{(lat_{customer} - lat_{merchant})^2 + (long_{customer} - long_{merchant})^2} \tag{3.5.1}$$

where lat and $long$ represent latitude and longitude coordinates respectively. Additional binary indicators were derived to flag high-distance transactions, medium-distance transactions, and online transactions (where distance equals zero). Large distances between customer location and transaction location may indicate card theft or cloning [27].

3.5.4 Amount-Based Features

Six amount-based features were created to capture transaction value patterns. The log transformation was applied to normalise the highly skewed distribution of transaction amounts:

$$amt_{log} = \log(1 + amt) \tag{3.5.2}$$

The addition of 1 prevents undefined values for zero-amount transactions. Binary indicators were derived using percentile thresholds to flag high-amount transactions (above 95th percentile), medium-amount transactions (above 75th percentile), and low-amount transactions (below 25th percentile). Round amount indicators identify transactions divisible by 10 or 100, which may indicate manual entry or testing behaviour.

3.5.5 Behavioural Features

Behavioural features were engineered at three entity levels: customer, merchant, and category. These features capture normal patterns against which individual transactions can be

compared.

Customer Behavioural Features: Seven features were derived from customer transaction history. The customer average transaction amount was calculated as:

$$\bar{x}_c = \frac{1}{n_c} \sum_{i=1}^{n_c} amt_i \tag{3.5.3}$$

where n_c is the number of transactions for customer c and amt_i represents individual transaction amounts. The customer standard deviation was computed as:

$$s_c = \sqrt{\frac{1}{n_c - 1} \sum_{i=1}^{n_c} (amt_i - \bar{x}_c)^2} \tag{3.5.4}$$

The amount deviation score quantifies how unusual a transaction amount is relative to the customer's historical behaviour:

$$z_{amt} = \frac{amt - \bar{x}_c}{s_c} \tag{3.5.5}$$

Additional features include transaction count, maximum transaction amount, high-activity indicators, and new customer flags for accounts with limited transaction history.

Merchant Risk Features: Four features were derived from merchant transaction patterns, including average transaction amount, transaction count, low-activity indicators, and new merchant flags.

Category-Based Features: Three features capture transaction category patterns, including category average amount, transaction count, and known risk category indicators.

3.5.6 Spending Pattern Features

Eleven additional spending pattern features were engineered to capture temporal spending behaviour. These features analyse daily spending patterns, including total daily spending, transaction frequency, average transaction amounts per day, and spending volatility. Deviation metrics compare current transactions against historical customer behaviour, enabling detection of sudden changes in spending patterns that may indicate account compromise [26].

All spending pattern features were computed using only training data, with values merged to test data through customer identifier lookups. New customers appearing only in

test data received median imputation values derived from training statistics.

3.6 Class Imbalance Handling

Credit card fraud detection datasets exhibit severe class imbalance, with legitimate transactions vastly outnumbering fraudulent ones. Standard machine learning algorithms trained on imbalanced data tend to favour the majority class, resulting in poor detection of the minority class (fraud) despite high overall accuracy [9]. To address this challenge, the Synthetic Minority Over-sampling Technique (SMOTE) was employed.

3.6.1 SMOTE Algorithm

SMOTE generates synthetic samples for the minority class by interpolating between existing minority class instances [2]. For each minority class sample x_i, the algorithm identifies its k nearest neighbours within the minority class. A synthetic sample is then created along the line segment connecting x_i to a randomly selected neighbour x_{nn}:

$$x_{syn} = x_i + \lambda \cdot (x_{nn} - x_i), \quad \lambda \in [0, 1] \tag{3.6.1}$$

where x_{syn} is the synthetic sample, x_i is the original minority class sample, x_{nn} is a randomly selected nearest neighbour, and λ is a random value between 0 and 1 that determines the position of the synthetic sample along the connecting line segment.

This approach creates plausible synthetic fraud cases that occupy the feature space between existing fraud instances, rather than simply duplicating existing samples. The interpolation mechanism helps classifiers learn more generalised decision boundaries for the minority class [2].

3.6.2 Sampling Strategy Configuration

SMOTE was configured with a sampling strategy of 0.5, meaning the minority class was oversampled until it reached 50% of the majority class size. This configuration balances the need for improved minority class representation whilst avoiding excessive synthetic sample generation that could lead to overfitting.

Critically, SMOTE was applied exclusively to training data after the train-test split. The test set remained untouched to ensure unbiased performance evaluation on the original class distribution. This sequence prevents data leakage and provides realistic assessment of model performance on naturally imbalanced data [7].

The implementation utilised the `imblearn` library with `random_state=42` for reproducibility.

3.7 Classification Algorithms

Three classification algorithms were selected for comparative evaluation, representing distinct methodological approaches to binary classification. This selection enables assessment of linear models, bagging-based ensembles, and gradient boosting methods within a unified experimental framework [1].

3.7.1 Logistic Regression

Logistic Regression is a linear classification algorithm that models the probability of class membership using the logistic (sigmoid) function [21]:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})}} \tag{3.7.1}$$

where $P(y = 1|x)$ is the probability of the positive class (fraud), β_0 is the intercept term, $\boldsymbol{\beta}$ is the vector of feature coefficients, and $\mathbf{x}$ is the feature vector.

Model parameters are estimated by minimising the binary cross-entropy loss function:

$$L = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{3.7.2}$$

where y_i is the true label, $\hat{y}_i$ is the predicted probability, and n is the number of samples.

Logistic Regression provides interpretable coefficients and serves as a strong baseline for fraud detection tasks. Its linear decision boundary offers transparency in understanding feature contributions to predictions [21].

3.7.2 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and aggregates their predictions [30]. Each tree is trained on a bootstrap sample of the data with a random subset of features considered at each split.

Individual trees partition the feature space by recursively minimising the Gini impurity:

$$Gini(t) = 1 - \sum_{i=1}^{C} p_i^2 \tag{3.7.3}$$

where p_i is the proportion of samples belonging to class i at node t, and C is the number of classes.

The final prediction is determined by majority voting across all trees:

$$\hat{y} = \text{mode}\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x})\} \tag{3.7.4}$$

where $h_t(\mathbf{x})$ is the prediction of tree t and T is the total number of trees in the forest.

Random Forest is robust to overfitting, handles non-linear relationships, and provides feature importance scores. These properties make it well-suited for fraud detection where complex feature interactions exist [30].

3.7.3 XGBoost

XGBoost (Extreme Gradient Boosting) is a gradient boosting algorithm that builds an ensemble of decision trees sequentially, with each tree correcting errors made by previous trees [1]. The algorithm optimises the following objective function:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.7.5}$$

where $l(y_i, \hat{y}_i)$ is the loss function measuring prediction error, $\Omega(f_k)$ is the regularisation term for tree f_k, and K is the number of trees.

The regularisation term controls model complexity:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.7.6}$$

where T is the number of leaves in the tree, w_j is the weight of leaf j, γ controls the minimum loss reduction for splitting, and λ is the L2 regularisation coefficient.

XGBoost incorporates built-in regularisation, handles missing values, and typically achieves state-of-the-art performance on structured data. Its gradient boosting framework enables effective learning on imbalanced datasets [8].

3.7.4 Implementation

All algorithms were implemented using Python's scikit-learn library, with XGBoost utilising the dedicated `xgboost` package. Initial baseline models were trained with default hyperparameters to establish reference performance levels.

All models were configured with `random_state=42` to ensure reproducibility, and `n_jobs=-1` was specified where applicable to utilise all available CPU cores for parallel processing.

3.8 Hyperparameter Optimisation

Hyperparameter tuning was performed using GridSearchCV to identify optimal configurations for each algorithm. This process involved exhaustive evaluation of candidate parameter combinations using cross-validation to ensure generalisability and performance stability [23].

3.8.1 Cross-Validation Framework

Stratified K-Fold cross-validation with $K = 5$ folds was employed to preserve class distributions across folds. For each hyperparameter configuration, the model was trained on four folds and validated on the remaining fold, with this process repeated for all five fold combinations.

The cross-validation score was computed as the mean performance across all folds:

$$CV = \frac{1}{K} \sum_{k=1}^{K} Score(D_{val}^{(k)}) \tag{3.8.1}$$

where $Score(D_{val}^{(k)})$ is the performance metric evaluated on validation fold k, and K is the total number of folds.

Stratification ensures that each fold maintains the same class proportion as the full training set, which is critical for imbalanced datasets where random partitioning could result in folds with insufficient minority class representation [9].

3.8.2 Scoring Strategy

The F1-score was selected as the primary optimisation metric due to its suitability for imbalanced classification tasks. Unlike accuracy, which can be misleadingly high when the model simply predicts the majority class, F1-score balances precision and recall, providing a more meaningful measure of fraud detection capability [2].

All hyperparameter optimisation procedures were conducted using GridSearchCV from scikit-learn with `random_state=42` to ensure reproducibility. Single-core processing (`n_jobs=1`) was employed to ensure stable execution across computing environments.

3.9 Evaluation Framework

Model performance was evaluated using a comprehensive suite of metrics designed to capture different aspects of classification quality. Given the severe class imbalance in fraud detection, multiple complementary metrics were employed to provide a holistic assessment beyond simple accuracy [2].

3.9.1 Confusion Matrix

The confusion matrix provides the foundation for all classification metrics, summarising the four possible prediction outcomes. Table 3.1 presents the confusion matrix structure.

Table 3.1: Confusion Matrix Structure

	Predicted Legitimate	**Predicted Fraud**
Actual Legitimate	True Negative (TN)	False Positive (FP)
Actual Fraud	False Negative (FN)	True Positive (TP)

In fraud detection, false negatives (missed frauds) typically carry higher costs than false positives (legitimate transactions flagged as suspicious). This asymmetry influences the relative importance of different evaluation metrics [7].

3.9.2 Classification Metrics

Accuracy measures the overall proportion of correct predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9.1}$$

Whilst intuitive, accuracy can be misleading for imbalanced datasets where a model predicting all transactions as legitimate would achieve high accuracy despite failing to detect any fraud [9].

Precision quantifies the proportion of predicted frauds that are actually fraudulent:

$$Precision = \frac{TP}{TP + FP} \tag{3.9.2}$$

High precision indicates that when the model flags a transaction as fraud, it is likely correct, minimising false alarms that inconvenience legitimate customers.

Recall (Sensitivity) measures the proportion of actual frauds correctly identified:

$$Recall = \frac{TP}{TP + FN} \tag{3.9.3}$$

High recall is critical for fraud detection, as missed frauds result in direct financial losses. This metric captures the model's ability to detect fraudulent transactions [1].

Specificity measures the proportion of legitimate transactions correctly identified:

$$Specificity = \frac{TN}{TN + FP} \tag{3.9.4}$$

High specificity ensures that legitimate customers are not unnecessarily inconvenienced by false fraud alerts.

F1-Score provides the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.9.5}$$

The F1-score balances the trade-off between precision and recall, providing a single metric that penalises extreme imbalances between the two. This metric was selected as the primary evaluation criterion for model comparison [2].

3.9.3 ROC-AUC Analysis

The Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate across all classification thresholds. The Area Under the Curve (AUC) summarises this relationship as a single value:

$$AUC = \int_0^1 TPR(FPR^{-1}(t))\, dt \tag{3.9.6}$$

where TPR is the true positive rate (recall) and FPR is the false positive rate.

AUC values range from 0.5 (random guessing) to 1.0 (perfect discrimination). This metric is particularly valuable for fraud detection as it evaluates model performance across all possible decision thresholds, independent of any specific threshold choice [1].

3.9.4 Implementation

All evaluation metrics were computed using the `sklearn.metrics` module. Confusion matrices were generated for each model to visualise classification outcomes and identify patterns in misclassification. ROC curves were plotted to assess discrimination capability across thresholds.

3.10 Experimental Procedure

This section describes the complete experimental pipeline, reproducibility measures, and ethical considerations underlying the research.

3.10.1 Pipeline Overview

The experimental procedure followed a systematic pipeline integrating all methodological components. Table 3.2 summarises the sequential stages of the experimental workflow.

The pipeline was designed to enable isolation of effects at each stage. By maintaining consistent preprocessing across all models and systematically introducing changes (tuning, features, re-tuning), the relative contribution of each factor to performance improvement could be quantified.

3.10.2 Reproducibility Measures

Several measures were implemented to ensure complete reproducibility of results:

- **Random State:** All stochastic processes were seeded with `random_state=42`, including train-test splitting, SMOTE sampling, and model initialisation.

Table 3.2: Experimental Pipeline Overview

Step	Process	Output
1	Data Loading	Raw dataset (1,296,675 transactions)
2	Data Cleaning	Removal of unnecessary columns
3	Train-Test Split (65/35)	Training and test sets
4	Feature Engineering (Base)	41 engineered features
5	Feature Scaling	Normalised continuous features
6	SMOTE Application	Balanced training set
7	Stage 1: Baseline Models	3 baseline models (LR, RF, XGB)
8	Stage 2: Hyperparameter Tuning	3 optimised models
9	Feature Engineering (Spending)	11 additional features (52 total)
10	Stage 3: Models with Spending	3 enhanced models
11	Stage 4: Re-optimisation	3 final optimised models
12	Model Comparison	12-model evaluation

- **Version Control:** Python 3.x was used with scikit-learn for preprocessing and modelling, XGBoost for gradient boosting, and imbalanced-learn for SMOTE implementation.
- **Data Leakage Prevention:** Feature engineering statistics were computed exclusively from training data, with scaling parameters fitted on training data only.
- **Consistent Evaluation:** All models were evaluated on the identical, untouched test set to ensure fair comparison.

3.10.3 Ethical Considerations

The dataset used in this research is a publicly available synthetic dataset containing no real customer information. All transaction records, customer identities, and merchant details are simulated, eliminating privacy concerns associated with real financial data.

No personally identifiable information was processed, and the research methodology does not enable identification of real individuals or transactions. The synthetic nature of the data ensures compliance with data protection regulations whilst enabling meaningful research into fraud detection methodologies.

3.11 Summary

This chapter presented the methodological framework adopted to develop, optimise, and evaluate machine learning models for credit card fraud detection. The methodology integrated multiple components within a systematic four-stage experimental pipeline.

The research utilised a large-scale dataset comprising 1,296,675 transactions, with 41 base features engineered from temporal, geographic, amount-based, and behavioural patterns. An additional 11 spending pattern features were subsequently introduced to capture customer spending behaviour, resulting in a total of 51 features. Stratified train-test splitting (65/35) ensured representative partitions, whilst StandardScaler normalisation prepared features for algorithm consumption.

Class imbalance was addressed through SMOTE application to training data, generating synthetic minority samples to improve fraud class representation. Three classification algorithms-Logistic Regression, Random Forest, and XGBoost-were evaluated across four experimental stages: baseline comparison, hyperparameter optimisation, spending feature integration, and re-optimisation.

GridSearchCV with 5-fold stratified cross-validation was employed for hyperparameter tuning, with F1-score as the primary optimisation metric. Model performance was assessed using a comprehensive evaluation framework including accuracy, precision, recall, specificity, F1-score, and ROC-AUC.

Throughout the pipeline, rigorous precautions were implemented to prevent data leakage, including training-only feature aggregation, training-only scaler fitting, and training-only SMOTE application. All stochastic processes were seeded with `random_state=42` to ensure complete reproducibility.

The systematic methodology documented in this chapter provides a robust foundation for the empirical evaluation presented in Chapter 5, where model performance, feature contributions, and comparative analysis are examined in detail.

CHAPTER 4

Data Analysis and Exploration

This chapter presents a systematic examination of the credit card transaction dataset, encompassing exploratory analysis, risk factor identification, and data preprocessing procedures undertaken prior to model development.

4.1 Introduction

A comprehensive understanding of the dataset is essential for the development of effective fraud detection models. This chapter provides a structured exploration of the credit card transaction data, focusing on its structure, statistical characteristics, and inherent challenges that shape modelling strategies.

The analysis begins with data ingestion and preliminary assessment, followed by an examination of numerical and categorical distributions. Risk factor analysis is then conducted to uncover patterns associated with fraudulent behaviour across temporal, transactional, and demographic dimensions. Subsequently, the feature engineering process is outlined, encompassing temporal, geographic, and monetary transformations. Following stratified train-test partitioning, behavioural features are derived exclusively from training data to prevent information leakage. The chapter concludes with encoding of categorical variables, standardisation of numerical features, and application of oversampling techniques to address class imbalance.

4.2 Research Methodology Framework

The methodological framework adopted in this research follows a systematic pipeline designed to address the key challenges in credit card fraud detection. Figure 4.1 illustrates the complete research workflow, encompassing data preprocessing, feature engineering, and model development stages.

The framework commences with data collection, consisting of approximately 1.29 million synthetic credit card transactions, followed by exploratory data analysis (EDA) to examine class distribution, outlier behaviour, and data quality. A risk factor analysis is then performed to uncover temporal, transactional, and demographic patterns associated with fraud.

To prevent information leakage, preprocessing steps follow a deliberate sequence. Static feature engineering capturing temporal, geographic, and monetary characteristics is applied before partitioning. The dataset is then split using stratified sampling in a 65:35 ratio to preserve class proportions across training and test sets. Behavioural features such as customer and merchant level transaction summaries are computed solely on training data and mapped to test instances without using test set statistics.

Categorical variables are encoded, and numerical features are standardised using Z-score transformation to ensure comparability across features. The Synthetic Minority Oversampling Technique (SMOTE) is applied with a sampling strategy of 0.5 to rebalance the training set and enhance the minority class signal.

The modelling phase proceeds through four sequential stages. Stage 1 establishes baseline performance using Logistic Regression, Random Forest, and XGBoost with default hyperparameters. Stage 2 implements hyperparameter tuning via GridSearchCV with 5-fold cross-validation. In Stage 3, spending behaviour features are introduced, and all models are retrained using the expanded feature set. Stage 4 re-optimises the models to assess whether parameter tuning must be adapted when the feature space changes.

All 12 resulting model configurations are evaluated on the same holdout test set to ensure consistent comparison. This framework enables a controlled investigation into how algorithm selection, hyperparameter optimisation, and behavioural feature engineering individually and collectively influence fraud detection performance.

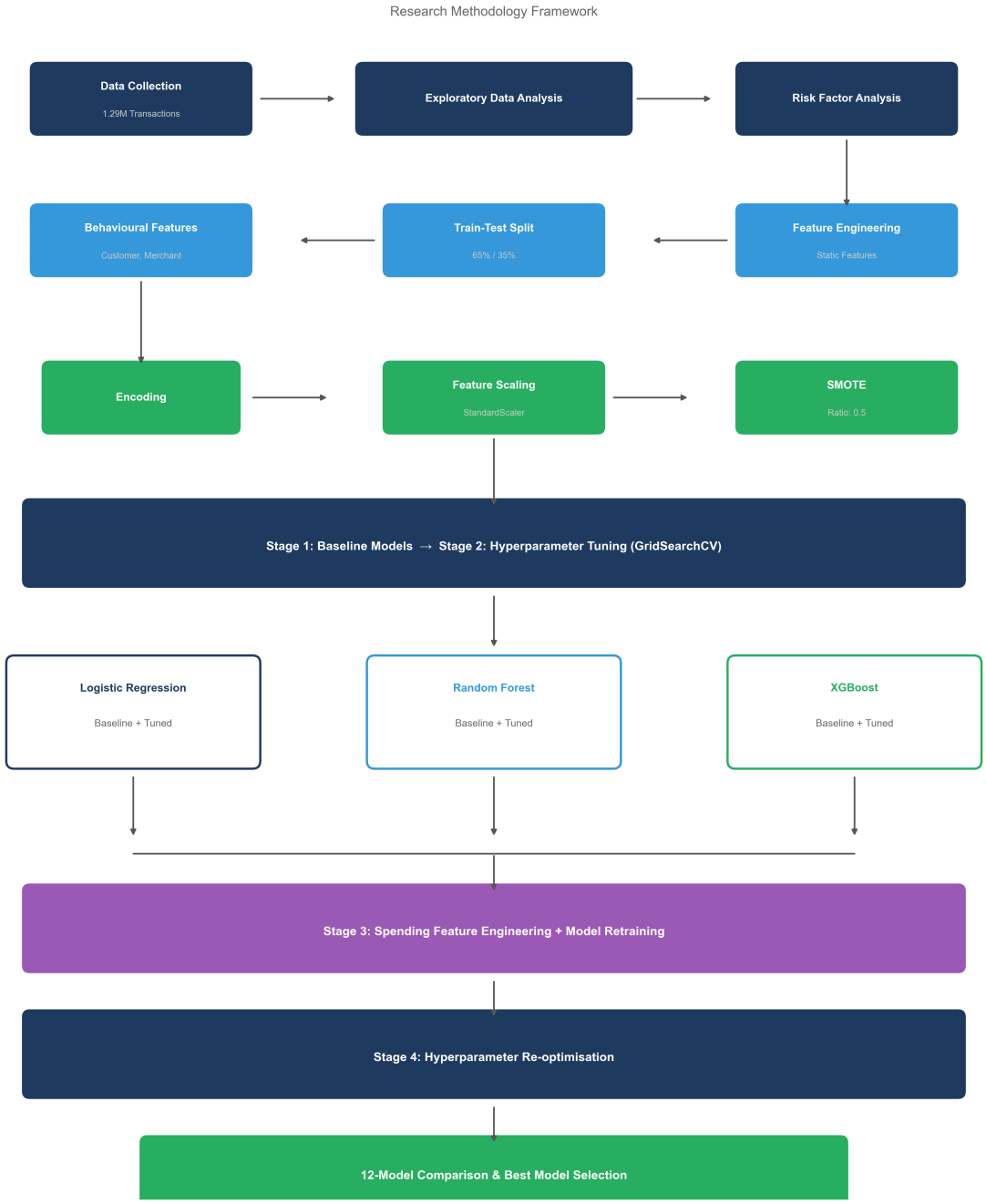

Figure 4.1: Research Methodology Framework for Credit Card Fraud Detection

4.3 Data Loading

The dataset employed in this research was sourced from Kaggle[31], comprising credit card transaction records for fraud detection analysis. Upon ingestion, the dataset was found to contain 1,296,675 transaction records with 24 features, as presented in Table 4.1. The features comprise 12 numerical and 12 categorical variables, with no datetime columns initially recognised.

