

# **EXPLAINABLE-AI (XAI) IN DEEP LEARNING MODELS FOR CREDIT CARD FRAUD DETECTION**

*A seminar report submitted to University of Huddersfield  
in partial fulfilment of degree of*

*BSc. Computer Science*

with

Artificial Intelligence

by

**Thong Minh Lai**

**Student ID: U2259343**



**DEPARTMENT OF COMPUTER SCIENCE  
SCHOOL OF COMPUTING AND ENGINEERING**

April 2025

DEPARTMENT OF COMPUTER SCIENCE  
SCHOOL OF COMPUTING AND ENGINEERING



*Certificate*

*This is to certify that this report entitled “Explainable-AI (XAI) in Deep Learning Models for Credit Card Fraud Detection” is a bonafied record of the seminar presented by **Mr. Thong Minh Lai, ID No. U2259343** under the guidance towards the partial fulfilment of the requirements for the award of **Bachelor of Science in Computer Science with Artificial Intelligence** of the **University of Huddersfield**.*

**Dr. Hyunkook Lee**  
Professor and Supervisor  
Sch. of Comp & Engg  
Huddersfield

## **Abstract**

This report addresses the important challenge of interpretation in the Deep Learning model used to detect credit card transaction fraud. While Deep Learning approaches offer better performance in detecting fraud transactions, their black-box nature creates significant obstacles for practical deployment in financial institutions, where transparency and accountability by rules are mandatory. The project utilized and evaluated the XAI methods on several Deep Learning architectures that are widely known to be used in detecting credit card transaction fraud, including Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) with attention mechanisms, trained on Sparkov's synthetic dataset. The main contribution lies in the integration and comparative analysis of three state-of-the-art Explainable AI techniques: SHAP (SHapley Additive exPlanations), LIME (Local Interpretable Model-agnostic Explanations), and Anchors (rule-based explanations). Research further evaluates the effectiveness of each XAI method based on Faithfulness, Monotonicity, and Completeness metrics. Results suggest that both models provide the most consistent explanation in architecture, while the Anchors provide more action despite the low completeness score. This work contributes to the growing field of detection by setting up a broad structure for explanation method integration, potentially bridging the difference between high performance Deep Learning models and regulatory compliance requirements in the financial sector.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Reference and problem statement . . . . .	3
1.1.1	Credit Card Fraud Detection Challenges . . . . .	3
1.1.2	Need for Explainability in Fraud Detection Models . . . . .	4
1.1.3	Black Box Problem in Deep Learning . . . . .	4
1.2	Project Scope and Objects . . . . .	5
1.3	Final Product Description . . . . .	5
1.3.1	Overview of Implemented Models (CNN, LSTM) . . . . .	5
1.3.2	Summary of XAI Techniques Integrated (SHAP, LIME, Anchors) . . . . .	5
1.3.3	Description of the System Design . . . . .	6
1.4	Intended Market and Users . . . . .	6
<b>2</b>	<b>Literature Review</b>	<b>7</b>
2.1	Credit Card Fraud Detection Fundamentals . . . . .	7
2.1.1	Evolution of Machine Learning in Fraud Detection . . . . .	7
2.2	Deep Learning Models for Fraud Detection . . . . .	8
2.2.1	CNN Architecture . . . . .	8
2.2.2	LSTM Architecture . . . . .	8
2.3	Explainable AI (XAI) in Financial Services . . . . .	8
2.3.1	Importance of Model Transparency in Financial Sector . . . . .	9
2.3.2	Regulatory Requirements (GDPR, Financial Regulation) . . . . .	9
2.3.3	Stakeholder Trust Considerations . . . . .	9
2.4	XAI Techniques . . . . .	9
2.4.1	SHAP (SHapley Additive exPlanations) . . . . .	10
2.4.2	LIME (Local Interpretable Model-agnostic Explanations) . . . . .	10
2.4.3	Anchors and Rule-Based Explanation Approach . . . . .	10
2.4.4	Comparative Analysis of XAI Approaches . . . . .	10
2.5	Synthetic Data in Model Development . . . . .	11
2.5.1	Benefits and Limitations of Synthetic Data . . . . .	11
2.5.2	The Sparkov Dataset Characteristics . . . . .	11
2.6	Ethical and Legal Considerations . . . . .	12
2.6.1	Privacy Implications of Fraud Detection Systems . . . . .	12
2.6.2	Bias and Fairness in Automated Decision Systems . . . . .	13
2.6.3	Compliance with Financial Regulations . . . . .	13

<b>3 Methodology</b>	<b>14</b>
3.1 Research Approach . . . . .	14
3.1.1 Justification of Selected Development Approach . . . . .	14
3.1.2 Experimental Design Logic . . . . .	14
3.1.3 Evaluation Framework Selection . . . . .	15
3.2 Dataset Description and Preprocessing . . . . .	16
3.2.1 Synthetic Data (Sparkov) Characteristics . . . . .	16
3.2.2 Data Cleaning and Transformation . . . . .	17
3.2.3 Feature Engineering Approaches . . . . .	17
3.2.4 Handling Class Imbalance (SMOTE Implementation) . . . . .	18
3.3 Model Architecture and Development Method . . . . .	18
3.3.1 CNN Architecture Design . . . . .	18
3.3.2 LSTM with Attention Mechanism . . . . .	19
3.4 XAI Methodology . . . . .	20
3.4.1 SHAP Integration Methodology . . . . .	20
3.4.2 LIME Integration Methodology . . . . .	21
3.4.3 Anchors Integration Methodology . . . . .	22
3.4.4 XAI Evaluation Metrics and Approach . . . . .	22
3.5 Technical Environment . . . . .	24
3.5.1 Implementation Tools and Frameworks . . . . .	24
3.5.2 Hardware Specifications . . . . .	24
3.5.3 Project Directory Structure and Development Environment . . . . .	24
<b>4 Implementation, Findings, and Analysis</b>	<b>27</b>
4.1 System Architecture . . . . .	27
4.1.1 Data Processing Pipeline . . . . .	27
4.1.2 XAI Integration Components . . . . .	29
4.1.3 System Interaction Diagram . . . . .	31
4.2 CNN Model Implementation . . . . .	32
4.2.1 Detailed Architecture . . . . .	32
4.2.2 Training Process and Optimisation . . . . .	33
4.2.3 Performance Metrics . . . . .	34
4.3 LSTM Model Implementation . . . . .	35
4.3.1 Architecture with Attention Mechanism . . . . .	35
4.3.2 Training Process and Challenges . . . . .	36
4.3.3 Performance Metrics . . . . .	37

# Chapter 1

## Introduction

Credit Card fraud detection represents an important challenge for financial institutions worldwide with significant monetary and iconic implications. This chapter refers to the problem location, establishes the requirement of sophisticated detection mechanisms, and shows the main research questions addressed in this project. Integration of Deep Learning models with explainability methods presents a promising approach to balance the detection accuracy with interpretations in this domain.