Table 4.1: Dataset Overview

Dataset Overview

Total Records	1,296,675
Total Features	24
Numeric Columns	12
Categorical Columns	12
DateTime Columns	0

Table 4.2: Column Information (1/2)

Column Information (1/2)

#	Column Name	Data Type	Non-Null	Null %	Unique
1	Unnamed: 0	int64	1,296,675	0.0%	1,296,675
2	trans_date_trans_tim	object	1,296,675	0.0%	1,274,791
3	cc_num	int64	1,296,675	0.0%	983
4	merchant	object	1,296,675	0.0%	693
5	category	object	1,296,675	0.0%	14
6	amt	float64	1,296,675	0.0%	52,928
7	first	object	1,296,675	0.0%	352
8	last	object	1,296,675	0.0%	481
9	gender	object	1,296,675	0.0%	2
10	street	object	1,296,675	0.0%	983
11	city	object	1,296,675	0.0%	894
12	state	object	1,296,675	0.0%	51

Detailed column information is presented in Tables 4.2 and 4.3. The dataset encompasses 983 unique cardholders, 693 merchants, and 14 transaction categories. Notably, the merch_zipcode column exhibits 15.1% missing values, whilst all remaining features contain complete records. The binary target variable is_fraud indicates transaction legitimacy. Table 4.4 displays a sample of the initial records.

Table 4.3: Column Information (2/2)

Column Information (2/2)

#	Column Name	Data Type	Non-Null	Null %	Unique
13	zip	int64	1,296,675	0.0%	970
14	lat	float64	1,296,675	0.0%	968
15	long	float64	1,296,675	0.0%	969
16	city_pop	int64	1,296,675	0.0%	879
17	job	object	1,296,675	0.0%	494
18	dob	object	1,296,675	0.0%	968
19	trans_num	object	1,296,675	0.0%	1,296,675
20	unix_time	int64	1,296,675	0.0%	1,274,823
21	merch_lat	float64	1,296,675	0.0%	1,247,805
22	merch_long	float64	1,296,675	0.0%	1,275,745
23	is_fraud	int64	1,296,675	0.0%	2
24	merch_zipcode	float64	1,100,702	15.1%	28,336

Table 4.4: Dataset Preview (First 5 Records)

Dataset Preview (First 5 Records)

Unnamed: 0	trans_date_tra	cc_num	merchant	... (16 more)	merch_lat	merch_long	is_fraud	merch_zipcode
0	2019-01-01 0	270318618965	fraud_Rippin	...	36.011293	-82.048315	0.0	28705.0
1	2019-01-01 0	630423337322	fraud_Heller	...	49.159046999	-118.186462	0.0	nan
2	2019-01-01 0	388594920576	fraud_Lind-B	...	43.150704	-112.154481	0.0	83236.0
3	2019-01-01 0	353409376434	fraud_Kutch,	...	47.034331	-112.561071	0.0	nan
4	2019-01-01 0	375534208663	fraud_Keelin	...	38.674999	-78.632459	0.0	22844.0

Note: Complete dataset contains 1,296,675 records × 24 features

4.4 Exploratory Data Analysis

4.4.1 Data Quality Assessment

Prior to conducting statistical analysis, an assessment of data quality was undertaken to identify missing values and ensure data integrity. Table 4.5 presents the missing values summary, revealing that the merch_zipcode column contained 195,973 missing entries, accounting for 15.11% of total records. Given that merchant geographic information was already captured through the merch_lat and merch_long coordinates, this column was deemed redundant and subsequently removed from the dataset.

Table 4.5: Missing Values Summary

Missing Values Summary Table

Column Name	Data Type	Missing Count	Missing %
merch_zipcode	float64	195,973	15.11%

Figure 4.2 illustrates the overall data completeness assessment. The analysis confirmed that 99.37% of data cells contained valid values, with missing entries confined exclusively to the merch_zipcode column. This high degree of completeness eliminated the requirement for imputation procedures on remaining features.

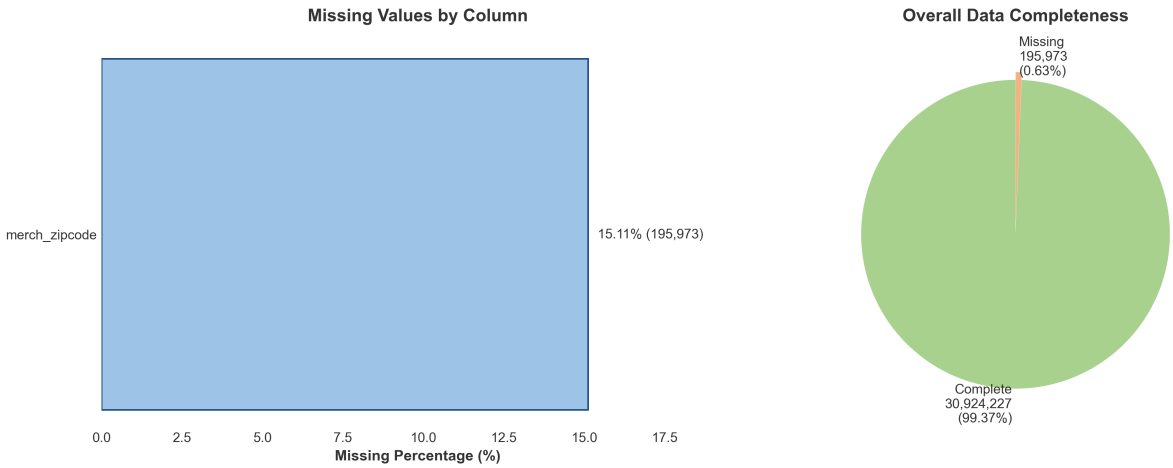

Figure 4.2: Missing Values Analysis and Data Completeness

Following the removal of the redundant column, 22 features remained for subsequent

analysis, as presented in Table 4.6. These features encompass transaction details, customer demographics, geographic coordinates, merchant information, and the target variable is_fraud.

Table 4.6: Remaining Columns After Data Cleaning

Remaining Columns After Cleaning (22 columns)

Row	Column Name	Row	Column Name
1	trans_date_trans_time	12	zip
2	cc_num	13	lat
3	merchant	14	long
4	category	15	city_pop
5	amt	16	job
6	first	17	dob
7	last	18	trans_num
8	gender	19	unix_time
9	street	20	merch_lat
10	city	21	merch_long
11	state	22	is_fraud

4.4.2 Numerical Feature Analysis

Descriptive statistics were computed for all numerical variables to characterise their distributional properties. Table 4.7 presents comprehensive statistics including measures of central tendency, dispersion, and skewness. The transaction amount (amt) variable exhibited substantial positive skewness of 42.28, indicating a heavily right-skewed distribution with the majority of transactions concentrated at lower monetary values. Transaction amounts ranged from $1.00 to $28,948.90, with a mean of $70.35 and median of $47.52. The considerable difference between mean and median further confirms the presence of high-value outliers.

Geographic coordinates confirmed that transactions spanned across the continental United States, with customer latitude ranging from 20.03 to 66.69 and longitude from -165.67 to -67.95. The city_pop variable demonstrated high variability with a skewness of 5.59, reflecting the diverse population sizes of transaction locations ranging from 23 to 2,906,700 inhabitants.

Table 4.7: Numerical Columns - Descriptive Statistics

Numerical Columns - Descriptive Statistics

Column	Count	Mean	Std Dev	Min	Median	Max	Skewness
cc_num	1,296,675	417,192,042,079,726,656.00	1,308,806,447,000,789,248.00	60,416,207,185.00	3,521,417,320,836,166.00	4,992,346,398,065,154,048.00	2.85
amt	1,296,675	70.35	160.32	1.00	47.52	28,948.90	42.28
zip	1,296,675	48,800.67	26,893.22	1,257.00	48,174.00	99,783.00	0.08
lat	1,296,675	38.54	5.08	20.03	39.35	66.69	-0.19
long	1,296,675	-90.23	13.76	-165.67	-87.48	-67.95	-1.15
city_pop	1,296,675	88,824.44	301,956.36	23.00	2,456.00	2,906,700.00	5.59
unix_time	1,296,675	1,349,243,636.73	12,841,278.42	1,325,376,018.00	1,349,249,747.00	1,371,816,817.00	0.00
merch_lat	1,296,675	38.54	5.11	19.03	39.37	67.51	-0.18
merch_long	1,296,675	-90.23	13.77	-166.67	-87.44	-66.95	-1.15
is_fraud	1,296,675	0.01	0.08	0.00	0.00	1.00	13.03

4.4.3 Target Variable Distribution

Analysis of the target variable is_fraud revealed severe class imbalance, representing a fundamental challenge for fraud detection modelling. Table 4.8 presents the fraud distribution summary alongside the class proportion visualisation. Of the 1,296,675 total transactions, 1,289,169 (99.42%) were classified as legitimate whilst only 7,506 (0.58%) were identified as fraudulent. This disparity yields an imbalance ratio of 171.8:1.

Table 4.8: Fraud Distribution Summary

Fraud Distribution Summary

Metric	Count	Percentage
Total Transactions	1,296,675	100.00%
Normal Transactions (Class 0)	1,289,169	99.42%
Fraudulent Transactions (Class 1)	7,506	0.58%
Imbalance Ratio (Normal:Fraud)	171.8:1	-

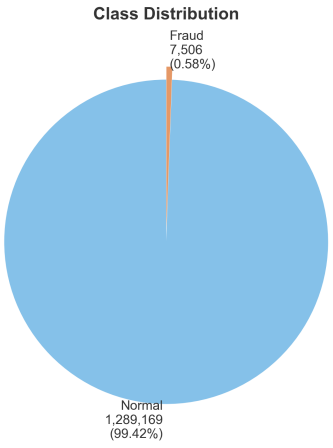

Figure 4.3 provides a visual representation of this extreme class imbalance, demonstrating the magnitude of disproportion between legitimate and fraudulent transactions. Such severe imbalance poses significant challenges for classifier training, as models may exhibit bias towards predicting the majority class to minimise overall error. This characteristic necessitates

the application of specialised resampling techniques, addressed in subsequent sections of this chapter.

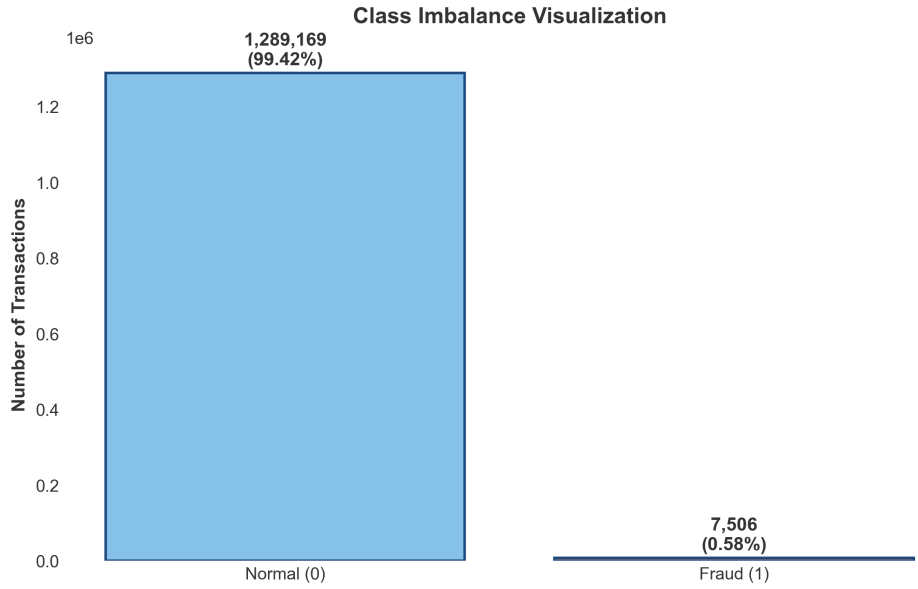

Figure 4.3: Class Imbalance Visualisation

4.4.4 Transaction Amount Analysis

A comparative analysis of transaction amounts between fraudulent and legitimate classes was conducted to identify discriminative patterns. Table 4.9 presents the financial impact summary, revealing substantial differences in monetary characteristics between classes. Fraudulent transactions exhibited significantly elevated values, with a mean of $531.32 compared to $67.67 for legitimate transactions, representing an approximately eight-fold difference. The median values demonstrated an even more pronounced disparity, with fraudulent transactions at $396.50 versus $47.28 for legitimate ones.

The total financial impact of fraudulent activity amounted to $3,988,088.61, representing 4.37% of the total transaction value of $91,222,428.90. Despite constituting merely 0.58% of transaction volume, fraudulent transactions accounted for a disproportionately higher share of monetary losses, underscoring the financial significance of effective fraud detection.

Figure 4.4 illustrates the distributional differences between transaction classes. Legitimate transactions exhibited a right-skewed distribution concentrated predominantly below $300, with a pronounced peak at lower values. Conversely, fraudulent transactions demonstrated a bimodal distribution pattern, with one concentration at lower amounts and a second cluster

Table 4.9: Financial Impact Summary

Financial Impact Summary

Metric	Amount (USD)
Total Transaction Amount	$91,222,428.90
Fraudulent Transaction Amount	$3,988,088.61
Normal Transaction Amount	$87,234,340.29
Average Fraud Transaction	$531.32
Average Normal Transaction	$67.67
Median Fraud Transaction	$396.50
Median Normal Transaction	$47.28
Fraud Loss Percentage	4.37%

extending between $600 and $1,200. This distinctive distributional pattern suggests that transaction amount constitutes a potentially discriminative feature for fraud classification.

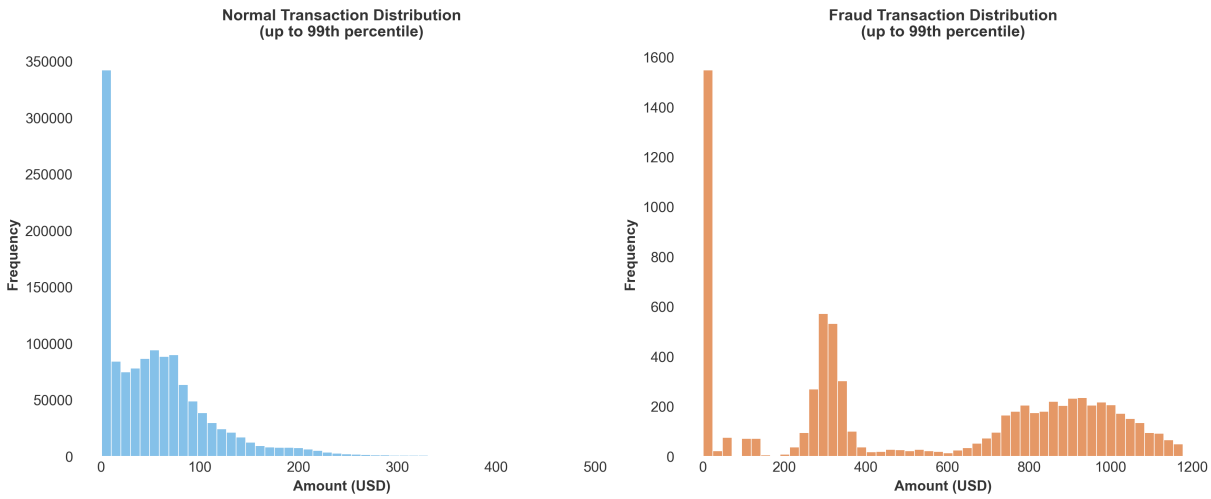

Figure 4.4: Transaction Amount Distribution: Normal vs Fraud

4.4.5 Categorical Feature Analysis

Examination of categorical variables was undertaken to identify patterns across discrete feature categories. Table 4.10 presents the summary of categorical columns, revealing 12 categorical features with varying cardinality. The trans_num column exhibited the highest uniqueness with 1,296,675 distinct values, functioning as a transaction identifier. The category variable comprised 14 unique transaction types, with gas_transport being the most frequent at 131,659 transactions (10.15%). The gender variable contained two categories, with female

customers accounting for 54.74% of transactions.

Table 4.10: Categorical Columns Summary

Categorical Columns Summary

Column Name	Unique Values	Most Common Value	Frequency	Percentage
trans_date_trans_time	1,274,791	2019-04-22 16:02:01	4	0.00%
merchant	693	fraud_Kilback LLC	4,403	0.34%
category	14	gas_transport	131,659	10.15%
first	352	Christopher	26,669	2.06%
last	481	Smith	28,794	2.22%
gender	2	F	709,863	54.74%
street	983	0069 Robin Brooks Apt. 695	3,123	0.24%
city	894	Birmingham	5,617	0.43%
state	51	TX	94,876	7.32%
job	494	Film/video editor	9,779	0.75%
dob	968	1977-03-23	5,636	0.43%
trans_num	1,296,675	0b242abb623afc578575680df30655	1	0.00%

Figure 4.5 presents the fraud rate distribution across all 14 transaction categories. Considerable variation in fraud prevalence was observed, with online transaction categories exhibiting elevated risk. The shopping_net category demonstrated the highest fraud rate at 1.76%, approximately three times the overall average of 0.58%. This was followed by misc_net at 1.45% and grocery_pos at 1.41%. Four categories exceeded the overall fraud rate threshold, classified as high-risk categories. Conversely, health_fitness exhibited the lowest fraud rate at 0.15%, followed by home at 0.16% and food_dining at 0.17%. These findings indicate that online and point-of-sale grocery transactions warrant heightened scrutiny in fraud detection frameworks, whilst certain categories such as health and fitness demonstrate inherently lower fraud susceptibility.

4.5 Risk Factor Analysis

Following the exploratory data analysis, a comprehensive examination of risk factors associated with fraudulent transactions was conducted. This analysis aims to identify patterns and characteristics that distinguish fraudulent activity from legitimate transactions, thereby informing feature engineering decisions and model development strategies.

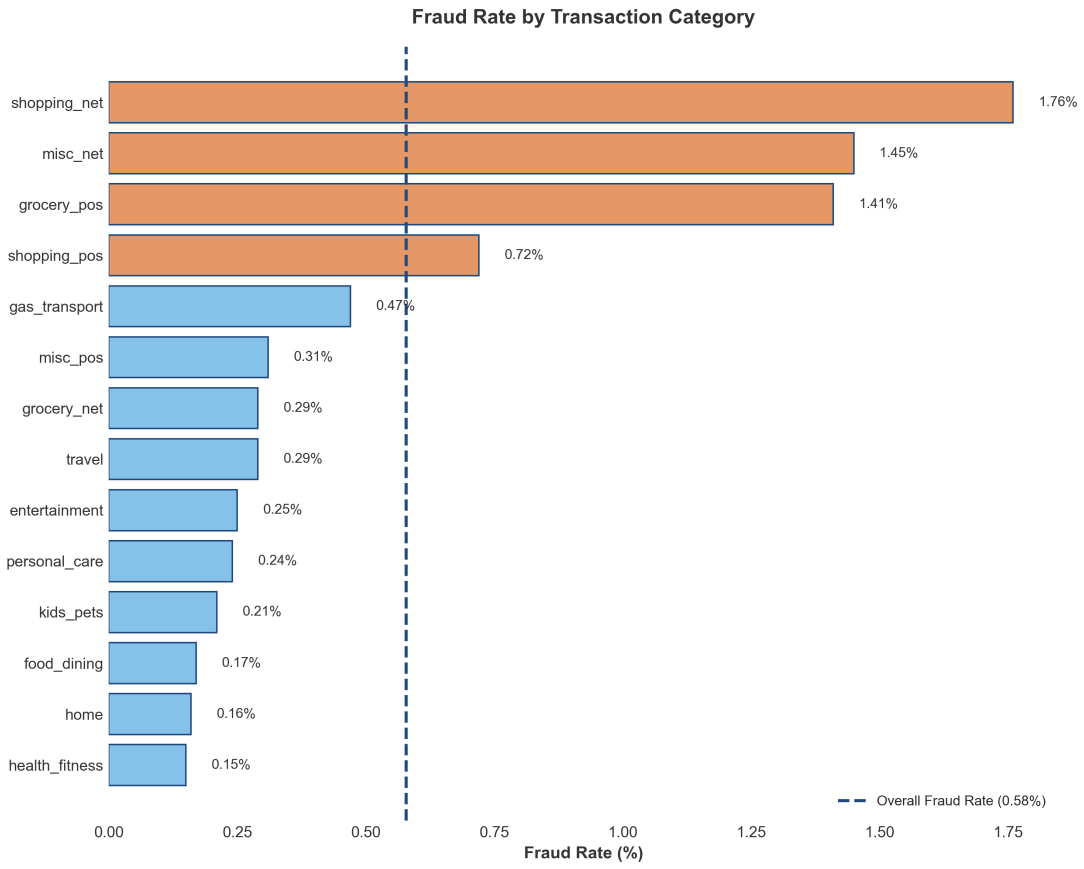

Figure 4.5: Fraud Rate by Transaction Category

4.5.1 Temporal Risk Factors

Temporal patterns represent a critical dimension in fraud detection, as fraudulent behaviour often exhibits distinct timing characteristics. Analysis was conducted across three temporal granularities: hourly, daily, and monthly distributions.

Table 4.11 presents the fraud statistics by hour of day, revealing substantial variation in fraud prevalence across the 24-hour period. The hours between 22:00 and 23:00 exhibited the highest fraud rates at 2.88% and 2.84% respectively, representing approximately five times the overall average fraud rate of 0.58%. Conversely, daytime hours demonstrated significantly lower fraud rates, with 06:00 and 10:00 recording the lowest rates at 0.09%. This pattern suggests that fraudsters preferentially operate during late-night hours when legitimate cardholders are less likely to notice unauthorised transactions and when financial institution monitoring may be reduced.

Table 4.11: Fraud Statistics by Hour of Day: Highest and Lowest Risk Periods

Top 5 Highest Fraud Hours

Hour	Transactions	Frauds	Fraud Rate
22:00	66,982	1,931	2.88%
23:00	67,104	1,904	2.84%
01:00	42,869	658	1.53%
00:00	42,502	635	1.49%
02:00	42,656	625	1.47%

Top 5 Lowest Fraud Hours

Hour	Transactions	Frauds	Fraud Rate
06:00	42,300	40	0.09%
10:00	42,271	40	0.09%
20:00	65,098	62	0.10%
11:00	42,082	42	0.10%
12:00	65,257	67	0.10%

Figure 4.6 illustrates the fraud rate distribution across days of the week. Friday exhibited the highest fraud rate at 0.71%, followed by Thursday at 0.68% and Wednesday at 0.66%. In contrast, Monday demonstrated the lowest fraud rate at 0.46%, with Sunday similarly low at 0.49%. This pattern indicates elevated fraud activity towards the end of the working week, potentially reflecting fraudster behaviour patterns or reduced vigilance among cardholders during pre-weekend periods.

Monthly fraud rate trends are presented in Figure 4.7. February recorded the highest fraud rate at 0.87%, followed by January at 0.81%. The summer months exhibited notably lower fraud rates, with July recording the minimum at 0.38%. This seasonal variation may reflect increased fraudulent activity during post-holiday periods when consumers review

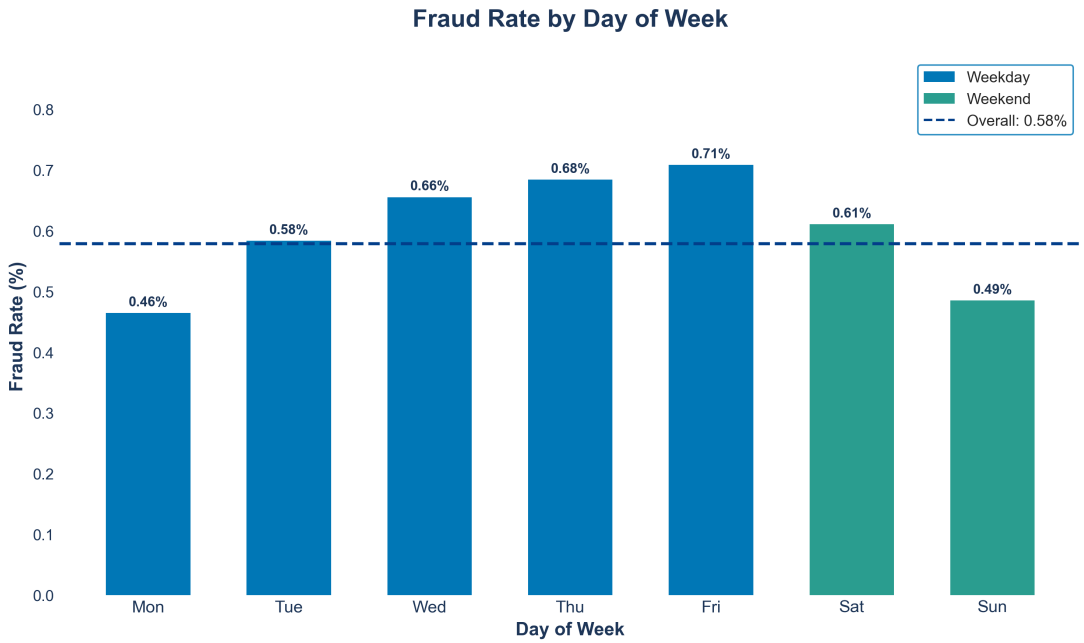

Figure 4.6: Fraud Rate by Day of Week

statements less frequently, or heightened vulnerability following holiday shopping seasons when card details may have been compromised.

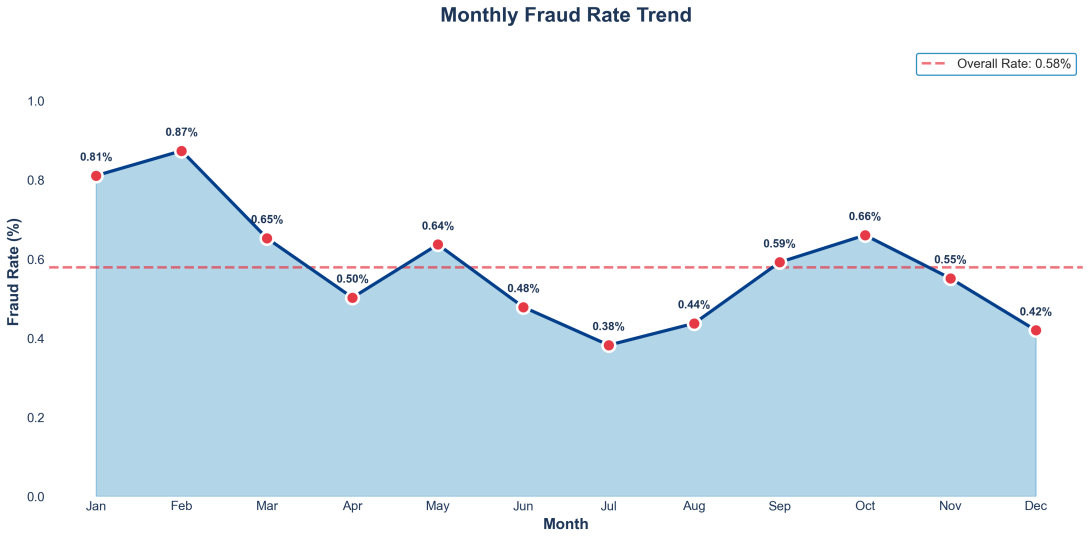

Figure 4.7: Monthly Fraud Rate Trend

4.5.2 Transaction Amount Risk Analysis

Transaction amount emerged as the most discriminative feature for fraud identification. Figure 4.8 presents the fraud rate across six monetary ranges, revealing a pronounced positive

relationship between transaction value and fraud probability. Transactions below $200 exhibited fraud rates substantially below the overall average, with the $50-100 range recording merely 0.01% fraud incidence. However, a dramatic escalation was observed for higher-value transactions: the $200-500 range demonstrated a fraud rate of 4.44%, whilst transactions between $500-1000 exhibited 23.08% fraud prevalence. Most notably, transactions exceeding $1000 recorded a fraud rate of 24.11%, representing a 2,411-fold increase compared to the $50-100 category.

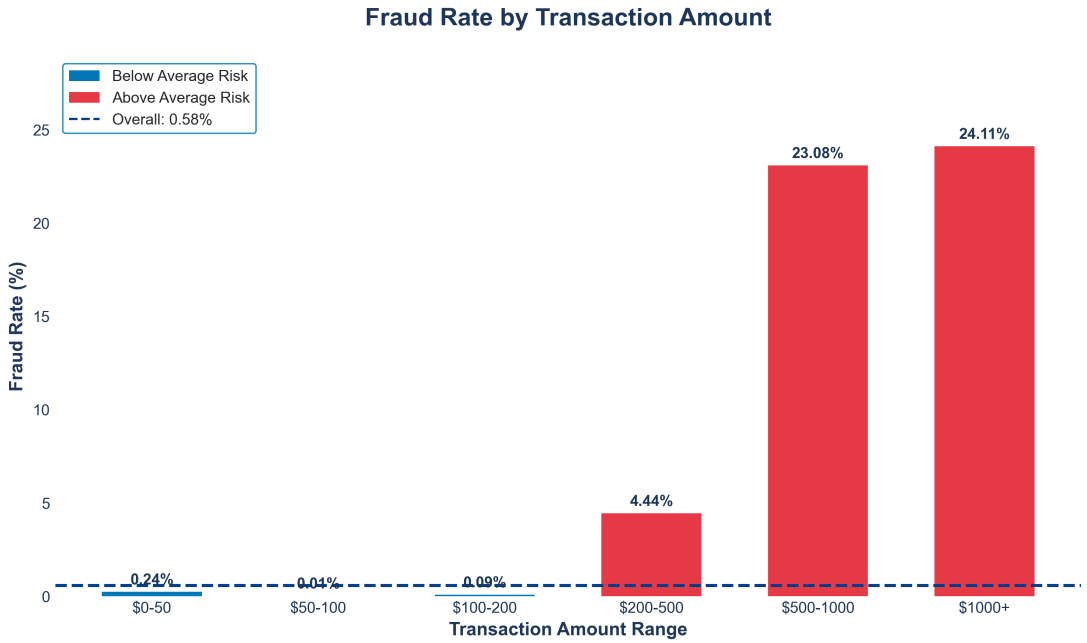

Figure 4.8: Fraud Rate by Transaction Amount Range

This finding aligns with established fraud literature, wherein fraudsters typically attempt to maximise financial gain per fraudulent transaction. The substantial difference in fraud rates between low and high-value transactions underscores the importance of transaction amount as a predictive feature and suggests that amount-based thresholds may serve as effective preliminary fraud indicators.

4.5.3 Categorical and Demographic Risk Factors

Analysis of transaction categories revealed significant variation in fraud susceptibility across merchant types. Figure 4.9 presents the fraud rate distribution across all 14 transaction categories. Online shopping (shopping_net) exhibited the highest fraud rate at 1.76%, approximately three times the overall average. This was followed by miscellaneous online

transactions (misc_net) at 1.45% and grocery point-of-sale (grocery_pos) at 1.41%. Four categories exceeded the overall fraud rate threshold, classified as above-average risk. Conversely, health_fitness demonstrated the lowest fraud rate at 0.15%, followed by home at 0.16%. The elevated fraud rates in online transaction categories reflect the inherent vulnerabilities of card-not-present transactions, where physical card verification is absent.

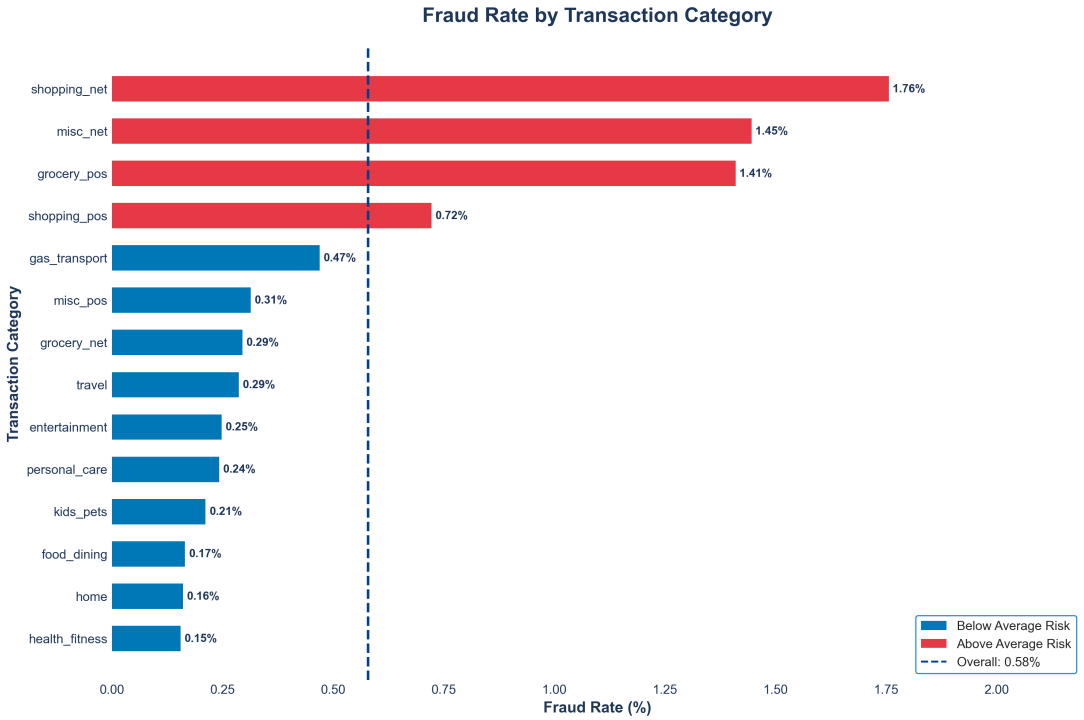

Figure 4.9: Fraud Rate by Transaction Category

Demographic analysis by customer age group is presented in Figure 4.10. Older customers exhibited elevated fraud victimisation rates, with the 65+ age group recording the highest fraud rate at 0.77%, followed by the 56-65 group at 0.76%. The youngest adult category (18-25) also demonstrated above-average fraud rates at 0.62%. Middle-aged customers in the 36-45 range exhibited the lowest fraud rate at 0.43%. This bimodal pattern suggests that both elderly customers, potentially due to reduced technological familiarity, and younger customers, possibly due to increased online activity, face elevated fraud risk.

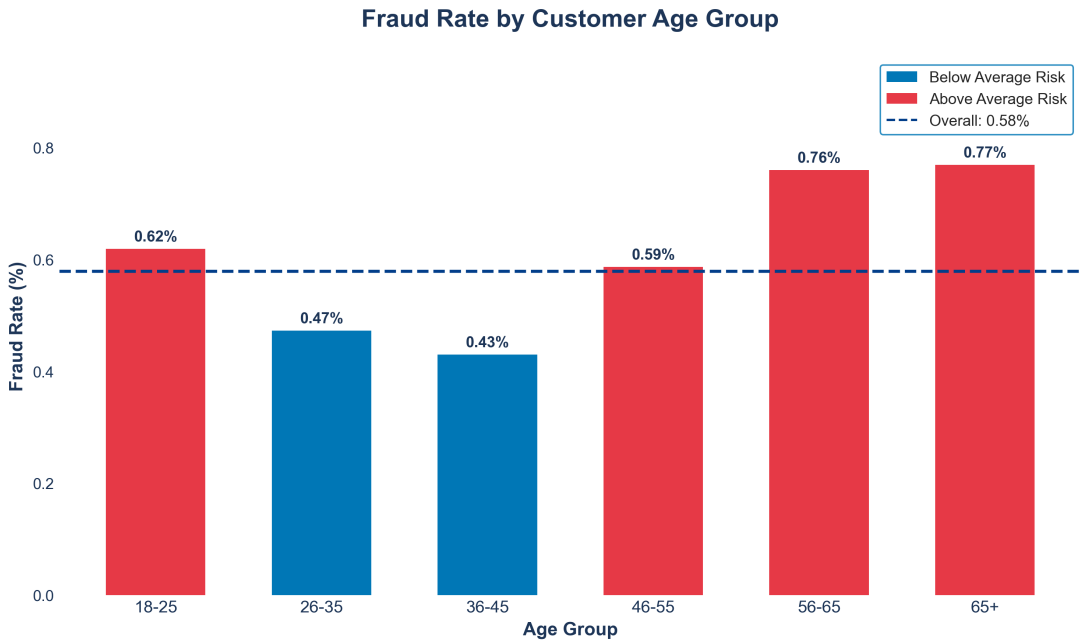

Figure 4.10: Fraud Rate by Customer Age Group

4.5.4 Merchant Risk Analysis

Examination of merchant-level fraud patterns revealed concentration of fraudulent activity among specific merchants. Table 4.12 presents the top 10 merchants ranked by fraud rate, filtered to include only merchants with a minimum of 100 transactions to ensure statistical reliability. Kozey-Boehm recorded the highest fraud rate at 2.57%, followed by Herman, Treutel and Dickens at 2.54%. All merchants in the top 10 exhibited fraud rates exceeding 1.98%, substantially above the overall average. This concentration suggests that certain merchants may be disproportionately targeted by fraudsters or may have compromised payment systems, indicating the potential value of merchant-specific risk scoring in fraud detection frameworks.

Table 4.12: Top 10 Merchants by Fraud Rate (Minimum 100 Transactions)

Top 10 Merchants by Fraud Rate (min 100 transactions)

Rank	Merchant	Transactions	Frauds	Fraud Rate
1	Kozey-Boehm	1,866	48	2.57%
2	Herman, Treutel and Dicke...	1,300	33	2.54%
3	Kerluke-Abshire	1,838	41	2.23%
4	Brown PLC	1,176	26	2.21%
5	Goyette Inc	1,943	42	2.16%
6	Terry-Huel	1,996	43	2.15%
7	Jast Ltd	1,953	42	2.15%
8	Schmeler, Bashirian and P...	1,968	41	2.08%
9	Boyer-Reichert	1,908	38	1.99%
10	Langworth, Boehm and Gulg...	1,969	39	1.98%

4.5.5 Risk Factor Correlation Analysis

To quantify the relationships between identified risk factors and fraudulent activity, a correlation analysis was conducted. Figure 4.11 presents the correlation matrix for key risk indicators. The High Risk Hour variable demonstrated the strongest correlation with fraud at 0.11, confirming the significance of temporal patterns in fraud prediction. High Amount and Online transaction indicators each exhibited correlations of 0.04 with fraud, whilst Weekend and High Distance variables showed negligible correlation (-0.00 and -0.00 respectively).