### 1.1 Reference and problem statement

Credit card fraud represents an important and growing challenge in the financial sector, causing significant monetary losses worldwide. According to the data from the UK Finance, fraudsters stole over £1.3 billion in 2021 alone through authorised and unauthorised fraud, with card fraud accounting for a significant part of these damages (UK Finance, 2022). Detection of fraud transactions presents several important challenges that increase the need for advanced computational approaches.

#### 1.1.1 Credit Card Fraud Detection Challenges

Credit cards include primary challenges in detecting fraud:

- **Interpretability:** Complex deep neural networks receiving high fraud detection accuracy often act as "black box" where decision-making processes remain opaque (Mienye & Jere, 2024). Financial institutions should not only detect fraud, but should also explain how and why specific transactions were flagged as fraud to meet regulatory requirements, build customer trust and enable human monitoring. These create a fundamental stress between performance and clarity, as the most accurate models usually provide the least transparency. Developed models that balance higher identification rates with explanatory outputs represent a significant technical challenge in credit card fraud detection systems.
- **Highly imbalanced dataset:** Fraud transactions usually represent 0.172% of all transactions (Pozzolo et al., 2015). In the Sparkov dataset used in this project (Grover et al., 2023), 1,289,169 general transactions vs are a significant imbalance with only 7,506 fraud (Grover et al., 2023; Shenoy, 2021). This imbalance creates an accurate identity of the exceptionally difficult minority fraud classes without special techniques.

- **Real-time detection requirements:** Financial institutions need to immediately identify fraud activities to prevent damage, require models that can process the transactions efficiently while maintaining accuracy (Hafez et al., 2025). This temporary barrier adds complexity to the implementation of the refined identification mechanism.
- **Complex and Evolving Fraud Pattern:** The fraudsters are constantly adapting their techniques, making sophisticated patterns that are difficult to detect using traditional rule-based systems (Sundararamaiah et al., 2024). As West and Bhattacharya (2016) notes, "Some researchers have considered models for adaptive classification, however, further research is required to fully develop these for use in practical fraud detection problems".

### 1.1.2 Need for Explainability in Fraud Detection Models

While machine learning models can achieve high accuracy in fraud detection, their "black box" nature presents significant concerns:

- **Regulator Compliance:** Financial institutions must follow rules such as GDPR in the European Union and Financial Conduct Authority (FCA) guidelines in the UK, including "right to explanation" for automated decisions affecting customers (Goodman & Flaxman, 2017). Without clear models, organisations risk regulatory punishment and legal challenges.
- **Stakeholder Trust:** Both customers and financial institutions need to understand that some transactions have been flagged as fraud to maintain confidence in the system of detection. When models cannot explain their decisions, valid transactions can be rejected without clear justification, damaging customer relationships (Vihurskyi, 2024).
- **Model verification and improvement:** Understanding how models decide is important to validate their arguments and identify potential biases or weaknesses (Barredo Arrieta et al., 2020). This understanding enables continuous improvement and adaptation to new fraud patterns.

### 1.1.3 Black Box Problem in Deep Learning

Deep learning models, while achieving state-of-the-art performance in the detection of fraud, present particular challenges regarding interpretability:

- **Complex Neural Network Architecture:** In this project, both CNN and LSTM models use a multi-layered architecture with several parameters. For example, the CNN implementation uses convolutional layers with various kernel sizes after dense layers, while the LSTM model incorporates sequential processing with the attention mechanism, making it difficult to interpret their internal decision processes (Ali et al., 2023).
- **Non-linear Transformations:** Deep Learning models apply many non-linear changes to input features, obscuring the relationship between input and output (Samek et al., 2017). This complexity makes it almost impossible to find out how specific features contribute to the final prediction without special clarity techniques.
- **Lack of transparency:** Unlike traditional rules-based systems where decision arguments are clear, deep learning models learn clearly patterns from data, give no internal explanation for their predictions (Ali et al., 2023). This lack of transparency is particularly problematic in detecting fraud, where analysts need to understand and justify why specific transactions are marked as suspicious.

## 1.2 Project Scope and Objects

The primary objective of this project is to build an explainable model to detect strong fraud by addressing several major objectives: ,

- **Handling Imbalanced Data:** Fraud transactions form only a minute fraction of financial activities. The project employs Synthetic Minority Over-sampling Technique (SMOTE) (Bowyer et al., 2011; Zhu et al., 2024), which aims to cure class imbalances.
- **Integrate XAI Techniques for Interpretability:** It is necessary that model decisions can be understood and are reliable. I unify state-of-the-art XAI methods such as SHAP (Lundberg & Lee, 2017), LIME (Ribeiro et al., 2016) and Anchors (Ribeiro et al., 2018) to distribute local interpretability.
- **Deep Learning Architectures:** To determine the optimal approach, this project included Deep Learning architectures that are widely known for their highly effective credit card fraud detection: Convolutional Neural Networks (CNNs) (Siddhartha, 2023) and LSTM-Attention networks (Ibtissam, 2021; Ibtissam et al., 2021). A comparative analysis is reported in the Raufi et al. (2024), which indicates the benchmarking strategy.
- **Synthetic Data Viability:** To augment scarce fraud data and to safeguard sensitive information, following the strategy of Breskuvienė and Dzemyda (2024), synthetic data from the Sparkov dataset (Grover et al., 2023; Harris, n.d.) is employed. This strategy, commonly used in credit card fraud detection research, enhances model robustness as elaborated by recent industry reviews.
- **Analyse Explainability Methods Effectiveness:** Finally, the project assesses the effectiveness of the integrated XAI techniques using metrics: Faithfulness, Monotonicity, and Completeness. Such analyses are crucial for ensuring that the explanations are reliable and actionable. (Seth & Sankarapu, 2025)

## 1.3 Final Product Description

The final product is envisaged as a comprehensive fraud detection system that consolidates multiple deep learning models and XAI techniques within a unified notebook (Lai, 2025d) on this GitHub repository Lai (2025a).