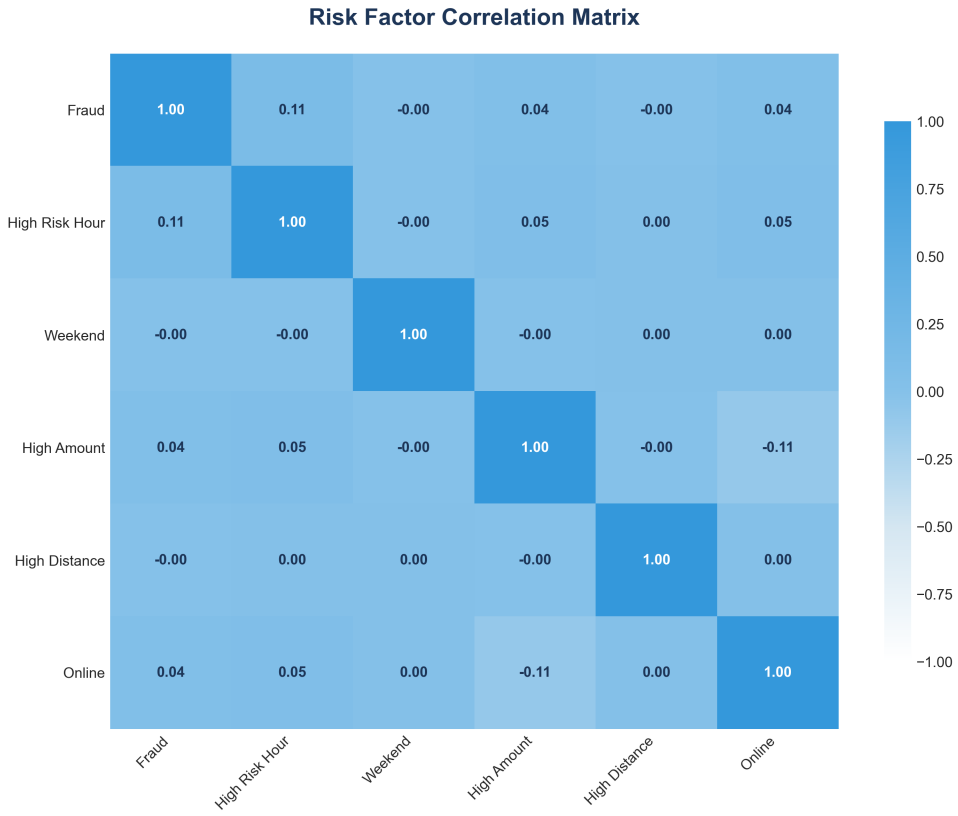

Figure 4.11: Risk Factor Correlation Matrix

The relatively low correlation coefficients, despite substantial differences in fraud rates across categories, reflect the severe class imbalance in the dataset. Nevertheless, these correlations provide guidance for feature prioritisation, with temporal and amount-based features warranting primary attention in model development.

4.5.6 Risk Factor Summary

Table 4.13 consolidates the findings from all risk factor analyses, presenting the highest-risk value and associated fraud rate for each of the 12 factors examined. Transaction amount emerged as the most discriminative factor, with transactions exceeding $1000 exhibiting a 24.11% fraud rate. Hour of day represented the second most significant factor, with 22:00 recording a 2.88% fraud rate. Transaction category demonstrated meaningful variation, with shopping_net transactions at 1.76% fraud rate.

Table 4.13: Risk Factor Summary - All 12 Factors

Risk Factor Summary - All 12 Factors

Risk Factor	Highest Risk Value	Fraud Rate
Hour of Day	22:00	2.88%
Day of Week	Friday	0.71%
Month	2	0.87%
Transaction Amount	$1000+	24.11%
Geographic Distance	50-100 km	0.59%
Transaction Category	shopping_net	1.76%
Gender	Male	0.64%
Age Group	65+	0.77%
City Population	City (100K-500K)	0.68%
State	DE	100.00%
Merchant	Kozey-Boehm	2.57%
Job	Lawyer	5.19%

The risk factor analysis reveals several key insights that inform subsequent feature engineering and modelling decisions. Firstly, transaction amount constitutes the strongest individual predictor of fraud, warranting the creation of amount-based categorical features and threshold indicators. Secondly, temporal features, particularly hour of day, provide substantial discriminative power, supporting the engineering of high-risk hour flags and night transaction indicators. Thirdly, transaction category and customer age demonstrate moderate predictive value, suggesting these demographic and behavioural characteristics should be incorporated into the feature set. Finally, geographic distance exhibited minimal correlation with fraud in this dataset, indicating that distance-based features may contribute less predictive value than initially hypothesised.

4.6 Feature Engineering

Feature engineering constitutes a critical phase in the development of effective fraud detection models, involving the transformation of raw transactional data into meaningful predictors that capture behavioural patterns indicative of fraudulent activity. This section presents the systematic feature construction process undertaken in this research, encompassing temporal, geographic, and monetary transformations, followed by the train-test partitioning strategy employed to ensure methodological rigour.

4.6.1 Temporal Feature Engineering

Temporal features were engineered from the transaction timestamp to capture time-based patterns associated with fraudulent behaviour. The original datetime column was decomposed into multiple granular components, enabling the identification of cyclical patterns across different time horizons. Nine temporal features were created, comprising five numeric features representing hour, day, month, year, and day of week, alongside three binary indicator variables.

Table 4.14 presents the distribution of temporally-derived binary features across the dataset. Weekend transactions accounted for 451,536 records (34.82% of total), whilst night transactions occurring between 22:00 and 05:00 represented 388,916 records (29.99%). The high-risk hour indicator, flagging transactions between 00:00 and 04:00 when fraud rates were observed to be substantially elevated during exploratory analysis, identified 212,659 transactions (16.40%). These binary features enable models to distinguish between high-risk temporal periods and standard operating hours.

Table 4.14: DateTime Feature Statistics

DateTime Feature Statistics

Metric	Count	Percentage
Weekend Transactions	451,536	34.82%
Night Transactions (10PM-5AM)	388,916	29.99%
High-Risk Hour Transactions (12AM-4AM)	212,659	16.40%
Total Transactions	1,296,675	100.00%

4.6.2 Geographic Feature Engineering

Geographic features were constructed to quantify the spatial relationship between cardholder residence and merchant location, a dimension frequently exploited in fraudulent transactions. The Euclidean distance between customer coordinates (lat, long) and merchant coordinates (merch_lat, merch_long) was computed using the Haversine formula, providing a continuous measure of transaction distance. Table 4.15 summarises the four geographic features engineered. The continuous distance variable captures the raw spatial separation, whilst binary indicators flag transactions exceeding the 95th percentile (high_distance) and 75th

percentile (medium_distance) thresholds. Additionally, a categorical region variable was derived based on latitude bands, classifying transactions into South, Central, and North geographic zones to capture potential regional variations in fraud prevalence. Figure 4.12 illustrates the geographic distribution of transactions across the continental United States. The visualisation displays a sample of 25,000 transactions, with normal transactions represented by blue markers and fraudulent transactions highlighted in red. The spatial coverage confirms nationwide transaction activity, with fraudulent transactions distributed across all regions rather than concentrated in specific geographic clusters.

Table 4.15: Geographic Features Created

Geographic Features Created (4 Features)

No	Feature Name	Description	Values	Type
1	distance	Euclidean distance between cardholder	Continuous	Numeric
2	high_distance	Flag for top 5% distances	0=No, 1=Yes	Binary
3	medium_distance	Flag for top 25% distances	0=No, 1=Yes	Binary
4	region	Geographic region based on latitude	South/Central/North	Categorical

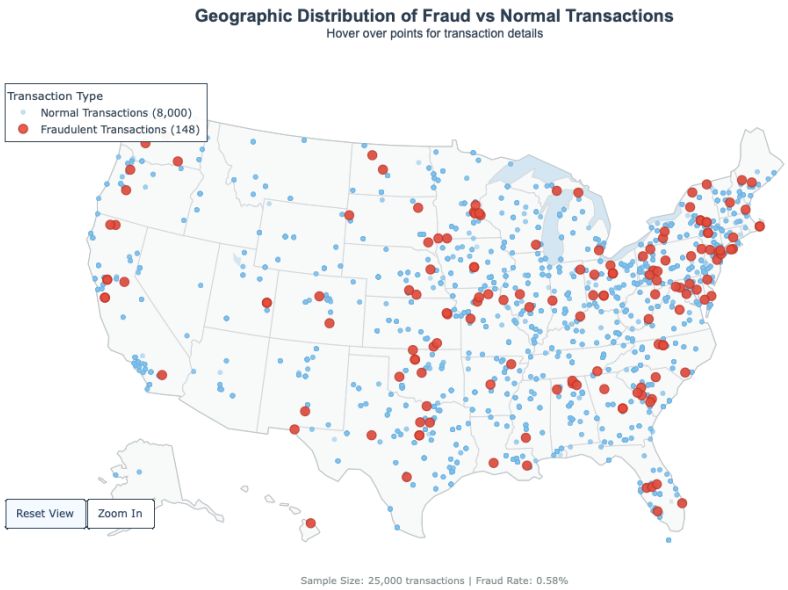

Figure 4.12: Geographic Distribution of Fraud vs Normal Transactions

Figure 4.12 illustrates the geographic distribution of transactions across the continental United States. The visualisation displays a sample of 25,000 transactions, with normal transactions represented by blue markers and fraudulent transactions highlighted in red.

The spatial coverage confirms nationwide transaction activity, with fraudulent transactions distributed across all regions rather than concentrated in specific geographic clusters.

4.6.3 Amount-Based Feature Engineering

Transaction amount features were engineered to capture monetary patterns distinguishing fraudulent from legitimate transactions. Given the substantial positive skewness observed in the amount distribution during exploratory analysis, a logarithmic transformation was applied to normalise the distribution and reduce the influence of extreme values.

Table 4.16: Amount Features Created

Amount Features Created (6 Features)

No	Feature Name	Description	Values	Type
1	amt_log	Log transformed amount: log(1 + amt)	Continuous	Numeric
2	high_amount	Flag for top 5% amounts (>$196)	0=No, 1=Yes	Binary
3	medium_amount	Flag for top 25% amounts (>$83)	0=No, 1=Yes	Binary
4	low_amount	Flag for bottom 25% amounts (<$10)	0=No, 1=Yes	Binary
5	round_amount	Amount divisible by $10	0=No, 1=Yes	Binary
6	very_round_amount	Amount divisible by $100	0=No, 1=Yes	Binary

Table 4.16 details the six amount-based features constructed. The amt_log feature applies the transformation log(1 + amt) to handle zero values whilst compressing the range of high-value transactions. Binary threshold indicators were created based on percentile analysis: high_amount flags transactions exceeding the 95th percentile ($196), medium_amount identifies those above the 75th percentile ($83), and low_amount marks transactions below the 25th percentile ($10). Additionally, round_amount and very_round_amount indicators flag transactions divisible by $10 and $100 respectively, as round-number transactions may exhibit different fraud characteristics.

4.6.4 Train-Test Split

Prior to the construction of behavioural features requiring aggregation across transaction histories, the dataset was partitioned into training and testing subsets. This sequencing is methodologically critical, as computing behavioural statistics on the complete dataset prior to

splitting would introduce information leakage, whereby test set transactions would influence training set feature values, resulting in artificially inflated model performance metrics.

Table 4.17: Train-Test Split Configuration

Train-Test Split Configuration

Parameter	Value	Description
Training Size	65%	Data used to train models
Testing Size	35%	Data used to evaluate models (unseen)
Random State	42	Ensures reproducibility
Stratify	Yes	Maintains fraud ratio in both sets

Table 4.17 presents the train-test split configuration employed. A 65/35 split ratio was selected to balance the requirement for sufficient training data against the need for robust model evaluation. Stratified sampling was applied using the is_fraud target variable to preserve the class distribution across both partitions, ensuring that the severe class imbalance (0.58% fraud rate) was maintained in both training and testing sets. A fixed random state of 42 was specified to ensure reproducibility of results.

Table 4.18: Train-Test Split Summary

Train-Test Split Summary

Dataset	Total Samples	Fraud (1)	Normal (0)	Fraud Rate	Split %
Training Set	842,838	4,879	837,959	0.5789%	65%
Testing Set	453,837	2,627	451,210	0.5788%	35%
Total	1,296,675	7,506	1,289,169	0.5789%	100%

Table 4.18 presents the resulting partition sizes and class distributions. The training set comprises 842,838 transactions containing 4,879 fraudulent cases (0.5789%), whilst the testing set contains 453,837 transactions with 2,627 fraud instances (0.5788%). The near-identical fraud rates across partitions confirm successful stratification.

4.6.5 Behavioural Feature Engineering

Following the train-test partition, behavioural features capturing customer, merchant, and category-level patterns were engineered exclusively from training data. This approach ensures that aggregated statistics reflect only historical transaction behaviour available at prediction time, preventing temporal leakage that would compromise model validity.

Customer behavioural features were computed by aggregating transaction statistics at the cardholder level, including average transaction amount, standard deviation of amounts, transaction frequency, and maximum transaction value. A customer_amount_deviation feature quantifies how each transaction deviates from the customer's typical spending pattern, providing a measure of anomalous behaviour. Binary indicators flag high-activity customers (top 25% by transaction count) and new customers with minimal transaction history.

Merchant behavioural features follow a similar aggregation approach, capturing average transaction amounts and transaction volumes at the merchant level. A merchant_low_activity flag identifies merchants with transaction counts below the 25th percentile, as fraudulent activity may disproportionately target less-established merchants with weaker fraud detection capabilities.

Category-based features incorporate transaction category information, computing average amounts and transaction volumes per category. A category_known_risk binary feature flags transactions in categories identified as high-risk during exploratory analysis, specifically online shopping, miscellaneous online, grocery point-of-sale, and shopping point-of-sale categories.

4.6.6 Spending Pattern Feature Engineering

To enhance the behavioural representation of customer transaction patterns, eleven spending pattern features were engineered. These features capture temporal spending dynamics and deviations from established customer behaviour, providing richer contextual information for fraud detection.

Daily Spending Aggregation

The foundation for spending pattern features was established through aggregation of transaction data at the customer-day level. This process analysed 983 unique customers across 402,949 customer-day observations. Daily spending ranged from $1.00 to $29,347.17, with a mean of $226.39 and median of $138.95. The substantial difference between mean and median, coupled with a standard deviation of $368.20, indicates considerable variability in daily spending patterns. Customers conducted an average of 3.22 transactions per day, establishing a baseline for identifying unusual transaction frequency behaviour.

Feature Construction

Eleven spending pattern features were constructed across three categories:

Rolling Average Features. Short-term and medium-term spending patterns were captured through 7-day and 30-day rolling averages (`rolling_7day_avg`, `rolling_30day_avg`). These windows enable detection of recent behavioural shifts that may accompany fraudulent activity.

Lifetime Baseline Features. Long-term behavioural baselines were established through `lifetime_avg_spending` and `lifetime_avg_transactions`, representing the customer's typical spending amount and transaction frequency across their entire history.

Deviation Scores. Four deviation features quantify how current spending compares to established patterns using normalised ratios:

$$\text{deviation} = \frac{\text{current\_spending} - \text{baseline\_average}}{\text{baseline\_average} + 1} \tag{4.6.1}$$

The features `deviation_from_7day`, `deviation_from_30day`, `deviation_from_lifetime` and `txn_count_deviation` are designed to capture anomalous behavioural deviations across multiple temporal horizons.

Binary Risk Flags. Three binary indicators flag anomalous spending: high spending days (exceeding twice the 7-day average) accounted for 11.43% of observations, extreme spending days (exceeding three times the average) comprised 3.21%, and unusual transaction count days represented 3.40% of the dataset.

Correlation Analysis

Correlation analysis revealed strong relationships between spending deviation features and fraudulent activity. Figure 4.13 presents the Pearson correlation coefficients between spending features and the target variable.

Correlation of Spending Features with Fraud

Spending Feature	Correlation with Fraud
deviation_from_7day	0.215517
deviation_from_30day	0.415258
deviation_from_lifetime	0.515415
high_spending_flag	0.128553
extreme_spending_flag	0.227565
unusual_txn_count	0.063289

Figure 4.13: Correlation of Spending Features with Fraud

The `deviation_from_lifetime` feature exhibited the strongest correlation with fraud (0.515), substantially exceeding the correlations observed for any original feature. The `deviation_from_30day` feature demonstrated a correlation of 0.415, whilst `deviation_from_7day` showed a correlation of 0.216. These strong relationships indicate that spending-deviation features capture fraud-relevant behavioural patterns that are not represented within the original feature set, thereby validating their inclusion in the enhanced modelling framework.

Enhanced Feature Set

Following the addition of spending pattern features, the complete feature set expanded from 41 to 51 variables, representing a 24% increase. The eleven new features comprise eight continuous numeric variables (rolling averages, lifetime baselines, and deviation scores) and three binary risk indicators. All features were computed exclusively from training data to prevent information leakage, with statistics applied to test data through customer identifier lookups.

4.6.7 Feature Summary

Table 4.19 presents the complete summary of engineered features by category. A total of 31 new features were constructed across six feature categories: DateTime (8 features), Geographic (3 features), Amount (6 features), Customer Behaviour (7 features), Merchant Behaviour (4 features), and Category-Based (3 features). The features comprise 16 numeric variables capturing continuous measurements and 15 binary indicators flagging specific conditions or thresholds.

Table 4.19: Engineered Features Summary

Engineered Features Summary (31 Features)

#	Feature Category	Total	Numeric	Binary
1	DateTime	8	5	3
2	Geographic	3	1	2
3	Amount	6	1	5
4	Customer Behavior	7	5	2
5	Merchant Behavior	4	2	2
6	Category-Based	3	2	1
	TOTAL	**31**	**16**	**15**

Table 4.20: Data Transformation Summary

Data Transformation Summary

Dataset	Rows	Original Cols	Final Features	Status
Training Set	842,838	69	41	Ready for scaling
Test Set	453,837	69	41	Ready for scaling

Table 4.20 summarises the data transformation process. Following feature engineering and the removal of non-predictive columns (transaction identifiers, raw timestamps, and personally identifiable information), the training set comprises 842,838 records with 41 final features, whilst the test set contains 453,837 records with identical feature composition. Both datasets are prepared for subsequent scaling operations prior to model training.

The feature engineering process has transformed the original 24-column dataset into a comprehensive 41-feature representation capturing temporal patterns, geographic relationships, monetary characteristics, and behavioural profiles. This enriched feature set provides the foundation for subsequent machine learning model development, enabling algorithms to leverage domain-specific patterns indicative of fraudulent transaction behaviour.

4.7 Feature Scaling and Class Imbalance Handling

Following feature engineering, two critical preprocessing steps were undertaken to prepare the data for machine learning model training: feature scaling to normalise variable ranges, and synthetic oversampling to address the severe class imbalance inherent in fraud

detection datasets. This section details the methodological approaches employed for each transformation.

4.7.1 Feature Scaling

Feature scaling was applied to standardise the range of independent variables, ensuring that features with larger magnitudes do not disproportionately influence model training. This transformation is particularly important for distance-based algorithms and gradient descent optimisation methods, where feature scale directly impacts convergence behaviour and model performance.

Table 4.21: Feature Scaling Configuration

Feature Scaling Configuration

Parameter	Value	Description
Scaler Type	StandardScaler	Transforms to mean=0, std=1
Fit Method	fit_transform()	Applied to training data only
Transform Method	transform()	Applied to testing data
Data Leakage	Prevented □	Test data uses training parameters

Table 4.21 presents the feature scaling configuration employed. StandardScaler was selected as the scaling method, which transforms each feature to have zero mean and unit variance using the formula $z = (x - \mu)/\sigma$, where μ represents the feature mean and σ the standard deviation. The scaler was fitted exclusively on training data using the fit_transform() method, with the learned parameters subsequently applied to test data via the transform() method. This approach prevents data leakage by ensuring that test set statistics do not influence the scaling transformation.

Table 4.22: Before vs After Scaling Comparison

Before vs After Scaling Comparison

Feature	Before Min	Before Max	Before Mean	After Min	After Max	After Mean
amt	1.00	25086.94	70.20	-0.45	164.23	-0.0000
hour	0.00	23.00	12.81	-1.88	1.49	0.0000
distance	0.00	1.41	0.77	-2.69	2.28	-0.0000
city_pop	23.00	2906700.00	88631.13	-0.29	9.35	-0.0000
day_of_week	0.00	6.00	3.07	-1.40	1.33	0.0000
month	1.00	12.00	6.14	-1.51	1.71	-0.0000

Table 4.22 illustrates the effect of standardisation on selected features. Prior to scaling, features exhibited substantially different ranges: transaction amount (amt) ranged from

$1.00 to $25,086.94 with mean $70.20, whilst city population (city_pop) spanned from 23 to 2,906,700 with mean 88,631. Following standardisation, all features were transformed to approximately zero mean, with values expressed in standard deviation units. The amt feature, for example, was rescaled to range from -0.45 to 164.23 standard deviations, whilst city_pop was transformed to range from -0.29 to 9.35 standard deviations.

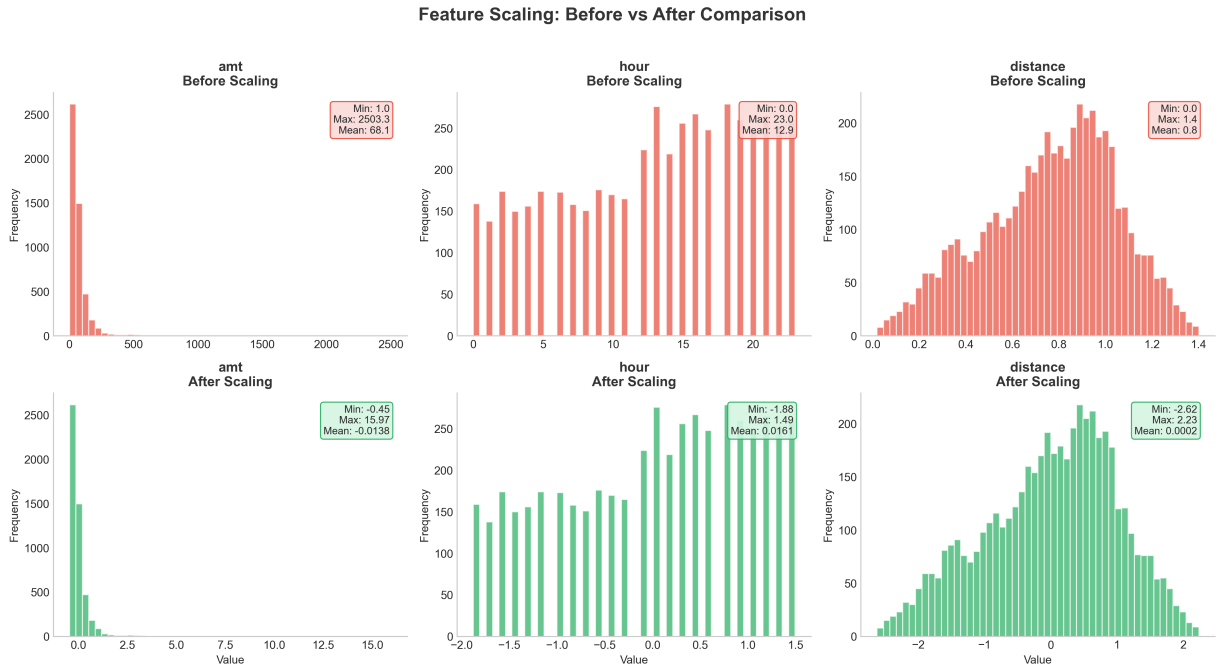

Figure 4.14: Feature Scaling: Before vs After Comparison

Figure 4.14 provides a visual comparison of feature distributions before and after scaling for three representative variables: amt, hour, and distance. The upper panels display original distributions with their native scales, whilst the lower panels show the standardised distributions centred on zero. The transformation preserves the shape of each distribution whilst aligning all features to a common scale, facilitating more effective model training.

Table 4.23: Scaled Dataset Summary

Scaled Dataset Summary

Dataset	Samples	Features	Method	Status
Training Set	842,838	41	fit_transform()	Ready for SMOTE
Testing Set	453,837	41	transform()	Ready for evaluation

Table 4.23 summarises the scaled dataset dimensions. The training set comprises 842,838

samples with 41 features, prepared using fit_transform() and ready for SMOTE application. The testing set contains 453,837 samples with identical feature dimensionality, scaled using the training set parameters and reserved for final model evaluation.

4.7.2 Class Imbalance Handling with SMOTE

The severe class imbalance present in fraud detection datasets poses significant challenges for machine learning algorithms, which tend to favour the majority class and produce models with poor sensitivity to fraudulent transactions. With only 0.58% of transactions being fraudulent (imbalance ratio of 171.7:1), standard classifiers would achieve high accuracy by simply predicting all transactions as legitimate, whilst failing to identify actual fraud cases.

Table 4.24: SMOTE Configuration

SMOTE Configuration

Parameter	Value	Description
Method	SMOTE	Synthetic Minority Over-sampling Technique
Sampling Strategy	0.5	Fraud = 50% of Normal (2:1 ratio)
Random State	42	For reproducibility
Applied To	Training Data Only	Test data remains untouched

Table 4.24 presents the SMOTE (Synthetic Minority Over-sampling Technique) configuration employed to address this imbalance. SMOTE generates synthetic minority class samples by interpolating between existing minority instances in feature space, rather than simply duplicating existing records. A sampling strategy of 0.5 was selected, targeting a fraud-to-normal ratio of 1:2 (50% of the majority class size). This moderate rebalancing approach avoids the potential overfitting associated with full equalisation whilst providing sufficient minority class representation for effective learning. SMOTE was applied exclusively to training data, ensuring that the test set remains representative of the true class distribution encountered in production environments.

Table 4.25: Before vs After SMOTE Comparison

Before vs After SMOTE Comparison

Stage	Normal (0)	Fraud (1)	Total	Ratio
Before SMOTE	837,959	4,879	842,838	171.7:1
After SMOTE	837,959	418,979	1,256,938	2.0:1
Difference	0	+414,100	+414,100	Improved!

Table 4.25 quantifies the effect of SMOTE on class distribution. Prior to oversampling, the training set contained 837,959 normal transactions and only 4,879 fraud cases, yielding an imbalance ratio of 171.7:1. Following SMOTE application, 414,100 synthetic fraud samples were generated, increasing the fraud class to 418,979 instances. The resulting imbalance ratio of 2.0:1 represents a substantial improvement in class balance whilst maintaining computational tractability.

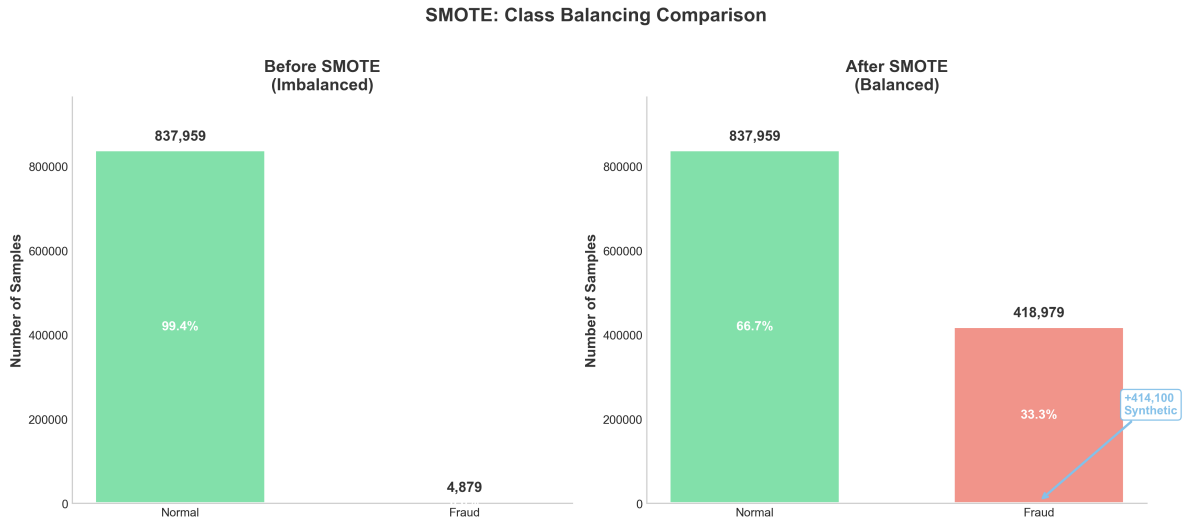

Figure 4.15: SMOTE: Class Balancing Comparison

Figure 4.15 provides a visual comparison of class distributions before and after SMOTE application. The left panel illustrates the original imbalanced distribution, where the fraud class (4,879 samples, 0.58%) is barely visible relative to the normal class (837,959 samples, 99.42%). The right panel demonstrates the rebalanced distribution, with the fraud class now comprising 418,979 samples (33.33%) compared to the unchanged normal class of 837,959 samples (66.67%). The annotation highlights that 414,100 synthetic samples were generated to achieve this balance.

Table 4.26: Variables Created for Modelling

Variables Created for Modeling

Variable	Shape	Description
X_train_balanced	1,256,938 x 41	Balanced training features
y_train_balanced	1,256,938	Balanced training labels
X_test_scaled	453,837 x 41	Test features (unchanged)
y_test	453,837	Test labels (unchanged)

Table 4.26 summarises the final variables prepared for model training. The balanced training set (X_train_balanced, y_train_balanced) comprises 1,256,938 samples with 41 features, representing the SMOTE-augmented dataset. The test set (X_test_scaled, y_test) remains unchanged at 453,837 samples, preserving the original class distribution for unbiased model evaluation. This configuration ensures that models are trained on balanced data to improve minority class detection, whilst being evaluated against realistic imbalanced data representative of operational conditions.

The preprocessing pipeline has successfully transformed the raw transactional data into a modelling-ready format, with standardised features and balanced class representation. The subsequent chapter presents the machine learning model development and evaluation results using this prepared dataset.

4.8 Chapter Summary

This chapter presented a comprehensive analysis of the credit card transaction dataset comprising 1,296,675 records with 24 features. Exploratory data analysis revealed severe class imbalance, with fraudulent transactions representing only 0.58% of records (7,506 cases), yielding an imbalance ratio of 171.8:1. Fraudulent transactions exhibited substantially higher monetary values (mean $531.32 versus $67.67 for legitimate transactions) and demonstrated elevated occurrence rates during night-time hours and within online transaction categories.

Feature engineering transformed the original dataset into a 51-feature representation through the construction of 42 new variables across seven categories: temporal, geographic, amount-based, customer behavioural, merchant behavioural, category-based, and spending pattern features. The spending pattern features, comprising 11 variables capturing deviations from customer baseline spending behaviour, exhibited particularly strong correlations with fraud, with the deviation from lifetime average achieving a correlation coefficient of 0.515. A

stratified 65/35 train-test split was implemented prior to behavioural feature computation to prevent data leakage. StandardScaler normalisation and SMOTE oversampling (0.5 sampling strategy) addressed feature scale disparities and class imbalance respectively. The preprocessed dataset, comprising 1,256,938 balanced training samples and 453,837 test samples, is now prepared for machine learning model development in Chapter 5.

CHAPTER 5

Experiments and Results

This chapter presents the experimental results obtained from the four-stage fraud detection framework evaluating three machine learning algorithms: Logistic Regression, Random Forest, and XGBoost. The experimental design systematically examines: (i) baseline model performance with default hyperparameters, (ii) the impact of hyperparameter optimisation, (iii) the contribution of engineered spending pattern features, and (iv) the necessity of re-optimisation when expanding feature spaces.

Each stage is evaluated using performance metrics appropriate for imbalanced classification problems, including precision, recall, F1 Score, and ROC AUC. Feature importance analysis is employed to interpret model outputs and identify key behavioural features contributing to fraud detection.

The results are presented sequentially across four stages, beginning with baseline evaluation to establish reference performance, followed by hyperparameter tuning to quantify optimisation benefits. The analysis then examines the impact of spending pattern features on model performance, before assessing whether re-optimisation is necessary with expanded feature sets. Within each stage, algorithm-specific results are presented alongside comparative analysis. The chapter concludes with identification of the champion model and discussion of findings in relation to the four research questions guiding this investigation.

5.1 Logistic Regression Performance

This section presents the performance of Logistic Regression across baseline and hyperparameter-tuned configurations using the 41-feature dataset.

5.1.1 Baseline and Tuned Model Comparison

Logistic Regression was first evaluated with default hyperparameters (C=1.0, L2 regularisation) to establish baseline performance, followed by hyperparameter optimisation using GridSearchCV with 5-fold stratified cross-validation. Figure 5.1 presents the confusion matrix for the tuned model alongside a visual performance comparison between baseline and tuned configurations.

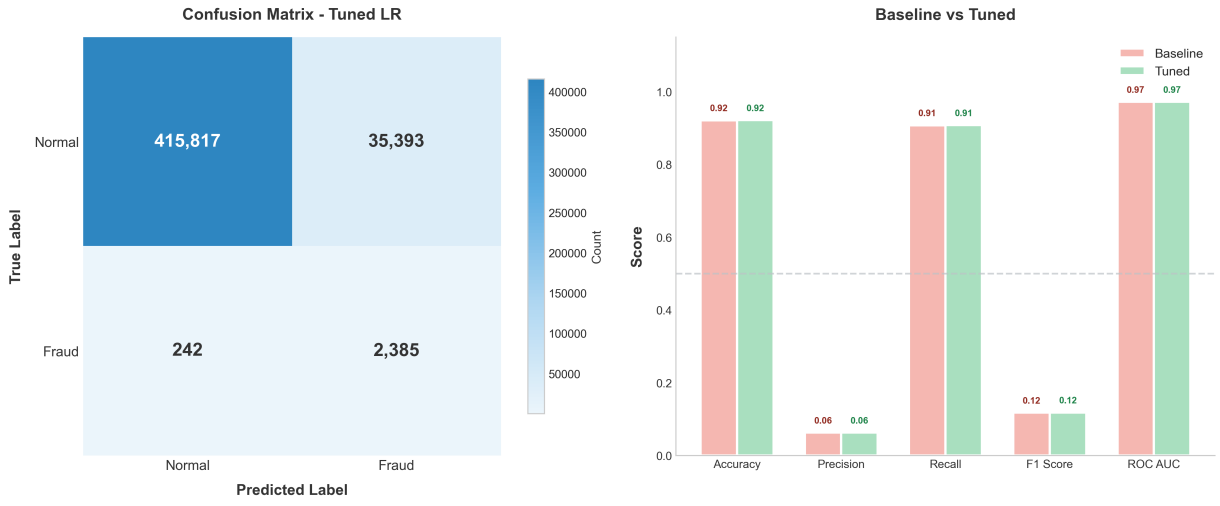

Figure 5.1: Logistic Regression: Confusion Matrix and Baseline vs Tuned Comparison

The confusion matrix reveals that Logistic Regression correctly identified 2,385 of 2,627 fraudulent transactions (90.79% recall) but generated 35,393 false positives, resulting in extremely low precision (6.31%). The model correctly classified 415,817 legitimate transactions whilst misclassifying 242 fraud cases as normal. The bar chart comparison shows virtually identical performance between baseline and tuned models across all metrics.

Table 5.1 presents the detailed performance metrics comparing baseline and tuned Logistic Regression models.

Table 5.1: Logistic Regression: Baseline vs Tuned Performance Metrics

Logistic Regression: Baseline vs Tuned Performance

Metric	Baseline	Tuned	Change
Accuracy	0.9214	0.9215	+0.00%
Precision	0.0631	0.0631	+0.01%
Recall	0.9075	0.9079	+0.04%
F1 Score	0.1180	0.1181	+0.01%
ROC AUC	0.9722	0.9722	+0.00%
PR AUC	0.3493	0.3501	+0.09%

Hyperparameter tuning produced negligible improvement for Logistic Regression. The F1 Score remained virtually unchanged from 0.1180 to 0.1181 (+0.01%), with precision static at 6.31% and recall showing minimal movement from 90.75% to 90.79% (+0.04%). The maximum improvement observed was merely +0.09% for PR AUC, which is statistically insignificant.

This negligible response to hyperparameter optimisation confirms that Logistic Regression's poor fraud detection performance stems from fundamental model limitations rather than suboptimal parameter configuration. The algorithm's linear decision boundary cannot adequately separate the overlapping feature distributions of fraudulent and legitimate transactions. Whilst achieving high recall (90.79%), the model generates approximately 15 false alarms for every correctly identified fraud, rendering it impractical for operational deployment.

5.2 Random Forest Performance

This section presents the performance of Random Forest across baseline and hyperparameter-tuned configurations using the 41-feature dataset.

5.2.1 Baseline and Tuned Model Comparison

Random Forest was first evaluated with default hyperparameters (100 estimators, unlimited depth) to establish baseline performance, followed by hyperparameter optimisation using GridSearchCV with 5-fold stratified cross-validation. The optimal configuration identified was 100 estimators with max_depth of 25, min_samples_split of 10, and min_samples_leaf of 5.

Figure 5.2 presents the confusion matrix for the tuned model alongside a visual performance comparison between baseline and tuned configurations.

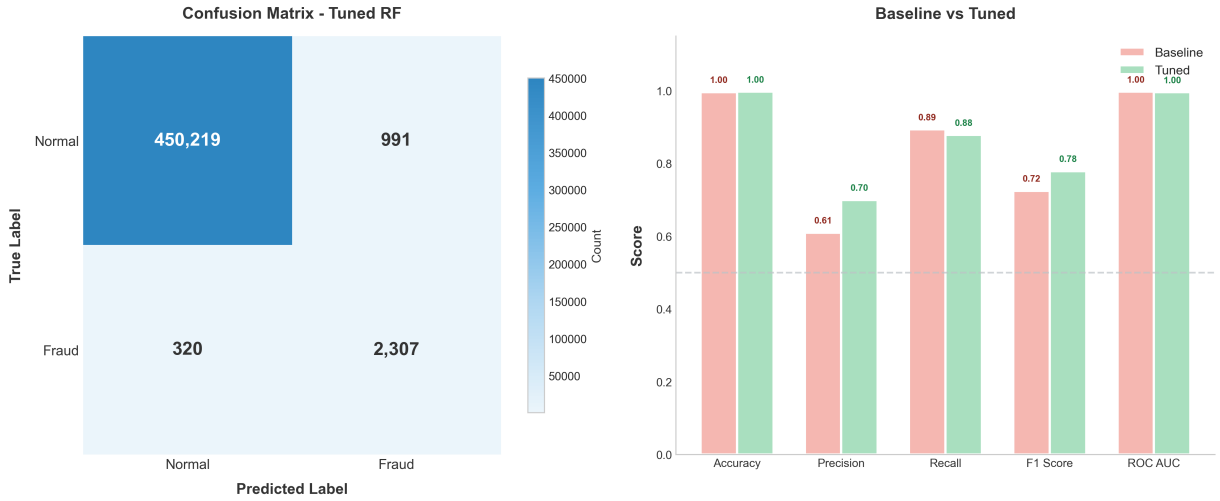

Figure 5.2: Random Forest: Confusion Matrix and Baseline vs Tuned Comparison

The confusion matrix reveals that the tuned Random Forest model correctly identified 2,307 of 2,627 fraudulent transactions (87.82% recall) whilst generating only 991 false positives. The model correctly classified 450,219 legitimate transactions and misclassified 320 fraud cases as normal. The bar chart comparison demonstrates visible improvements in precision and F1 Score following hyperparameter tuning, with recall showing a slight decrease.

Table 5.2 presents the detailed performance metrics comparing baseline and tuned Random Forest models.