### 1.3.1 Overview of Implemented Models (CNN, LSTM)

Two principal models form the backbone of the system:

- **CNN Model:** Designed to extract spatial features from transaction data, the CNN model achieves high accuracy in distinguishing fraudulent from genuine transactions. Detailed model architecture and performance outcomes are implemented and shown in Lai (2025c).
- **LSTM Model:** The LSTM model is used to capture temporal dependencies inherent in sequential transaction data. Its attentional mechanism that enhances fraud detection performance are implemented in Lai (2025b).

### 1.3.2 Summary of XAI Techniques Integrated (SHAP, LIME, Anchors)

To demystify model decisions, I have incorporated three primary XAI techniques:

- **SHAP:** Utilises Shapley values to assign feature importance, offering local insights through force plots, summary plots, and dependence plots.
- **LIME:** Provides local, interpretable explanations by generating surrogate models to mimic the prediction behaviour for individual instances.
- **Anchors:** Delivers rule-based explanations with high precision, clearly outlining the conditions under which predictions are made.

### 1.3.3 Description of the System Design

The fraud detection using XAI system comprises several integrated modules:

1. **Data and Model Collection:** Obtaining synthetic transaction data and collecting fraud detection models architectures.
2. **Data Preprocessing:** Raw transaction data is cleaned, normalised, and balanced using the SMOTE method.
3. **Model Training:** Separate pipelines are implemented for training the CNN and LSTM models.
4. **XAI Integration:** Post-training, XAI techniques are applied to generate explanations for model predictions.
5. **Performance Evaluation:** Comprehensive XAI evaluation metrics, including Faithfulness, Monotonicity, and Completeness, are computed.

A figure for the architecture is depicted in Figure 3.1.

## 1.4 Intended Market and Users

The fraud detection system is designed to meet different types of stakeholders:

- **Financial Institute and Fraud Analyst:** These users require high precision detection systems that effectively indicate fraud transactions. The explainability of feature characteristics helps analysts to verify the flag-based transactions and understand the underlying causes, as is discussed in Zhou et al. (2023).
- **Data Scientists and Model Developers:** Provides a modular structure for system use and additional model refinement. The comprehensive integration of Deep Learning models and XAI components makes it a valuable tool for research and development.

# Chapter 2

## Literature Review

This chapter establishes theoretical foundations for my research on explainable AI in detecting credit card fraud. I start by checking the development of fraud detection techniques, followed by my implementation with relevant Deep Learning approaches are analysed. I then found out the increasing importance of model interpretability in financial services, with a detailed examination of the major XAI techniques employed in my research. Finally, I address synthetic data ideas and moral implications that inform my methodology.

### 2.1 Credit Card Fraud Detection Fundamentals

The Credit card fraud detection landscape has developed significantly in recent decades, moving from rules-based systems to sophisticated machine learning approaches.

#### 2.1.1 Evolution of Machine Learning in Fraud Detection

Credit card fraud detection has always been a moving goal, in which fraudsters constantly find out the tactics to evade detection. Traditionally, banks and financial institutions rely on rules-based systems and thresholds to flag suspicious transactions. While these systems were straightforward and easy to explain, they were also rigid and struggled to keep pace with the new and upcoming nature of fraud. As is highlighted by Sundararamaiah et al. (2024), these initial approaches were "fast and interpretable, they struggle to keep pace with evolving fraud patterns, thus resulting in higher false positives and undetected fraud" (Sundararamaiah et al., 2024).

The introduction of Machine Learning (ML) carried forward a significant jump. ML models, such as Decision Trees and Support Vector Machines, can learn from historical transaction data and highlight complex patterns that would be impossible to encode manually. However, even with these advances, a frequent challenge remains: severe class imbalance in the fraud dataset. Fraud transactions typically less than 1% of all transactions (Du et al., 2024; Grover et al., 2023), meaning that a naive model can simply gain high accuracy by predicting every transaction as legitimate, which leads to models being biased towards the majority class, often missing rare but significant fraud cases (Du et al., 2024).

Recently, the Deep Learning approach has revolutionised the capabilities of detecting fraud, especially in the financial sector, where credit card fraud represents a serious and growing prob-

lem due to emerging technologies and communication methods. The Deep neural network architectures, which include Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, have demonstrated better performance in detecting complex fraud patterns by learning automatically learn hierarchical representations from transaction data (Cherif et al., 2023).

## 2.2 Deep Learning Models for Fraud Detection

Complex patterns in fraud behaviour have led researchers to more sophisticated Deep Learning architectures. My research focuses on two major approaches: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

### 2.2.1 CNN Architecture

The Convolutional Neural Networks, which are traditionally applied to image processing, have been successfully adapted to transactions by treating sequences of transactions as temporal patterns. CNNs identify patterns and facilitate hierarchy, making them effective for detecting fraud signals even when being embedded within valid transaction sequences (Paulraj, 2024).

In my implementation, based on Siddharth's approach (Siddhartha, 2023), the CNN architecture processes 15 transactions through 2 convolutional layers with separate kernel sizes, followed by dense layers for classification. The model obtained 98.83% accuracy and a ROC AUC value of 0.994.

### 2.2.2 LSTM Architecture

The LSTM networks are well suited to detecting fraud due to their ability to discover temporary correlations in transaction sequences (Ibtissam et al., 2021). Unlike traditional neural networks, LSTM maintains an internal memory position that enables it to "remember" the previous states when evaluating the current state - a significant ability to detect sophisticated fraud patterns (Ibtissam et al., 2021).

My LSTM implementation is based on Ibtissam's approach (Ibtissam, 2021), which covers an attention mechanism that allows the model to focus on the most relevant characteristics within transactions. This approach obtained 98.11% accuracy with a ROC AUC value of 0.973, which provides strong comparative performance metrics for my research.

## 2.3 Explainable AI (XAI) in Financial Services

As Deep Learning models become rapidly complicated, explainability and interpretation requirements have increased proportionally, especially in highly regulated areas such as financial services (Mienye & Jere, 2024).

### 2.3.1 Importance of Model Transparency in Financial Sector

Financial institutions need to balance detection accuracy with transparency requirements. While Deep Learning models provide better performance, their "black box" nature creates significant challenges for implementation in a regulated environment. The adaptive nature of the fraudsters requires equally adaptive identification systems, yet these systems should be transparent to maintain stakeholder trust (Tomy & Ojo, 2025; Vihurskyi, 2024).