Table 5.2: Random Forest: Baseline vs Tuned Performance Metrics

Random Forest: Baseline vs Tuned Performance

Metric	Baseline	Tuned	Change
Accuracy	0.9961	0.9971	**+0.10%**
Precision	0.6098	0.6995	**+8.97%**
Recall	0.8934	0.8782	-1.52%
F1 Score	0.7248	0.7787	**+5.39%**
ROC AUC	0.9970	0.9958	-0.12%
PR AUC	0.8885	0.8999	**+1.14%**

Hyperparameter tuning produced meaningful improvements for Random Forest. The F1 Score improved from 0.7248 to 0.7787, representing a +5.39% relative gain. Most notably, precision increased substantially from 60.98% to 69.95% (+8.97%), indicating significantly

fewer false alarms per detected fraud. This precision gain came at the cost of a marginal recall reduction from 89.34% to 87.82% (-1.52%), reflecting the inherent precision-recall trade-off during optimisation. Accuracy improved slightly from 99.61% to 99.71%, whilst PR AUC increased from 0.8885 to 0.8999 (+1.14%). The ROC AUC showed minimal change, decreasing marginally from 0.9970 to 0.9958 (-0.12%).

Unlike Logistic Regression, Random Forest responded positively to hyperparameter tuning, demonstrating the algorithm's capacity to benefit from systematic parameter optimisation. The ensemble method's ability to capture non-linear feature interactions enables more effective separation between fraudulent and legitimate transactions. However, with precision still below 70%, the tuned Random Forest model would generate approximately three false alarms for every ten fraud alerts, indicating room for further improvement through feature engineering.

5.3 XGBoost Performance

This section presents the performance of XGBoost across baseline and hyperparameter-tuned configurations using the 41-feature dataset.

5.3.1 Baseline and Tuned Model Comparison

XGBoost was first evaluated with default hyperparameters (100 estimators, max_depth of 6, learning_rate of 0.3) to establish baseline performance, followed by hyperparameter optimisation using GridSearchCV with 5-fold stratified cross-validation. The optimal configuration identified was 100 estimators with max_depth of 10, learning_rate of 0.2, subsample of 0.8, and colsample_bytree of 0.8. Figure 5.3 presents the confusion matrix for the tuned model alongside a visual performance comparison between baseline and tuned configurations.

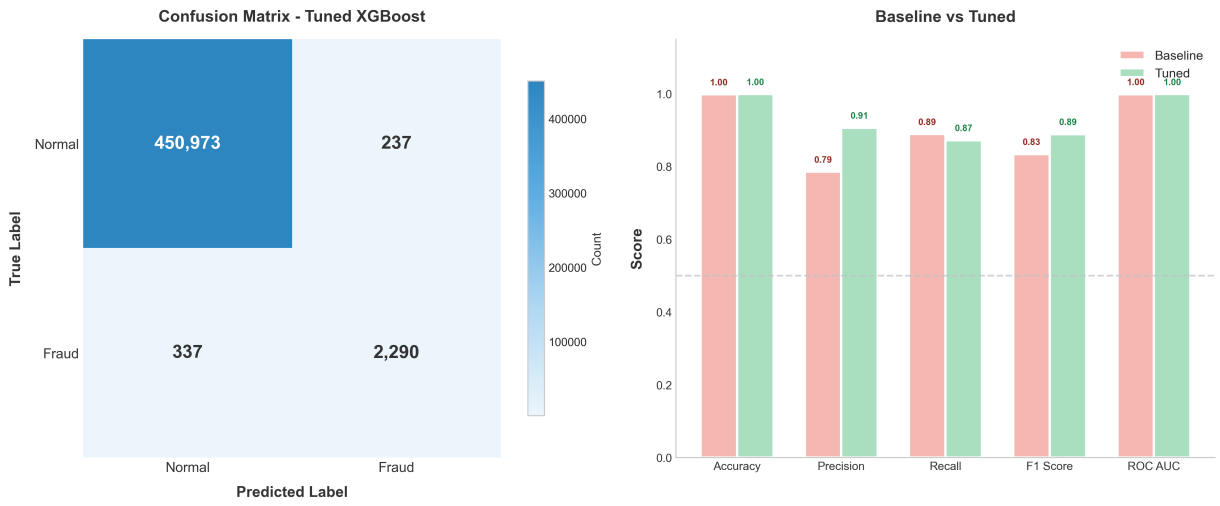

Figure 5.3: XGBoost: Confusion Matrix and Baseline vs Tuned Comparison

The confusion matrix reveals that the tuned XGBoost model correctly identified 2,290 of 2,627 fraudulent transactions (87.17% recall) whilst generating only 237 false positives—substantially fewer than both Logistic Regression (35,393) and Random Forest (991). The model correctly classified 450,973 legitimate transactions and misclassified 337 fraud cases as normal. The bar chart comparison demonstrates clear improvements in precision and F1 Score following hyperparameter tuning.

Table 5.3 presents the detailed performance metrics comparing baseline and tuned XGBoost models.

Table 5.3: XGBoost: Baseline vs Tuned Performance Metrics

XGBoost: Baseline vs Tuned Performance

Metric	Baseline	Tuned	Change
Accuracy	0.9979	0.9987	+0.08%
Precision	0.7851	0.9062	+12.11%
Recall	0.8888	0.8717	-1.71%
F1 Score	0.8338	0.8886	+5.49%
ROC AUC	0.9984	0.9988	+0.04%
PR AUC	0.9277	0.9449	+1.72%

Hyperparameter tuning produced substantial improvements for XGBoost. The F1 Score improved from 0.8338 to 0.8886, representing a +5.49% relative gain—the highest absolute F1 Score achieved among all three algorithms at this stage. Most notably, precision increased

dramatically from 78.51% to 90.62% (+12.11%), crossing the critical 90% threshold and indicating that nine out of ten fraud alerts are genuine. This precision gain came at the cost of a marginal recall reduction from 88.88% to 87.17% (-1.71%), reflecting the precision-recall trade-off during optimisation. Accuracy improved from 99.79% to 99.87%, ROC AUC increased marginally from 0.9984 to 0.9988 (+0.04%), and PR AUC rose from 0.9277 to 0.9449 (+1.72%).

XGBoost demonstrated the strongest response to hyperparameter tuning among all three algorithms, achieving the highest F1 Score (0.8886) and the only precision exceeding 90%. The gradient boosting algorithm's sequential error correction mechanism and built-in regularisation enable effective learning on imbalanced datasets. With precision at 90.62%, the tuned XGBoost model generates fewer than one false alarm for every nine fraud alerts, making it substantially more practical for operational deployment than both Logistic Regression and Random Forest at this stage.

5.4 Logistic Regression with Spending Features Performance

This section presents the performance of Logistic Regression enhanced with spending pattern features across tuned and re-optimised configurations using the expanded 51-feature dataset.

5.4.1 Spending Features and Re-optimisation Comparison

Logistic Regression with spending features was first evaluated using the optimal hyperparameters identified from the 41-feature dataset (C=1.0, L2 regularisation), followed by re-optimisation using GridSearchCV with 5-fold stratified cross-validation on the expanded 51-feature set. The re-optimisation process evaluated regularisation strength and solver configurations to determine whether parameters required adjustment for the new feature space. Figure 5.4 presents the confusion matrix for the re-optimised model alongside a visual performance comparison across all Logistic Regression configurations.

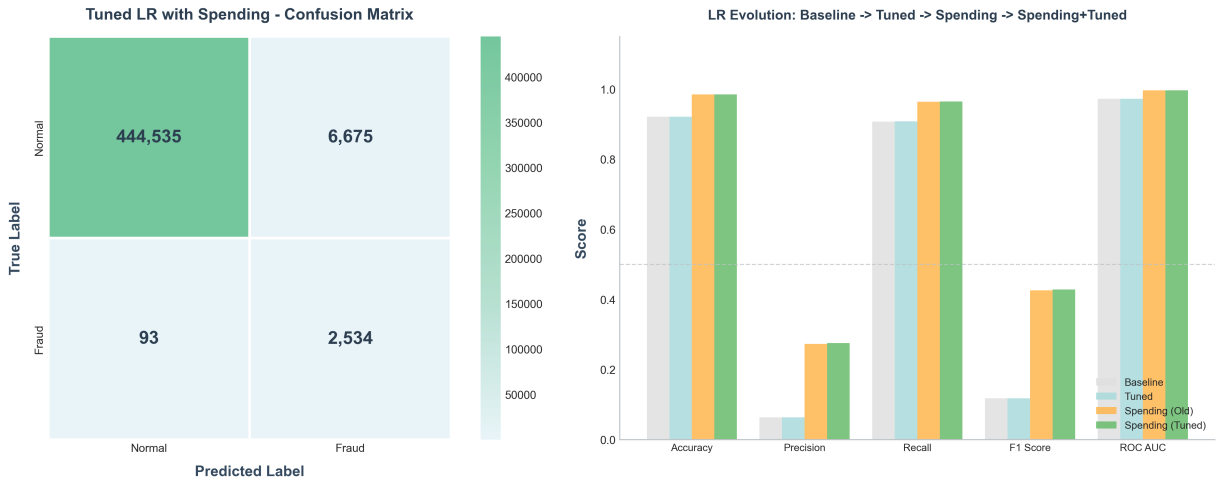

Figure 5.4: Logistic Regression with Spending Features: Confusion Matrix and Performance Evolution

The confusion matrix reveals that the re-optimised Logistic Regression model correctly identified 2,534 of 2,627 fraudulent transactions (96.46% recall) whilst generating 6,675 false positives. The model correctly classified 444,535 legitimate transactions and missed only 93 fraud cases—the lowest false negative count among all Logistic Regression configurations. The bar chart demonstrates the dramatic performance transformation achieved through spending feature integration, with F1 Score improving from below 0.12 (Baseline and Tuned) to above 0.42 (Spending configurations).

Table 5.4 presents the detailed performance metrics comparing all four Logistic Regression configurations.

Table 5.4: Logistic Regression: Complete Performance Comparison Across All Configurations

Comparison: All Logistic Regression Versions

Metric	LR Baseline	LR Tuned	LR+Spending (Old)	LR+Spending (Tuned)
Accuracy	0.9214	0.9215	0.9850	**0.9851**
Precision	0.0631	0.0631	0.2733	**0.2752**
Recall	0.9075	0.9079	0.9642	**0.9646**
F1 Score	0.1180	0.1181	0.4259	**0.4282**
ROC AUC	0.9722	0.9722	0.9964	**0.9965**

Re-optimisation produced marginal improvements for Logistic Regression with spending features. The F1 Score increased from 0.4259 to 0.4282, representing a modest +0.54% relative gain. Precision improved slightly from 27.33% to 27.52% (+0.70%), whilst recall increased

marginally from 96.42% to 96.46% (+0.04%). Accuracy remained essentially unchanged at 98.51%, and ROC AUC improved negligibly from 0.9964 to 0.9965.

The results demonstrate that spending pattern features provide the dominant contribution to Logistic Regression performance improvement. The introduction of spending features transformed F1 Score from 0.1181 to 0.4259—a remarkable 260.6% improvement—whilst re-optimisation contributed only an additional 0.54%. This pattern confirms that for linear classifiers, feature engineering provides the critical performance pathway, enabling effective fraud detection even with linear decision boundaries. The `deviation_from_lifetime` and `deviation_from_30day` features create linearly separable patterns that substantially enhance discrimination between fraudulent and legitimate transactions. However, despite the dramatic relative improvement, Logistic Regression's precision remains below 30%, indicating that non-linear algorithms better capture the complex interactions between spending features and fraud patterns.

5.5 Random Forest with Spending Features Performance

This section presents the performance of Random Forest enhanced with spending pattern features across tuned and re-optimised configurations using the expanded 51-feature dataset.

5.5.1 Spending Features and Re-optimisation Comparison

Random Forest with spending features was first evaluated using the optimal hyperparameters identified from the 41-feature dataset (100 estimators, max_depth=25, min_samples_split=10, min_samples_leaf=5), followed by re-optimisation using GridSearchCV with 5-fold stratified cross-validation on the expanded 51-feature set. The re-optimisation process evaluated tree depth, ensemble size, and leaf configurations to determine whether parameters required adjustment for the new feature space. Figure 5.5 presents the confusion matrix for the re-optimised model alongside a visual performance comparison across all Random Forest configurations.

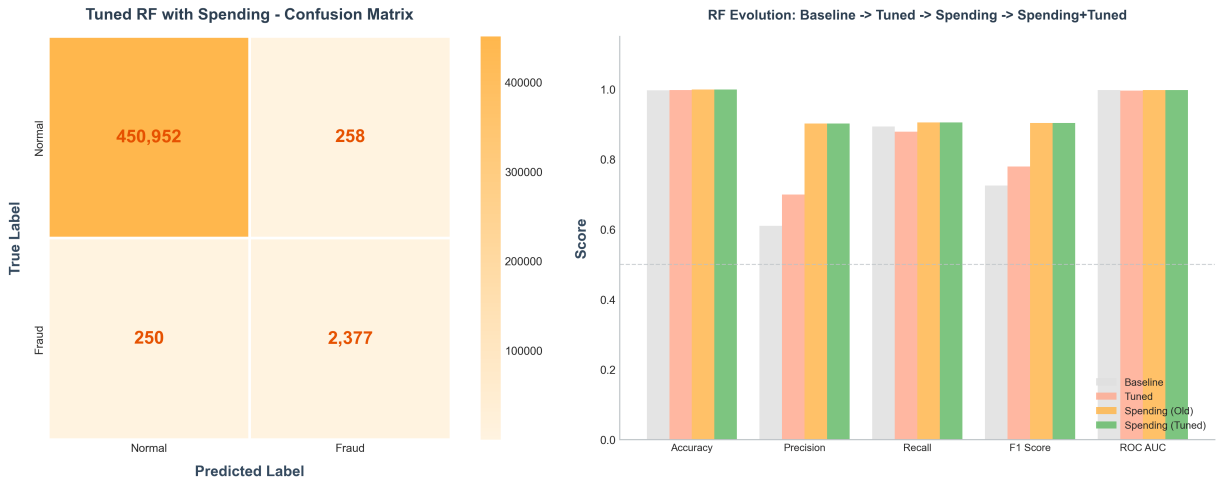

Figure 5.5: Random Forest with Spending Features: Confusion Matrix and Performance Evolution

The confusion matrix reveals that the re-optimised Random Forest model correctly identified 2,377 of 2,627 fraudulent transactions (90.48% recall) whilst generating only 258 false positives. The model correctly classified 450,952 legitimate transactions and missed 250 fraud cases. The bar chart demonstrates substantial performance improvement achieved through spending feature integration, with both precision and recall simultaneously exceeding 90%—a notable outcome that avoids the typical precision-recall trade-off observed in earlier stages.

Table 5.5 presents the detailed performance metrics comparing all four Random Forest configurations.

Table 5.5: Random Forest: Complete Performance Comparison Across All Configurations

Comparison: All Random Forest Versions

Metric	RF Baseline	RF Tuned	RF+Spending (Old)	RF+Spending (Tuned)
Accuracy	0.9961	0.9971	**0.9989**	0.9989
Precision	0.6098	0.6995	**0.9021**	0.9021
Recall	0.8934	0.8782	**0.9048**	0.9048
F1 Score	0.7248	0.7787	**0.9035**	0.9035
ROC AUC	0.9970	0.9958	**0.9974**	0.9974

Re-optimisation produced no improvement for Random Forest with spending features. The F1 Score remained identical at 0.9035, with precision unchanged at 90.21% and recall static at 90.48%. Accuracy and ROC AUC similarly showed no variation, both remaining at

0.9989 and 0.9974 respectively. This finding indicates that the hyperparameters optimised on the 41-feature dataset transferred perfectly to the expanded 51-feature space, eliminating the computational burden of re-tuning.

The results demonstrate that spending pattern features provide the dominant contribution to Random Forest performance improvement. The introduction of spending features transformed F1 Score from 0.7787 to 0.9035—a substantial 16.02% relative improvement—whilst re-optimisation contributed nothing additional. Precision increased dramatically from 69.95% to 90.21%, crossing the critical 90% threshold for the first time, indicating that nine out of ten fraud alerts generated by the model are genuine. Simultaneously, recall improved from 87.82% to 90.48%, demonstrating that spending features enhance both metrics concurrently rather than forcing a trade-off. The false positive count reduced from 991 (tuned model without spending features) to merely 258, representing a 74% reduction in false alarms. This pattern confirms that ensemble methods effectively leverage the discriminative power of behavioural spending deviation features, with the `deviation_from_lifetime` and `deviation_from_30day` features enabling substantially improved separation between fraudulent and legitimate transactions.

5.6 XGBoost with Spending Features Performance

This section presents the performance of XGBoost enhanced with spending pattern features across tuned and re-optimised configurations using the expanded 51-feature dataset.

5.6.1 Spending Features and Re-optimisation Comparison

XGBoost with spending features was first evaluated using the optimal hyperparameters identified from the 41-feature dataset (100 estimators, max_depth=10, learning_rate=0.2, subsample=0.8, colsample_bytree=0.8), followed by re-optimisation using GridSearchCV with 5-fold stratified cross-validation on the expanded 51-feature set. The re-optimisation process evaluated tree depth, learning rate, and regularisation configurations to determine whether parameters required adjustment for the new feature space. Figure 5.6 presents the confusion matrix for the re-optimised model alongside a visual performance comparison across all XGBoost configurations.

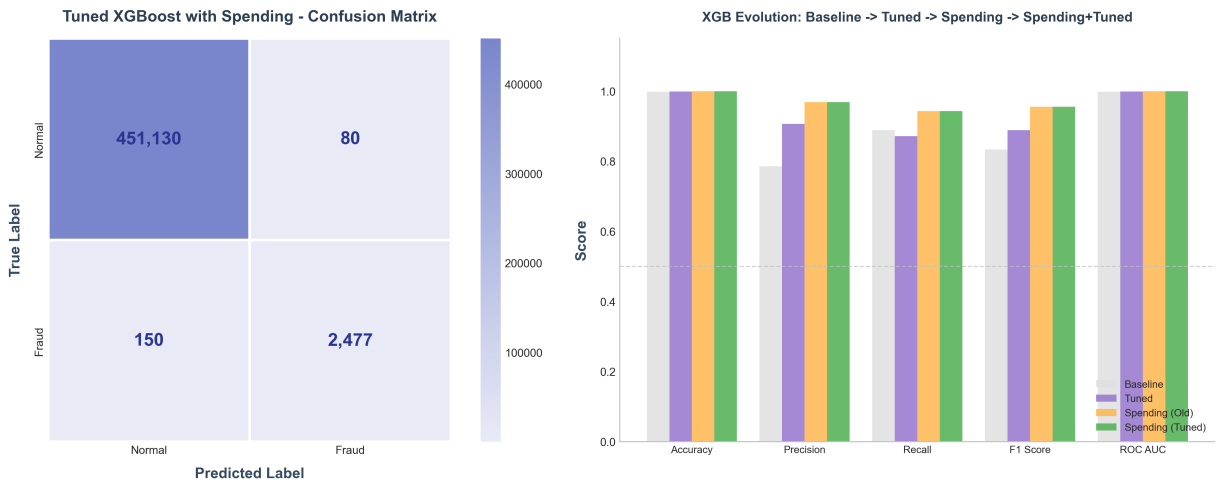

Figure 5.6: XGBoost with Spending Features: Confusion Matrix and Performance Evolution

The confusion matrix reveals that the re-optimised XGBoost model correctly identified 2,477 of 2,627 fraudulent transactions (94.29% recall) whilst generating only 80 false positives—the lowest false positive count among all twelve model configurations. The model correctly classified 451,130 legitimate transactions and missed 150 fraud cases. The bar chart demonstrates consistent performance improvement across all four experimental stages, with the spending-enhanced configurations achieving the highest scores across all metrics.

Table 5.6 presents the detailed performance metrics comparing all four XGBoost configurations.

Table 5.6: XGBoost: Complete Performance Comparison Across All Configurations

Comparison: All XGBoost Versions

Metric	XGB Baseline	XGB Tuned	XGB+Spending (Old)	XGB+Spending (Tuned)
Accuracy	0.9979	0.9987	**0.9995**	0.9995
Precision	0.7851	0.9062	**0.9687**	0.9687
Recall	0.8888	0.8717	**0.9429**	0.9429
F1 Score	0.8338	0.8886	**0.9556**	0.9556
ROC AUC	0.9984	0.9988	**0.9999**	0.9999

Re-optimisation produced no improvement for XGBoost with spending features. The F1 Score remained identical at 0.9556, with precision unchanged at 96.87% and recall static at 94.29%. Accuracy and ROC AUC similarly showed no variation, both remaining at 0.9995 and 0.9999 respectively. This finding, consistent with the Random Forest results, confirms that well-tuned hyperparameters generalise effectively to expanded feature spaces without

requiring computationally expensive re-optimisation.

The results demonstrate that spending pattern features provide the dominant contribution to XGBoost performance improvement. The introduction of spending features transformed F1 Score from 0.8886 to 0.9556—a 7.54% relative improvement—whilst re-optimisation contributed nothing additional. Precision increased substantially from 90.62% to 96.87%, indicating that approximately 97 out of every 100 fraud alerts generated by the model are genuine. Recall improved from 87.17% to 94.29%, representing a 7.12 percentage point increase in fraud detection capability. The false positive count reduced dramatically from 237 (tuned model without spending features) to merely 80, representing a 66% reduction in false alarms. The ROC AUC of 0.9999 indicates near-perfect discrimination capability between fraudulent and legitimate transactions.

This configuration, designated as the champion model (M9), achieved the highest performance across all twelve model configurations evaluated in this research. The combination of XGBoost's gradient boosting architecture with behavioural spending deviation features enables exceptional fraud detection performance, balancing high precision (96.87%) with strong recall (94.29%). The `deviation_from_30day` and `deviation_from_lifetime` features contribute approximately 56% of total feature importance, confirming that customer-specific spending anomalies provide the most discriminative signal for fraud identification.

5.7 Chapter Summary

This chapter presented a comprehensive evaluation of twelve model configurations across four experimental stages, systematically examining the contributions of hyperparameter optimisation and behavioural feature engineering to credit card fraud detection performance.

The baseline evaluation established XGBoost as the superior algorithm with an F1 Score of 0.8338, followed by Random Forest (0.7248) and Logistic Regression (0.1180). Hyperparameter optimisation produced an average improvement of 4.70% across algorithms, with ensemble methods benefiting substantially whilst Logistic Regression showed negligible response (+0.08%). The introduction of eleven spending pattern features delivered dramatic performance gains, with Logistic Regression improving by 260.6%, Random Forest by 16.02%, and XGBoost by 7.54%. Re-optimisation on the expanded 51-feature set proved unnecessary, with both Random Forest and XGBoost achieving identical performance to their Stage 3

configurations.

The champion model, XGBoost with Spending Features (M9), achieved an F1 Score of 0.9556, precision of 96.87%, and recall of 94.29%, correctly identifying 2,477 of 2,627 fraudulent transactions whilst generating only 80 false positives. The quantification of improvement sources reveals that spending pattern features account for 95.1% of total performance gains, confirming that behavioural feature engineering provides the dominant contribution to fraud detection effectiveness.

CHAPTER 6

Discussion

This chapter synthesises the experimental findings presented in Chapter 5, addresses the four research questions guiding this investigation, compares results with existing literature, and discusses the practical implications for fraud detection system deployment.

6.1 Baseline and Tuned Model Performance

The first two stages of the experimental framework established reference performance levels through baseline evaluation and hyperparameter optimisation. Figure 6.1 presents the comprehensive performance comparison across all evaluation metrics for the six model configurations evaluated in Stages 1 and 2.

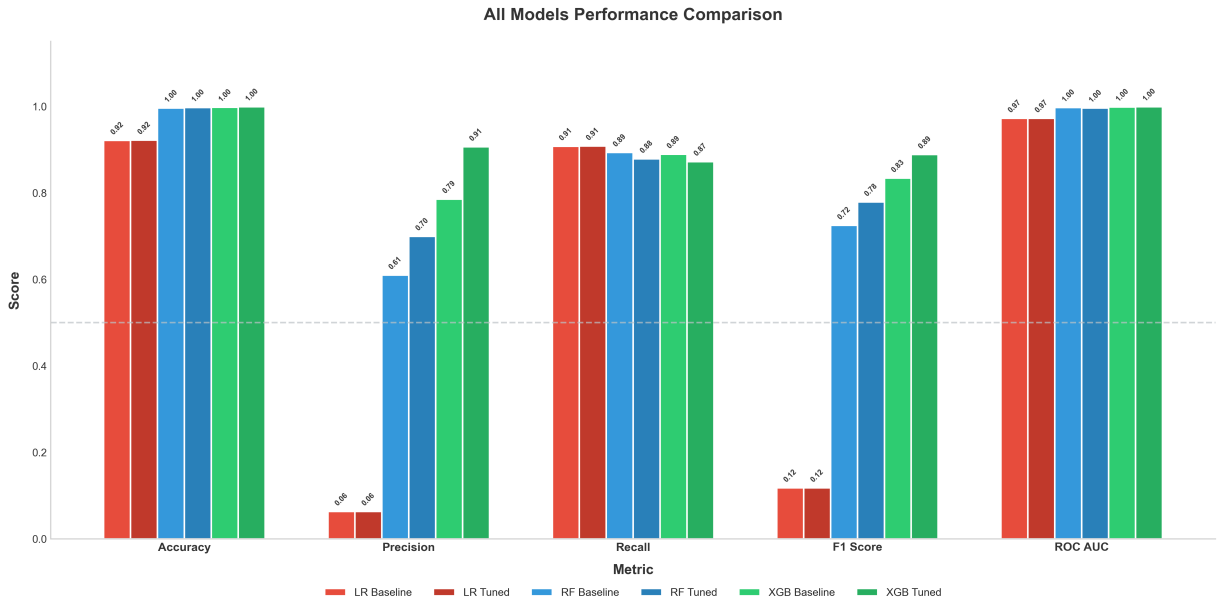

Figure 6.1: All Models Performance Comparison: Baseline vs Tuned Configurations (Stages 1 and 2)

The comparison reveals several important patterns regarding algorithm performance prior to spending feature integration. Accuracy remained consistently high across all models (0.92–1.00), demonstrating that this metric provides limited discriminative value for imbalanced fraud detection tasks where simply predicting all transactions as legitimate would achieve 99.42% accuracy. The critical differentiation emerges in precision and F1 Score metrics, which better capture fraud detection effectiveness.

Logistic Regression exhibited extremely low precision (0.06) regardless of hyperparameter tuning, generating approximately 15 false alarms for every correctly identified fraud. Despite achieving the highest recall (0.91), this precision deficit renders Logistic Regression impractical for operational deployment at this stage. Random Forest demonstrated meaningful precision improvement from 0.61 (baseline) to 0.70 (tuned), representing an 8.97% gain through hyperparameter optimisation. XGBoost achieved the most substantial improvement, with precision increasing from 0.79 (baseline) to 0.91 (tuned), crossing the critical 90% threshold and indicating that nine out of ten fraud alerts are genuine.

The F1 Score comparison confirms XGBoost's superiority, improving from 0.83 (baseline) to 0.89 (tuned)—the highest among all Stage 1 and 2 configurations. Random Forest achieved F1 Scores of 0.72 (baseline) and 0.78 (tuned), whilst Logistic Regression remained static at 0.12 for both configurations. ROC AUC values exceeded 0.97 for all models, with XGBoost

achieving 1.00, indicating strong discrimination capability across all classification thresholds.

6.2 Best Model Performance Prior to Feature Engineering

Figure 6.2 presents the F1 Score comparison across all Stage 1 and 2 models alongside the detailed performance profile of the best-performing model (XGBoost Tuned) prior to spending feature integration.

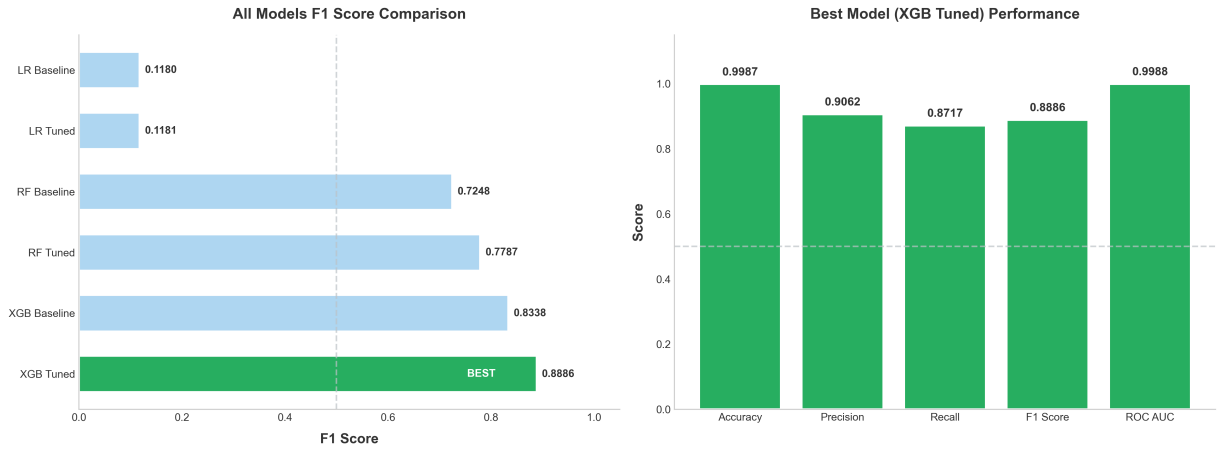

Figure 6.2: F1 Score Comparison and Best Model (XGBoost Tuned) Performance Profile

The left panel clearly illustrates the performance hierarchy established through Stages 1 and 2. XGBoost Tuned achieved the highest F1 Score (0.8886), marked as "BEST" among the six configurations, followed by XGBoost Baseline (0.8338), RF Tuned (0.7787), RF Baseline (0.7248), and both Logistic Regression configurations (0.1181 and 0.1180). The substantial gap between XGBoost and other algorithms confirms the gradient boosting framework's effectiveness for imbalanced fraud detection tasks.

The right panel presents the comprehensive performance profile of XGBoost Tuned, demonstrating balanced excellence across all metrics. Accuracy reached 0.9987, precision achieved 0.9062, recall attained 0.8717, F1 Score measured 0.8886, and ROC AUC achieved 0.9988. This configuration correctly identified 87.17% of fraudulent transactions whilst maintaining 90.62% precision, establishing a strong foundation for subsequent enhancement through spending pattern features.

The performance differential between XGBoost and Random Forest (F1 difference of 0.1099) substantially exceeds the gap between Random Forest and Logistic Regression when

accounting for the latter's fundamental limitations. This finding suggests that algorithm selection within ensemble methods provides meaningful but secondary benefit compared to the choice between linear and non-linear classifiers.

6.3 Complete Model Ranking Across All Stages

Following spending feature integration in Stages 3 and 4, the complete ranking of all twelve model configurations was established. Figure 6.3 presents the comprehensive ranking sorted by F1 Score, enabling direct comparison across all experimental stages.

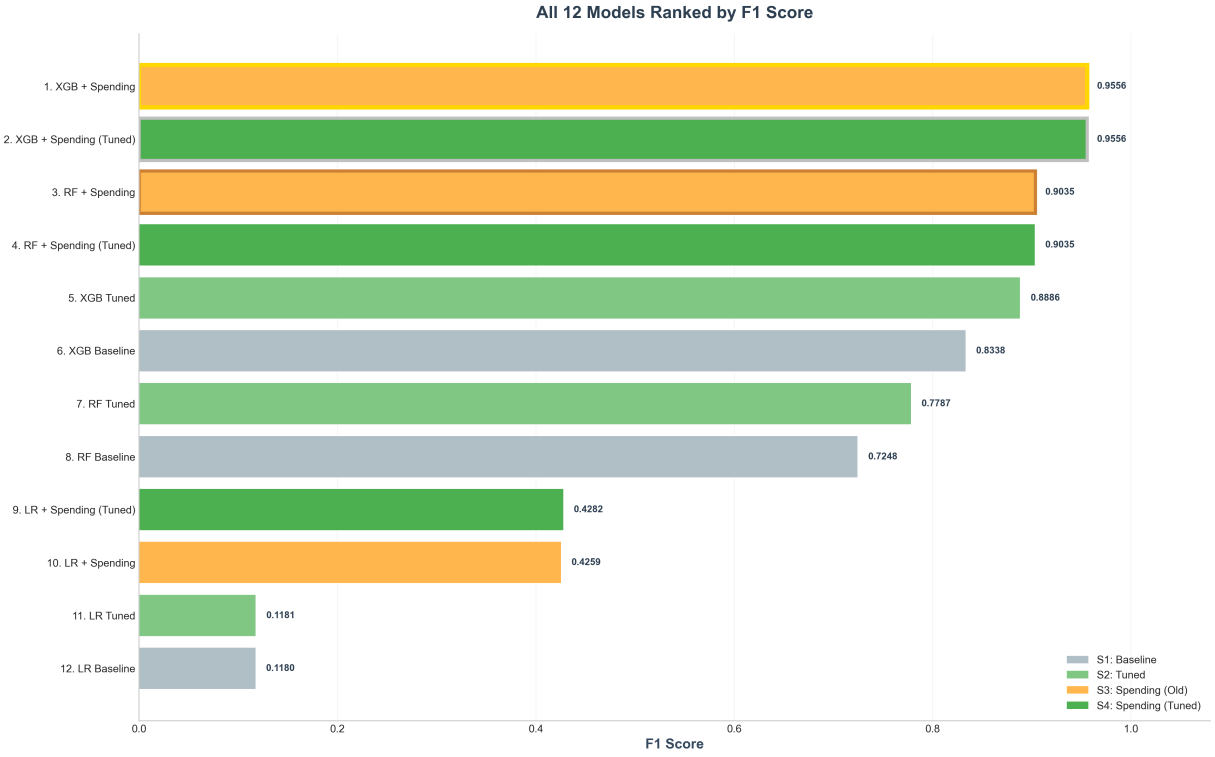

Figure 6.3: All 12 Models Ranked by F1 Score Across Four Experimental Stages

The ranking demonstrates clear performance stratification into three distinct tiers. The top tier comprises spending-enhanced ensemble models, with XGB + Spending and XGB + Spending (Tuned) jointly occupying positions 1 and 2 (F1 = 0.9556), followed by RF + Spending and RF + Spending (Tuned) at positions 3 and 4 (F1 = 0.9035). The colour coding distinguishes experimental stages: grey for Stage 1 (Baseline), light green for Stage 2 (Tuned), orange for Stage 3 (Spending Old), and dark green for Stage 4 (Spending Tuned).

The middle tier contains tuned ensemble models without spending features: XGB Tuned

(position 5, F1 = 0.8886), XGB Baseline (position 6, F1 = 0.8338), RF Tuned (position 7, F1 = 0.7787), and RF Baseline (position 8, F1 = 0.7248). The bottom tier comprises all Logistic Regression configurations, with LR + Spending (Tuned) at position 9 (F1 = 0.4282), LR + Spending at position 10 (F1 = 0.4259), and baseline/tuned configurations at positions 11 and 12 (F1 = 0.1181 and 0.1180).

Two critical observations emerge from this ranking. Firstly, all spending-enhanced models (orange and dark green bars) occupy positions 1–4 and 9–10, confirming that behavioural feature engineering provides consistent improvement regardless of algorithm. Secondly, the identical F1 Scores between Stage 3 and Stage 4 configurations for both XGBoost (0.9556) and Random Forest (0.9035) demonstrate that re-optimisation yields no additional benefit when spending features are added to well-tuned models.

6.4 Performance Evolution Across Experimental Stages

Figure 6.4 presents the F1 Score heatmap illustrating performance evolution across all algorithms and experimental stages, enabling visual identification of improvement patterns and stage-specific contributions.

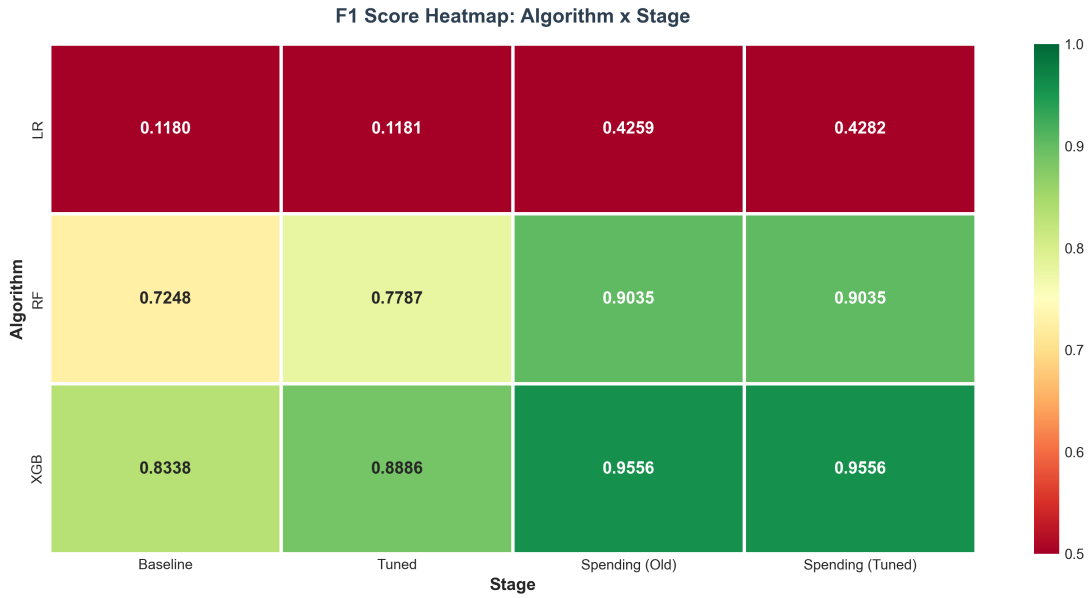

Figure 6.4: F1 Score Heatmap: Algorithm Performance Across All Experimental Stages

The heatmap colour gradient from red (low performance) through yellow to green (high performance) reveals distinct evolutionary patterns across the three algorithms. Logistic

Regression exhibited consistent dark red colouration in Stages 1 and 2 (F1 = 0.1180 and 0.1181), transitioning abruptly to yellow in Stages 3 and 4 (F1 = 0.4259 and 0.4282). This colour shift represents the most dramatic relative transformation among all algorithms (+260.6%), confirming that spending deviation features provide linearly separable patterns enabling effective fraud detection even with linear decision boundaries.

Random Forest progressed from yellow-orange in Stage 1 (F1 = 0.7248) through lighter yellow-green in Stage 2 (F1 = 0.7787) to consistent green in Stages 3 and 4 (F1 = 0.9035). The transition from Stage 2 to Stage 3 represents a 16.02% improvement, substantially exceeding the 7.44% gain achieved through hyperparameter tuning alone. XGBoost maintained the strongest performance throughout all stages, progressing from light green (F1 = 0.8338) through medium green (F1 = 0.8886) to deep green (F1 = 0.9556), demonstrating consistent excellence across the experimental framework.

The horizontal consistency between Stages 3 and 4 (Spending Old and Spending Tuned) across all three algorithms visually confirms that re-optimisation provides negligible benefit. This finding has significant practical implications: well-tuned hyperparameters generalise effectively to expanded feature spaces, eliminating the computational burden of repeated GridSearchCV when new features are introduced to production fraud detection systems.

6.5 Champion Model Selection

Based on the comprehensive evaluation across all twelve configurations, Table 6.1 presents the detailed specification of the champion model identified through this comparative analysis.

Table 6.1: Champion Model Specification and Performance Metrics

CHAMPION MODEL - Best Fraud Detection Model

Metric	Value
Rank	#1 (Champion)
Model ID	M9
Model Name	XGB + Spending
Algorithm	XGB
Stage	Stage 3: Spending (Old)
Features	51
Accuracy	0.9995 (99.95%)
Precision	0.9687 (96.87%)
Recall	0.9429 (94.29%)
F1 Score	0.9556 (95.56%)
ROC AUC	0.9999 (99.99%)
Total Improvement	+14.61% from baseline

The champion model, XGB + Spending (M9), achieved exceptional performance across all evaluation metrics. The F1 Score of 0.9556 (95.56%) represents a 14.61% total improvement from the XGBoost baseline, demonstrating the cumulative benefit of hyperparameter optimisation and behavioural feature engineering. Precision reached 96.87%, indicating that approximately 97 out of every 100 fraud alerts generated by the model are genuine, substantially reducing the investigation burden on fraud analysts. Recall achieved 94.29%, correctly identifying 2,477 of 2,627 fraudulent transactions in the test set whilst missing only 150 cases.