Model transparency fulfils many objectives in detecting financial fraud:

- Enabling human analysts to understand and trust model outputs (Ali et al., 2023)
- Providing insight for continuous improvement in model efficiency (Saeed & Omlin, 2023)
- Supporting auditability requirements for regulatory compliance (Goodman & Flaxman, 2017)
- Facilitating customers trustworthy explanation that the AI is making correct and non-biased decisions based on facts when transactions are flagged (Ali et al., 2023)

### 2.3.2 Regulatory Requirements (GDPR, Financial Regulation)

Financial institutions face increasing regulatory pressure on their systems of automated decision-making systems. The European Union includes the General Data Protection Regulation (GDPR) and similar rules in the UK include provisions that can be interpreted as "right to explanation" for the decisions made by automated systems (Goodman & Flaxman, 2017).

Article 22 of GDPR addresses a particularly automated decision making, in which requiring organisations to implement "suitable measures to safeguard the data subject's rights and freedoms and legitimate interests, at least the right to obtain human intervention on the part of the controller, to express his or her point of view and to contest the decision." (Goodman & Flaxman, 2017). These are implications that the detection systems should balance high performance with explainability.

### 2.3.3 Stakeholder Trust Considerations

Beyond the regulatory requirements, explainability creates confidence among various stakeholders in the ecosystem of detecting fraud:

- **Customers** requires clarification when their transactions are flagged
- **Fraud analysts** need to understand model decisions to investigate efficiently alerts
- **Compliance officers** must verify that decisions align with regulatory standards
- **Model Developers** rely on clarification to refine and improve the system

Research by Gilpin et al. (2018) indicates that transparent AI systems that can provide satisfactory explanations of their decisions can lead to wider user acceptance and increased trust compared to opaque systems, even when both get similar performance metrics.

## 2.4 XAI Techniques

To address the black box challenges of Deep Learning models, my research implements three major XAI techniques: SHAP, LIME and Anchors. Each one provide a different approach to

the explanation, with complementary strengths and boundaries.

#### 2.4.1 SHAP (SHapley Additive exPlanations)

The game theory concepts of SHAP values provides an integrated approach to explain individual predictions (Barredo Arrieta et al., 2020; Lundberg & Lee, 2017). The technique reflects the contribution of each feature for a specific prediction by calculating Shapley values, a concept from the cooperative game theory that considers the "payout" (prediction result) between "players" (features).

In my implementation, SHAP enables local explanation to one specific transaction prediction. The approach generates visualisations including force plots and summary plots that provide an intuitive representation of the feature contributions to model decisions (Lai, 2025d).

#### 2.4.2 LIME (Local Interpretable Model-agnostic Explanations)

LIME helps to build an explanatory surrogate model that estimates locally complex models around specific predictions (Ribeiro et al., 2016). The technique works by perturbing input features around a specific considering sample and measuring how the model predictions change, then fits a simple (linear) explanatory model for this local behavior.

To detect fraud, LIME provides a particularly valuable insight to individual transaction analysis, helping analysts understand why specific transactions were marked as suspicious. My implementation shows how LIME can also provide accessible clarification for complex CNN and LSTM architectures (Lai, 2025d).

#### 2.4.3 Anchors and Rule-Based Explanation Approach

Anchors produce the high-precision rules (IF-THEN rules) to explain the models decisions (Ribeiro et al., 2018). Unlike SHAP and LIME, which provide the important weights for each features, Anchors focuses on making clear, deterministic rules that are sufficient to explain predictions with high confidence.

In the implementation, Anchors generate rules such as:

```
IF transaction_amount > £1000 AND
    time_since_last_transaction < 5 minutes AND
    different_country = True
THEN prediction = Fraud (confidence 0.95)
```

(Lai, 2025d). These rules-based explanations are particularly valuable for fraud analysts who require clear decision criteria.

#### 2.4.4 Comparative Analysis of XAI Approaches

The evaluation of XAI techniques in this project is based on three core metrics: **Faithfulness**, **Monotonicity**, and **Completeness**. These metrics are widely recognised in literature, which is

essential to assess the quality and reliability of model explanations (Kadir et al., 2023; Pawlicki et al., 2024), especially in high-stakes domains such as fraud detection.

- **Faithfulness:** This metric measures how explanations reflect the relationship of the model in training data patterns (Kadir et al., 2023). This is usually evaluated by removing the features identified as important and by measuring the change in the prediction of the model. As is said in Kadir et al. (2023), "A high faithfulness correlation indicates that the model faithfully detects patterns and information in the training data".
- **Monotonicity:** It assesses that placing features in the order of their importance on the baseline surface leads to the expected changes in predictions Kadir et al. (2023). In other words, the features considered to be more important by the XAI method should have more impact on the output of the model when removed.
- **Completeness:** This metric evaluates how much explanation of the model is captured. It computes the difference between the original prediction and the prediction when the features in the explanation are used only (Pawlicki et al., 2024; Sundararajan et al., 2017).

## 2.5 Synthetic Data in Model Development

Given the sensitive nature of financial transaction data, synthetic datasets play an important role in the research and development in the fraud detection sector and in supporting data privacy preservation (Financial Conduct Authority, 2024; Vallarino, 2025).

### 2.5.1 Benefits and Limitations of Synthetic Data

Model development shows many advantages when using synthetic data fraud:

- **Privacy protection:** No risk of exposing sensitive customer information
- **Control data characteristics:** Spontaneous ability to generate specific fraud patterns
- **class balance management:** Construction of dataset with desired fraud/legitimate ratios
- **Volume scaling:** Generation of large dataset for model training

However, synthetic data also presents limitations, especially regarding its belief towards real-world patterns and cannot replace the real customer data (Financial Conduct Authority, 2024) or may not fully capture the complexities of real-world transactions (Kamalaruban et al., 2024). Ryman-Tubb et al. (2018) note that "As the profiles of genuine and fraudulent behaviours change over time, synthetic data may be insufficiently rich. Therefore, there is no reason to suspect that results cited will necessarily be the same when scaled using real-world data".

### 2.5.2 The Sparkov Dataset Characteristics

My research utilises the Sparkov dataset (Grover et al., 2023), a synthetic credit card transaction dataset specifically designed for fraud detection research. The train dataset includes 1,296,675 transactions, of which 7,506 (approximately 0.58%) are fraudulent and the test dataset includes 553,574 transactions, with 2,145 (approximately 0.39%) are fraudulent, mirroring the class imbalance challenge found in real-world scenarios (Grover et al., 2023; Lai, 2025d).

After feature engineering so it can fit the purpose of this project, the dataset contains 15 features, including:

- Transaction amount and time
- Merchant category and location
- Customer demographics and account details
- Geographical information and travel patterns

(Lai, 2025d)

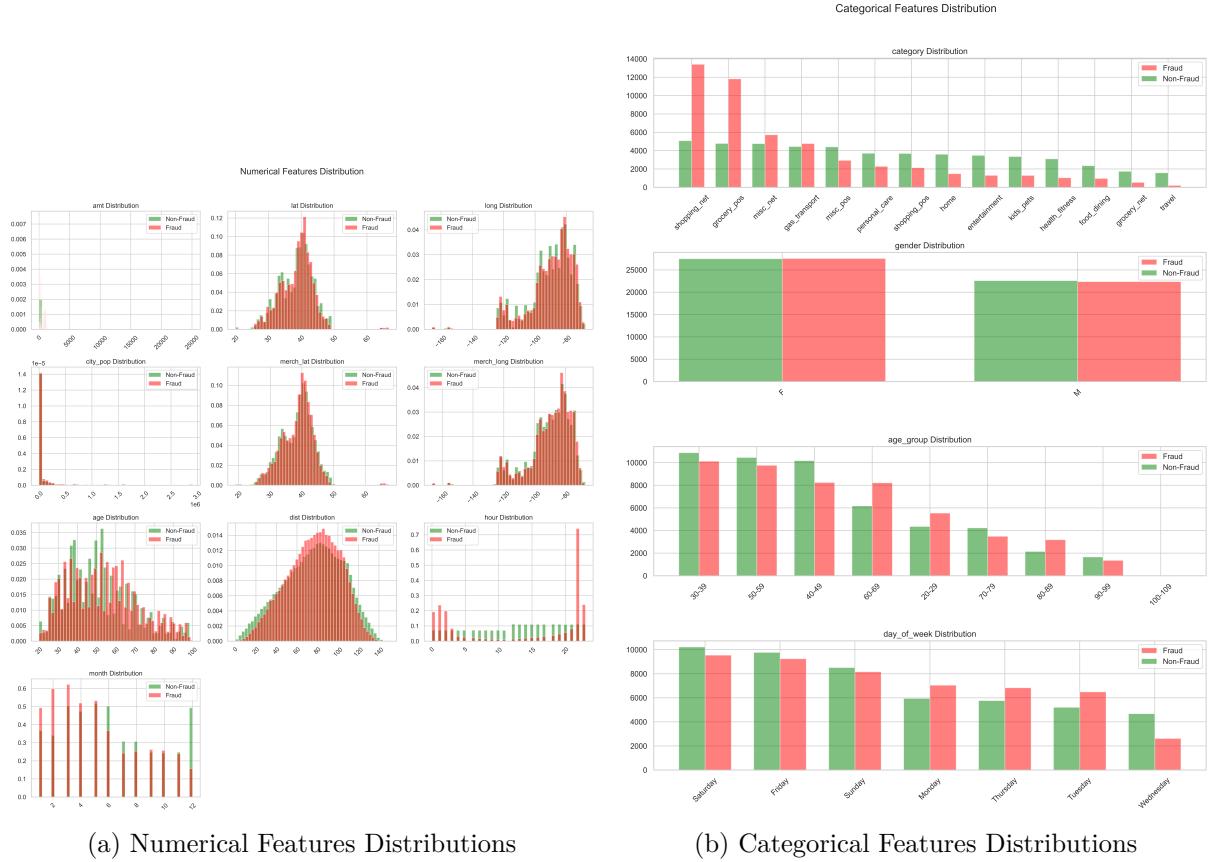


Figure 2.1: Features Distributions in the Sparkov Train Dataset

These features provide sufficient complexity to train refined detection models while maintaining realistic statistical distributions compared to real transaction data.

## 2.6 Ethical and Legal Considerations

The deployment of systems to detect AI-based fraud raises important moral and legal issues that must inform both research and implementation approaches (Financial Conduct Authority, 2024).

### 2.6.1 Privacy Implications of Fraud Detection Systems

Fraud detection systems process sensitive personal and financial data, raising significant privacy concerns. Even with synthetic training data, the system deployed will eventually interact with real customer information, which requires strong privacy and security. So it is important to

make sure to balance the need to prevent fraud and the need of customers for privacy Găbudeanu et al. (2021).

### **2.6.2 Bias and Fairness in Automated Decision Systems**

AI-based fraud detection systems, at risk of amplifying the biases present in training data. Research by Kamalaruban et al. (2024) suggests that high-value fraud missed may lead to models that can be unequally demolished transactions of specific demographic groups or geographical areas, despite a balanced detection rate.

My project also addresses these concerns:

- Applied the balanced sampling technique during training
- XAI methods to identify and address fairness issues with more interpretation
- Considering various user segments

### **2.6.3 Compliance with Financial Regulations**

In addition to the GDPR, the fraud detection system must follow several financial regulations, including anti-money laundering (AML) requirements, Know Your Customer (KYC) provisions and specific sector industry standards (Vallarino, 2025).

The integration of XAI techniques implemented in my research supports these compliance efforts, directly providing the transparency and auditability necessary for these rules. By allowing the revision of human inspection and the clarity of automated decisions, the system maintains the necessary balance between automation and compliance.

# **Chapter 3**

## **Methodology**

The chapter describes the methodological approach used in this project, including research design, dataset processing, model development, XAI implementation and technical environment. The methodology is based in established research practices and constructs upon the existing work in the field of Deep Learning and explainable AI for detecting fraud.

### **3.1 Research Approach**

#### **3.1.1 Justification of Selected Development Approach**

The development approach to this project is based on the integration of Deep Learning models with explainable AI techniques for detecting credit card fraud. This hybrid approach was chosen due to its proven effectiveness in addressing complex problems like fraud detection (Uwaezuoke & Swart, 2024), while maintaining interpretation. Deep Learning models excel in identifying complex relations in data, which is important to detect fraud where the fraud strategies can be subtle and evolve over time.

The specific combination of the Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks was chosen due to their complementary advantages (Uwaezuoke & Swart, 2024). The CNNs are effective in capturing spatial patterns of transaction features, while LSTMs excel in identifying temporary dependencies of sequential transaction data (Uwaezuoke & Swart, 2024). This multi-model approach allows for strong comparison and evaluation of explainability in various architectures for the detection of fraud.

#### **3.1.2 Experimental Design Logic**

The experimental design model follows a systematic approach to development, training and evaluation. First, both CNN and LSTM models are trained at the same preprocessed dataset to ensure proper comparison. Then, on each model, apply widely known XAI techniques (SHAP, LIME, and Anchors), which provide a variety of perspectives on the interpretation of the model. This enables a comprehensive evaluation of both design model performance and explainability.

As Seth and Sankarapu (2025) stated, "Combining standardised metrics with flexibility and human expertise can create a more robust, adaptable, and meaningful evaluation framework for XAI systems". Following this guidance, experimental design includes quantitative performance

metrics and qualitative explainability assessment (Ma, 2024; Seth & Sankarapu, 2025).