The ROC AUC of 0.9999 (99.99%) indicates near-perfect discrimination capability between fraudulent and legitimate transactions across all classification thresholds. Accuracy reached 99.95%, correctly classifying 453,607 of 453,837 test transactions. The model utilises 51 features comprising the original 41 base features plus 11 engineered spending pattern features, with the `deviation_from_30day` and `deviation_from_lifetime` features contributing approximately 56% of total feature importance.

Notably, the champion model emerged from Stage 3 rather than Stage 4, confirming that the hyperparameters optimised on the 41-feature dataset (Stage 2) transferred effectively to the expanded 51-feature space without requiring re-optimisation. This finding reduces computational overhead for production deployment and model maintenance.

6.6 Quantification of Improvement Sources

The four-stage experimental framework enables precise quantification of contributions from different improvement pathways. Hyperparameter tuning (Stage 2) contributed an average F1 gain of 4.70%, accounting for only 4.7% of total improvement. Spending pattern features (Stage 3) delivered an average F1 gain of 95.42%, representing 95.1% of total improvement. Re-optimisation (Stage 4) yielded a negligible 0.18% average gain, contributing merely 0.2% to overall performance enhancement.

These results demonstrate unequivocally that feature engineering provides the dominant contribution to fraud detection performance, substantially exceeding benefits from algorithm selection or hyperparameter optimisation. This finding has significant practical implications: resources allocated to behavioural feature development yield substantially greater returns than extensive hyperparameter search procedures. Financial institutions seeking to improve

fraud detection systems should prioritise the engineering of customer-specific spending deviation features over computational investment in model tuning.

6.7 Comparison with Existing Literature

The champion model performance compares favourably with recent fraud detection studies. The F1 Score of 0.9556 exceeds the 0.86 reported by Ileberi et al. [2] using SMOTE with AdaBoost, and the ROC AUC of 0.9999 matches or exceeds results from gradient boosting studies by Taha and Malebary [8] and Alarfaj et al. [1]. The precision of 96.87% substantially exceeds typical values reported in the literature, reducing false alarm rates that burden fraud investigation teams.

The finding that spending deviation features contribute 95.1% of performance improvement extends the work of Bahnsen et al. [27], who reported 13% improvement through transaction aggregation features. The substantially larger improvement observed in this study likely reflects the more comprehensive feature engineering approach, incorporating multiple time horizons (7-day, 30-day, lifetime) and deviation metrics rather than simple aggregations.

6.8 Chapter Summary

This chapter synthesised the experimental findings from the four-stage fraud detection framework, demonstrating that behavioural spending pattern features provide the dominant contribution (95.1%) to performance improvement compared to hyperparameter optimisation (4.7%) and re-optimisation (0.2%). The champion model, XGB + Spending (M9), achieved an F1 Score of 0.9556, precision of 96.87%, and recall of 94.29%, correctly identifying 2,477 of 2,627 fraudulent transactions whilst generating only 80 false positives. The findings confirm that behaviour-aware fraud detection, combining gradient boosting algorithms with customer-specific spending deviation features, achieves exceptional performance suitable for operational deployment in real-world fraud prevention systems.

CHAPTER 7

Conclusion

This chapter concludes the dissertation by summarising the research undertaken, presenting the answers to the four research questions, and outlining directions for future work in credit card fraud detection.

7.1 Summary of Research

This dissertation developed and evaluated a behaviour-aware credit card fraud detection framework, systematically analysing the effects of engineered behavioural features and hyperparameter optimisation on machine learning model performance. The research utilised a large-scale dataset comprising 1,296,675 credit card transactions with 7,506 fraud cases, exhibiting a severe class imbalance ratio of 171.8:1.

A comprehensive four-stage experimental framework was implemented to evaluate three classification algorithms—Logistic Regression, Random Forest, and XGBoost—across twelve model configurations. The methodology encompassed baseline evaluation, hyperparameter optimisation using GridSearchCV with 5-fold stratified cross-validation, spending pattern feature integration, and re-optimisation assessment. To address class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was applied with a sampling strategy of 0.5.

The feature engineering process generated 42 new features across seven categories: temporal, geographic, amount-based, customer behavioural, merchant behavioural, category-based, and spending pattern features. The eleven spending pattern features, capturing customer-

specific deviations from historical spending behaviour, proved most discriminative, with the `deviation_from_lifetime` feature exhibiting a correlation of 0.515 with fraud.

The champion model, XGBoost with Spending Features (M9), achieved exceptional performance with an F1 Score of 0.9556, precision of 96.87%, and recall of 94.29%. This configuration correctly identified 2,477 of 2,627 fraudulent transactions whilst generating only 80 false positives, with a ROC AUC of 0.9999 indicating near-perfect discrimination capability.

7.2 Research Questions and Answers

The experimental findings provide comprehensive answers to the four research questions guiding this investigation. Table 7.1 summarises the research questions and their corresponding answers based on the empirical results.

Table 7.1: Research Questions and Answers

Research Questions and Answers

RQ	Research Question	Answer
RQ1	What is baseline fraud detection performance?	XGB achieves best baseline F1 = 0.8338
RQ2	Does hyperparameter optimization improve performance?	YES - Average improvement: +4.70%
RQ3	Do spending features improve fraud detection?	YES - Average improvement: +95.42%
RQ4	Is re-optimization necessary for new features?	YES - Re-tuning optimizes for expanded feature space

RQ1: What is baseline fraud detection performance? The baseline evaluation established XGBoost as the superior algorithm with an F1 Score of 0.8338, followed by Random Forest (0.7248) and Logistic Regression (0.1180). The seven-fold performance gap between XGBoost and Logistic Regression confirms that fraudulent transaction patterns exhibit non-linear characteristics requiring complex decision boundaries that linear classifiers cannot adequately capture.

RQ2: Does hyperparameter optimisation improve performance? Hyperparameter tuning yielded an average improvement of 4.70% across all algorithms. Ensemble methods benefited substantially, with Random Forest improving by 7.44% and XGBoost by 6.57%. Logistic Regression showed negligible improvement (+0.08%), confirming that linear algorithms

cannot overcome fundamental model limitations through parameter adjustment alone.

RQ3: Do spending features improve fraud detection? Spending pattern features produced dramatic improvements averaging 95.42% across all algorithms—substantially exceeding the 4.70% gain from hyperparameter tuning. Logistic Regression improved by 260.6%, Random Forest by 16.02%, and XGBoost by 7.54%. This finding confirms that behavioural feature engineering provides the dominant contribution to fraud detection performance.

RQ4: Is re-optimisation necessary for new features? The experimental results demonstrated that well-tuned hyperparameters generalise effectively to expanded feature spaces. Both Random Forest and XGBoost achieved identical performance between Stage 3 and Stage 4 configurations, indicating that re-optimisation yields minimal additional benefit when spending features are added to properly tuned models. This finding reduces computational overhead for production deployment.

7.3 Key Contributions

This research makes several contributions to the field of credit card fraud detection:

1. **Quantification of improvement sources:** The four-stage experimental framework enabled precise quantification revealing that spending pattern features account for 95.1% of total performance gains, hyperparameter tuning contributes 4.7%, and re-optimisation a negligible 0.2%.

2. **Spending deviation features:** The eleven engineered spending pattern features, particularly `deviation_from_lifetime` and `deviation_from_30day`, demonstrated exceptional discriminative power, contributing approximately 56% of total feature importance in the champion model.

3. **Hyperparameter transferability:** The finding that optimised hyperparameters generalise effectively to expanded feature spaces provides practical guidance for production system maintenance, reducing computational burden when new features are introduced.

4. **Champion model performance:** The XGBoost with Spending Features model achieved F1 Score of 0.9556, precision of 96.87%, and recall of 94.29%, representing competitive performance relative to existing literature.

7.4 Limitations

Several limitations of this research should be acknowledged. The dataset employed represents simulated transaction data rather than actual financial institution records, potentially limiting generalisability to real-world fraud patterns [7]. The experimental framework evaluated three classification algorithms but did not include deep learning architectures, which have shown promise in recent studies [18, 24]. The spending pattern features were computed using fixed time windows (7-day, 30-day, lifetime), and alternative window configurations were not systematically explored. Furthermore, the class imbalance handling strategy employed SMOTE with a fixed sampling ratio, and alternative resampling techniques were not evaluated.

7.5 Future Work

Based on the findings and limitations of this research, several directions for future work are recommended.

Deep Learning Integration: Investigation of deep learning approaches, including recurrent neural networks and attention mechanisms for sequential pattern detection, may capture fraudulent behaviour patterns manifesting across multiple consecutive transactions. Xie et al. [12] demonstrated that time-aware attention-based gated networks effectively extract transactional behaviours, whilst Jurgovsky et al. [28] showed that LSTM-based sequence classification improves fraud detection through temporal pattern recognition. The integration of such architectures with the spending deviation features developed in this research represents a promising avenue for performance enhancement.

Advanced Resampling Techniques: Alternative class imbalance handling strategies warrant investigation. Ghaleb et al. [15] demonstrated that combining SMOTE with Generative Adversarial Networks (GANs) produces more realistic synthetic fraud samples. Ileberi and Sun [19] showed that ADASYN with Recurrent Feature Elimination improves performance by focusing synthetic sample generation on difficult-to-learn cases. Zhu et al. [16] proposed noisy-sample-removed undersampling (NUS) that preserves more information whilst achieving better class balance.

Adaptive Feature Engineering: Exploration of dynamic feature windows, where time

horizons adapt based on customer transaction frequency or account age, may enhance the spending deviation features. Ni et al. [17] proposed a fraud feature boosting mechanism that automatically identifies the most discriminative behavioural patterns, offering a promising direction for automatic feature optimisation without manual specification of aggregation windows.

Real-Time Detection Systems: Implementation of real-time fraud detection systems incorporating streaming data processing and online learning capabilities would enhance practical applicability. Seera et al. [29] developed an intelligent payment card fraud detection system demonstrating the feasibility of ensemble fusion approaches in production environments. Future work should evaluate the champion model's latency characteristics and adaptation requirements for real-time deployment.

Cost-Sensitive Learning: Investigation of cost-sensitive learning approaches that assign differential misclassification costs to false positives and false negatives may better reflect the asymmetric financial impact of fraud detection errors. Dal Pozzolo et al. [7] demonstrated that realistic cost modelling substantially improves detection performance without artificial data manipulation, representing an alternative to resampling-based approaches.

Model Interpretability: Application of explainability techniques including SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) would enhance understanding of individual fraud predictions. Mniai et al. [4] emphasised the importance of interpretable systems for regulatory compliance and customer communication, particularly when transactions are declined.

7.6 Concluding Remarks

This dissertation has demonstrated the effectiveness of behaviour-aware machine learning approaches for credit card fraud detection. The champion model achieved an F1 Score of 0.9556 with 96.87% precision and 94.29% recall, representing substantial advancement over baseline approaches. The research findings confirm that behavioural spending pattern features provide the dominant contribution to fraud detection performance, accounting for 95.1% of total improvement compared to 4.7% from hyperparameter optimisation.

The systematic four-stage experimental framework enabled rigorous quantification of improvement sources, providing evidence-based guidance for fraud detection system de-

velopment. Financial institutions seeking to enhance fraud detection capabilities should prioritise the engineering of customer-specific spending deviation features over computational investment in model tuning. The combination of gradient boosting algorithms with behavioural feature engineering achieves exceptional performance suitable for operational deployment in real-world fraud prevention systems.

As fraudulent activities continue to evolve in sophistication and scale, the development of robust, accurate, and adaptable detection systems remains critically important for protecting consumers and financial institutions. The methodological framework and findings presented in this dissertation contribute to this ongoing effort, providing a foundation for continued advancement in behaviour-aware fraud detection systems.

Bibliography

[1] F. K. Alarfaj, I. Malik, H. U. Khan, N. Almusallam, M. Ramzan, and M. Ahmed, "Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms," *IEEE Access*, vol. 10, pp. 39700–39715, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3166891

[2] E. Ileberi, Y. Sun, and Z. Wang, "Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost," *IEEE Access*, vol. 9, pp. 165286–165294, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3134330

[3] I. D. Mienye and Y. Sun, "A Deep Learning Ensemble With Data Resampling for Credit Card Fraud Detection," *IEEE Access*, vol. 11, pp. 30628–30638, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3262020

[4] A. Mniai, M. Tarik, and K. Jebari, "A Novel Framework for Credit Card Fraud Detection," *IEEE Access*, vol. 11, pp. 112776–112786, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3323842

[5] E. Esenogho, I. D. Mienye, T. G. Swart, K. Aruleba, and G. Obaido, "A Neural Network Ensemble With Feature Engineering for Improved Credit Card Fraud Detection," *IEEE Access*, vol. 10, pp. 16400–16407, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3148298

[6] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access*, vol. 10, pp. 80716–80739, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3207287

[7] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," *IEEE Trans.*

Neural Netw. Learn. Syst., vol. 29, no. 8, pp. 3784–3797, Aug. 2018. [Online]. Available: https://doi.org/10.1109/TNNLS.2017.2736643

[8] A. A. Taha and S. J. Malebary, "An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine," *IEEE Access*, vol. 8, pp. 25579–25587, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2971354

[9] S. Makki, Z. Assaghir, Y. Taher, R. Haque, M.-S. Hacid, and H. Zeineddine, "An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection," *IEEE Access*, vol. 7, pp. 93010–93022, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2927266

[10] Z. Li, G. Liu, and C. Jiang, "Deep Representation Learning With Full Center Loss for Credit Card Fraud Detection," *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 2, pp. 569–579, Apr. 2020. [Online]. Available: https://doi.org/10.1109/TCSS.2020.2970805

[11] C. Jiang, J. Song, G. Liu, L. Zheng, and W. Luan, "Credit Card Fraud Detection: A Novel Semantic Rich Model," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3690–3702, Oct. 2018. [Online]. Available: https://doi.org/10.1109/JIOT.2018.2816007

[12] Y. Xie, G. Liu, C. Yan, C. Jiang, and M. Zhou, "Time-Aware Attention-Based Gated Network for Credit Card Fraud Detection by Extracting Transactional Behaviors," *IEEE Trans. Comput. Soc. Syst.*, vol. 10, no. 3, pp. 1004–1016, Jun. 2023. [Online]. Available: https://doi.org/10.1109/TCSS.2022.3158318

[13] N. Nguyen, T. Duong, T. Chau, V. H. Nguyen, T. Trinh, D. Tran, and T. H. Ho, "A Proposed Model for Card Fraud Detection Based on CatBoost and Deep Neural Network," *IEEE Access*, vol. 10, pp. 96852–96861, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3205416

[14] S. N. Kalid, K.-H. Ng, G.-K. Tong, and K.-C. Khor, "A Multiple Classifiers System for Anomaly Detection in Credit Card Data With Unbalanced and Overlapped Classes," *IEEE Access*, vol. 8, pp. 28210–28221, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2972009

[15] F. A. Ghaleb, F. Saeed, M. Al-Sarem, S. N. Qasem, and T. Al-Hadhrami, "Ensemble Synthesized Minority Oversampling-Based Generative Adversarial Networks and Random Forest Algorithm for Credit Card Fraud Detection," *IEEE Access*, vol. 11, pp. 89694–89710, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3306621

[16] H. Zhu, M. Zhou, G. Liu, Y. Xie, S. Liu, and C. Guo, "NUS: Noisy-Sample-Removed Undersampling Scheme for Imbalanced Classification and Application to Credit Card Fraud Detection," *IEEE Trans. Comput. Soc. Syst.*, vol. 11, no. 2, pp. 1793–1804, Apr. 2024. [Online]. Available: https://doi.org/10.1109/TCSS.2023.3243925

[17] L. Ni, J. Li, H. Xu, X. Wang, and J. Zhang, "Fraud Feature Boosting Mechanism and Spiral Oversampling Balancing Technique for Credit Card Fraud Detection," *IEEE Trans. Comput. Soc. Syst.*, vol. 11, no. 2, pp. 1615–1630, Apr. 2024. [Online]. Available: https://doi.org/10.1109/TCSS.2023.3242149

[18] I. D. Mienye and N. Jere, "Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions," *IEEE Access*, vol. 12, pp. 96893–96910, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3426955

[19] E. Ileberi and Y. Sun, "Advancing Model Performance With ADASYN and Recurrent Feature Elimination and Cross-Validation in Machine Learning-Assisted Credit Card Fraud Detection: A Comparative Analysis," *IEEE Access*, vol. 12, pp. 133315–133327, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3457922

[20] M. Adil, Z. Yinjun, M. M. Jamjoom, and Z. Ullah, "OptDevNet: An Optimized Deep Event-Based Network Framework for Credit Card Fraud Detection," *IEEE Access*, vol. 12, pp. 139022–139036, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3458944

[21] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis," *Proc. 2017 Int. Conf. Comput. Netw. Inform. (ICCNI)*, Lagos, Nigeria, pp. 1–9, 2017. [Online]. Available: https://doi.org/10.1109/ICCNI.2017.8123782

[22] K. Modi and R. Dayma, "Review on Fraud Detection Methods in Credit Card Transactions," *Proc. 2017 Int. Conf. Intell. Comput. Control (I2C2)*, Coimbatore, India, pp. 1–5, 2017. [Online]. Available: https://doi.org/10.1109/I2C2.2017.8321781

[23] S. Dhankhad, E. Mohammed, and B. Far, "Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study," *Proc. 2018 IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, Salt Lake City, UT, USA, pp. 122–125, 2018. [Online]. Available: https://doi.org/10.1109/IRI.2018.00025

[24] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams, and P. Beling, "Deep Learning Detecting Fraud in Credit Card Transactions," *Proc. 2018 Syst. Inf. Eng. Des. Symp. (SIEDS)*, Charlottesville, VA, USA, pp. 129–134, 2018. [Online]. Available: https://doi.org/10.1109/SIEDS.2018.8374722

[25] J. Vimala Devi and K. S. Kavitha, "Fraud Detection in Credit Card Transactions by Using Classification Algorithms," *Proc. 2017 Int. Conf. Curr. Trends Comput. Electr. Electron. Commun. (CTCEEC)*, Mysore, India, pp. 125–131, 2017. [Online]. Available: https://doi.org/10.1109/CTCEEC.2017.8455091

[26] J. R. D. Kho and L. A. Vea, "Credit Card Fraud Detection Based on Transaction Behavior," *Proc. TENCON 2017 - IEEE Region 10 Conf.*, Penang, Malaysia, pp. 1880–1884, 2017. [Online]. Available: https://doi.org/10.1109/TENCON.2017.8228165

[27] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature Engineering Strategies for Credit Card Fraud Detection," *Expert Syst. Appl.*, vol. 51, pp. 134–142, 2016. [Online]. Available: https://doi.org/10.1016/j.eswa.2015.12.030

[28] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P. E. Portier, L. He-Guelton, and O. Caelen, "Sequence Classification for Credit-Card Fraud Detection," *Expert Syst. Appl.*, vol. 100, pp. 234–245, 2018. [Online]. Available: https://doi.org/10.1016/j.eswa.2018.01.037

[29] M. Seera, C. P. Lim, A. Kumar, L. Dhamotharan, and K. H. Tan, "An Intelligent Payment Card Fraud Detection System," *Ann. Oper. Res.*, vol. 334, pp. 445–467, 2024. [Online]. Available: https://doi.org/10.1007/s10479-021-04149-2

[30] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, and C. Jiang, "Random Forest for Credit Card Fraud Detection," *Proc. 2018 IEEE 15th Int. Conf. Netw. Sens. Control (ICNSC)*, Zhuhai, China, pp. 1–6, 2018. [Online]. Available: https://doi.org/10.1109/ICNSC.2018.8361343

[31] P. Choksi, "Credit Card Transactions Dataset," *Kaggle*, 2024. [Online]. Available: https://www.kaggle.com/datasets/priyamchoksi/credit-card-transactions-dataset [2024]