### 3.1.3 Evaluation Framework Selection

The evaluation structure employs a multidimensional approach to assess both predictive performance and interpretation. For predictive stating performance, the standard metrics are used: accuracy, precision, recall, F1-score, and AUC-ROC. These metrics are suitable for this imbalanced dataset, where the positive class (fraud) is quite low.

For explainability assessment, I employ specific metrics for each XAI evaluation. Faithfulness measures how well the explanations reflect the actual decision process of the model, Monotonicity assesses whether the importance of the feature increases leads to stronger predictions and Completeness evaluates how comprehensively the explanations cover enough information for the output (Kadir et al., 2023; Pawlicki et al., 2024).

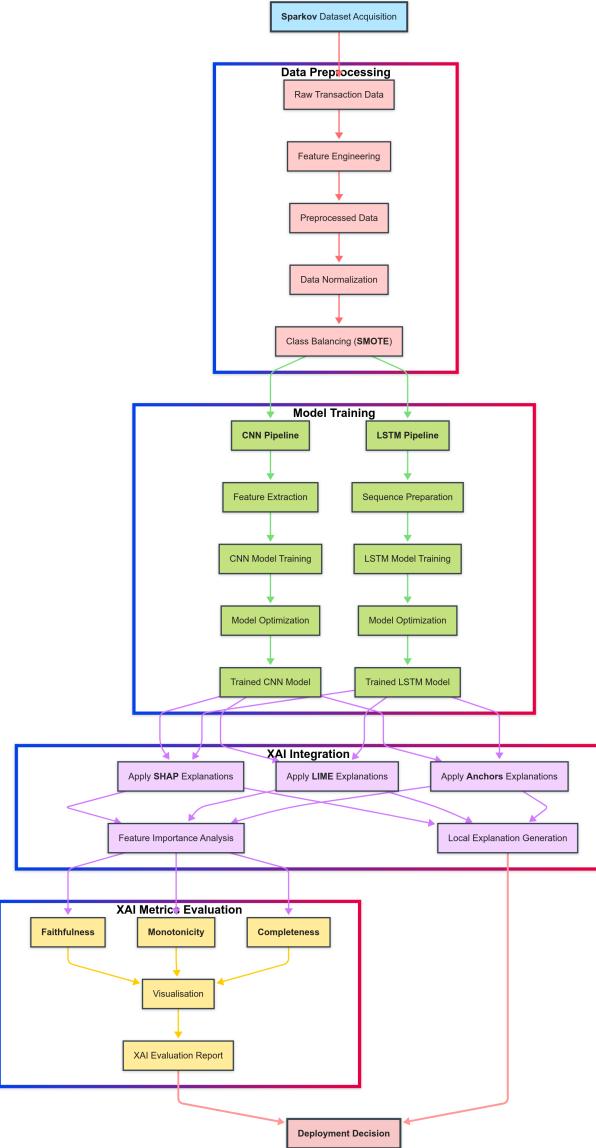


Figure 3.1: Illustrative diagram of the fraud detection system architecture showing data flow from preprocessing to model training, XAI integration and evaluation.

## 3.2 Dataset Description and Preprocessing

### 3.2.1 Synthetic Data (Sparkov) Characteristics

The synthetic credit card transactions data was generated using the Sparkov data generator, as real financial datasets are often limited by privacy concerns and regulatory restrictions. The Sparkov dataset imitates realistic credit card transactions, including both legitimate and fraudulent activities, based on the pattern seen in the real-world financial data (Grover et al., 2023).

Characteristic	Value
Total Transactions	1,296,675
Fraud Rate	0.58%
Features	22
Temporal Range	2 years
Transaction Types	Multiple categories

Table 3.1: Sparkov Synthetic Dataset Characteristics

The dataset includes 22 fields, including transaction amount, timestamp, location data (latitude and longitude for both customer and businessman), and customer demographic information:

Table 3.2: Credit Card Transaction Data Fields

Field Name	Description
trans_date_trans_time	Date and time when transaction occurred
cc_num	Credit card number of customer
merchant	Name of merchant where transaction occurred
category	Category of merchant (e.g., travel, shopping_net, etc.)
amt	Amount of transaction
first	First name of credit card holder
last	Last name of credit card holder
gender	Gender of credit card holder
street	Street address of credit card holder
city	City of credit card holder
state	State of credit card holder
zip	ZIP code of credit card holder
lat	Latitude location of credit card holder
long	Longitude location of credit card holder
city_pop	Population of credit card holder's city
job	Occupation of credit card holder
dob	Date of birth of credit card holder
trans_num	Transaction number
unix_time	UNIX timestamp of transaction
merch_lat	Latitude location of merchant
merch_long	Longitude location of merchant
is_fraud	Target class indicating whether transaction is fraudulent (1) or legitimate (0)

### 3.2.2 Data Cleaning and Transformation

Raw Sparkov data requires several preprocessing stages before training models. First, the data is cleaned to handle missing values and remove discrepancies.

For changes, the transaction category and customer gender are converted into a numerical format using ordinal encoding. Numerical features are standardised to reduce the influence of outliers, which are common in transaction amount feature.

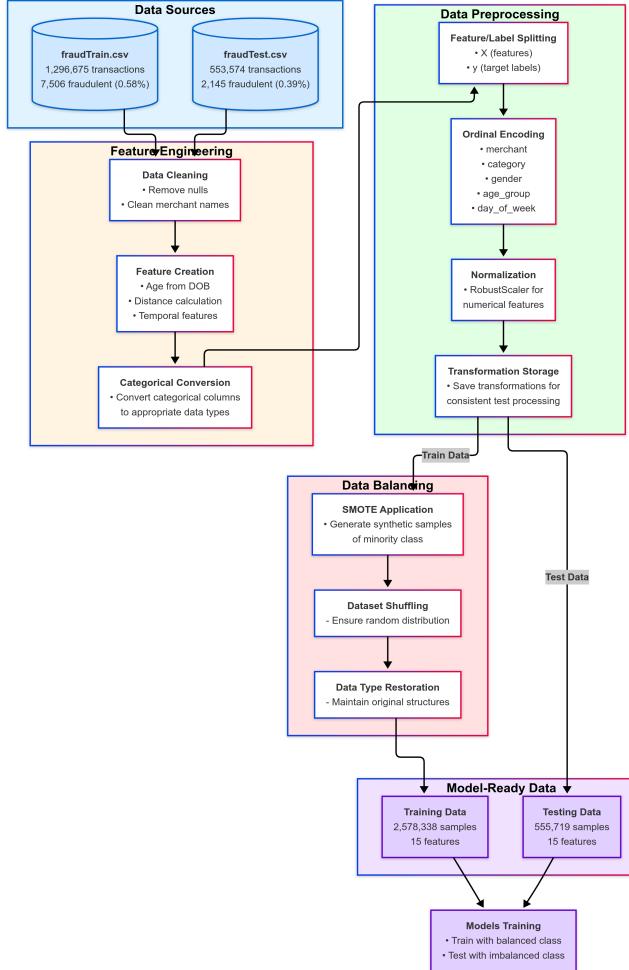


Figure 3.2: Data Preprocessing Pipeline for Fraud Detection

### 3.2.3 Feature Engineering Approaches

This process of feature engineering plays an important role in increasing the discriminatory for the models. Following approaches from the literature (Grover et al., 2023), remaining derived features are created:

- **Distance calculation:** The Haversine formula is applied to compute the distance between the customer and merchant locations, as geographical anomalies can indicate potential fraud.
- **Temporal features:** Transaction hour, day of week, and month are extracted from timestamps to capture time-based fraud patterns.

- **Age grouping:** Customer ages are categorised into meaningful groups (e.g., 18-25, 26-35) to capture demographic patterns.

These engineered features enhance the model's ability to identify fraudulent patterns that may not be clear from raw data alone.

### 3.2.4 Handling Class Imbalance (SMOTE Implementation)

The important imbalance of the classes in the data (about 0.58% fraud transactions) poses a challenge for model training. To address this, excessive SMOTE has been applied, which generates minority class synthetic examples by interpolating them among examples of existing fraud rather than simply duplicating them. (Bowyer et al., 2011; Grover et al., 2023; Pozzolo et al., 2015; Zhu et al., 2024)

The implementation uses an **imbalanced-learn** library with a sampling strategy that balances the dataset in a proportion of 50:50 for training dataset. This approach helps to prevent the model from predicting the majority class (legitimate transactions) and improves its ability to identify the minority of fraud cases.

## 3.3 Model Architecture and Development Method

### 3.3.1 CNN Architecture Design

The CNN model is designed to capture spatial patterns in the transaction data features. The architecture follows design principles established by Siddhartha (2023), adapted to the specific characteristics of credit card transaction data.

#### Layer Configuration

The CNN architecture consists of the following layers:

Table 3.3: CNN Layer Configuration

Layer	Configuration	Activation
Input	15 features	-
Convolutional 1	64 filters, kernel size 2	ReLU
Batch Normalisation	-	-
Convolutional 2	32 filters, kernel size 2	ELU
Dropout	Rate 0.3	-
Dense	64 units	ReLU
Flatten	-	-
Fully Connected	64 units	ReLU
Dropout	Rate 0.2	-
Output	1 unit	Sigmoid

This architecture results in a model with 59,937 trainable parameters, striking a balance between model capacity and computational efficiency.

## Training Procedure

Hyperparameter selection was guided by empirical evaluation and prior research in the domain. The key hyperparameters include:

- **Optimizer:** Adam with default parameters ( $\beta_1 = 0.9, \beta_2 = 0.999$ )
- **Loss function:** Binary cross-entropy
- **Batch size:** 30,000
- **Early Stopping:** Patient of 4 epochs
- **Epochs:** 100, Training until convergence or early stopping

These hyperparameters were selected based on their effectiveness in similar fraud detection tasks as documented by Ibtissam (2021) and Ibtissam et al. (2021).

### 3.3.2 LSTM with Attention Mechanism

#### Model Structure

The LSTM model is designed to capture temporal dependencies in transaction data, based on the architecture proposed by Ibtissam et al. (2021) and adapted for fraud detection. The model structure includes:

Table 3.4: LSTM Layer Configuration

Layer	Configuration	Activation
Input	15 features	-
LSTM 1	50 units, dropout 0.3, recurrent dropout 0.2	Tanh
LSTM 2	50 units, dropout 0.3, recurrent dropout 0.2	Tanh
Attention	Custom implementation	Softmax
Dense	1 unit	Sigmoid

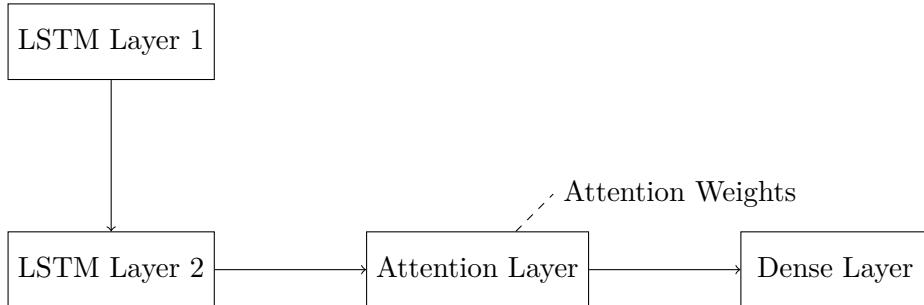


Figure 3.3: LSTM Model with Attention Mechanism

#### Attention Layer

The attention mechanism is applied based on work by Vaswani et al. (2023), which allows the model to focus on the most relevant features for predicting. The attention layer calculates a weighted sum of the LSTM output, which has a weight determined by a learning reference vector. Mathematically, the attention mechanism is defined as follow:

$$u_t = \tanh(W_w h_t + b_w) \quad (3.1)$$

$$\alpha_t = \frac{\exp(u_t^T u_w)}{\sum_t \exp(u_t^T u_w)} \quad (3.2)$$

$$v = \sum_t \alpha_t h_t \quad (3.3)$$

Where  $h_t$  represents the hidden state at the time  $t$ ,  $W_w$  and  $b_w$  are trainable parameters,  $u_w$  is the reference context vector,  $\alpha_t$  are attention weights, and  $v$  is the context vector that focuses on the most relevant parts of the input sequence.

This architecture results in a model with 33,502 trainable parameters, which is more parameter-efficient than the CNN model, while still maintaining a strong performance.

## Training Procedure

The LSTM model was trained using a similar approach to the CNN model, with the following specifics:

- **Optimizer:** Adam with learning rate 0.001
- **Loss Function:** Binary cross-entropy
- **Batch Size:** 30,000
- **Early Stopping:** Patient of 4 epochs
- **Epochs:** 50, Training until convergence or early stopping

## 3.4 XAI Methodology

### 3.4.1 SHAP Integration Methodology

SHAP was designed following a model-agnostic approach. This method allows for the explanation of any machine learning model by approximating Shapley values from game theory, which represent the average marginal contribution of a feature across all possible feature combinations (Lundberg & Lee, 2017).

---

#### Algorithm 1 SHAP for Credit Card Detection

---

```

1: Input: Trained model  $f$ , transaction  $x$ 
2: for each feature  $i$  in  $x$  do
3:   Initialize SHAP value  $\phi_i = 0$ 
4:   for each subset  $S$  of features not containing  $i$  do
5:     Create two samples:  $x_{S \cup \{i\}}$  and  $x_S$ 
6:     Compute marginal contribution:  $f(x_{S \cup \{i\}}) - f(x_S)$ 
7:     Weight the contribution based on subset size
8:     Add weighted contribution to  $\phi_i$ 
9:   end for
10: end for
11: Output: SHAP values  $\phi_1, \phi_2, \dots, \phi_n$  showing each feature's contribution to  $f(x)$ 

```

---

Design includes:

- **Background Dataset Creation:** A representative dataset of 100 samples from the training data to work as reference points for SHAP calculations.
- **Shapley Value Calculation:** For each feature in a transaction, calculate its contribution to the estimated possibility of fraud by measuring the average impact of having that feature in all possible combinations of features.
- **Core SHAP Visualisations:** with force plots and waterfall charts to provide the explanation of the model's prediction. The forces plot is greatly useful in showing how each feature pushes the prediction away from the base value, while the summary plots display the feature importance throughout the dataset.

The design handles both numerical features (eg, transactions, transaction times) and categorical features (e.g., merchant category) through appropriate preprocessing steps before feeding them to the SHAP explainer (as categorical features will need to be handled differently in the SHAP explainer).

### 3.4.2 LIME Integration Methodology

LIME was implemented to make a surrogate model that approximates deep learning models around individual predictions. This approach allows for the spontaneous, intuitive, feature-based explanation of specific transaction classifications (Ribeiro et al., 2016).

---

**Algorithm 2** LIME for Credit Card Detection

---

- 1: **Input:** Trained model  $f$ , transaction  $x$
  - 2: Generate  $N$  perturbed samples around  $x$  by randomly changing feature values
  - 3: **for** each perturbed sample  $x'$  **do**
  - 4:     Predict  $f(x')$
  - 5:     Compute similarity between  $x$  and  $x'$
  - 6: **end for**
  - 7: Fit a simple interpretable model  $g$  (e.g., linear model) to predict  $f(x')$  using the perturbed samples, weighted by similarity
  - 8: **Output:** Coefficients of  $g$  as explanations for  $f(x)$ 's prediction
- 

Design includes:

- **Data Perturbation:** To get explanations for each transaction, perturbation samples are obtained by perturbing the feature values around the original sample. This detects the model behaviour in the local neighbourhood.
- **Kernel Weighting:** Depending on their proximity to the original sample, the weights for closer samples are assigned higher.
- **Local Model Fitting:** Fit a weighted linear model for perturbation samples and their predictions from the black-box model. This linear model acts as an explanatory approximation of the complex models locally.
- **Feature Discretisation:** Continuous (numerical) variables were discriminated against to improve explainability while maintaining fidelity for the original model. For example, the amount of transactions, the discretionary thresholds were set based on distribution quantiles.

LIME explanations provide feature weights contribution, which helps fraud analysts understand which factors most affect a specific fraud prediction.

### 3.4.3 Anchors Integration Methodology

The Anchors method raises the IF-THEN rules that explain the predictions. Unlike SHAP and LIME, which provide feature continuous contribution scores, Anchors provide clear, actionable rules that are easy to understand for non-technical stakeholders (Ribeiro et al., 2018).

---

#### Algorithm 3 Anchors for Credit Card Detection

---

```

1: Input: Trained model  $f$ , transaction  $x$ 
2: Initialize anchor  $A = \emptyset$ 
3: while precision of  $A$  (fraction of perturbed samples where  $f$  predicts same as  $f(x)$ ) < threshold do
4:   For each candidate feature not in  $A$ :
5:     Add feature to  $A$  and estimate new precision
6:   Add feature that increases precision the most to  $A$ 
7: end while
8: Output: Anchor  $A$ : set of feature-value rules that "guarantee"  $f(x)$ 's prediction with high precision

```

---

Design includes:

- **Rule Candidate Generation:** Generate potential rules based on feature-value pairs from the considering sample that need to generate the explanation, as in the Example 2.4.3
- **Best Anchor Selection:** Select the best anchor (rule) based on precision, which is computed based on the proportion of instances where the rule holds true and the model makes the same prediction.

### 3.4.4 XAI Evaluation Metrics and Approach

To ensure the reliability and validity of XAI implementation, I developed a comprehensive evaluation assessment structure based on three key metrics widely used in the XAI field:

Table 3.5: XAI Evaluation Metrics Brief

Metric	Description	Evaluation Method
Faithfulness	Measures the correlation between feature importance and changes in the model's predictions when features are altered	Feature importance correlation analysis with sequential feature removal
Monotonicity	Evaluates whether removing features causes consistent changes in the model's predictions	Sequential feature removal testing with prediction tracking
Completeness	Assesses how much of the model's prediction is captured by the explanation	Coverage measurement comparing explained variance to total variance

Pseudo-algorithms:

---

**Algorithm 4** Faithfulness Metric for XAI Explanations

---

- 1: **Input:** Model  $f$ , sample  $x$ , explanation scores  $E$  for features
- 2: **for** each feature  $i$  in  $x$  **do**
- 3:     Remove or mask feature  $i$  in  $x$  to get  $x_{-i}$
- 4:     Compute prediction difference:  $\Delta_i = |f(x) - f(x_{-i})|$
- 5: **end for**
- 6: Compute correlation between  $\Delta_i$  and  $E_i$  across all features
- 7: **Output:** Faithfulness score (e.g., correlation coefficient)

---

---

**Algorithm 5** Monotonicity Metric for XAI Explanations

---

- 1: **Input:** Model  $f$ , sample  $x$ , explanation scores  $E$  for features
- 2: **for** each feature  $i$  in  $x$  **do**
- 3:     Increase feature  $i$ 's value to get  $x_{+i}$
- 4:     Compute prediction change:  $\Delta_i = f(x_{+i}) - f(x)$
- 5:     **if**  $E_i > 0$  **then**
- 6:         Check if  $\Delta_i > 0$  (prediction increases)
- 7:     **else if**  $E_i < 0$  **then**
- 8:         Check if  $\Delta_i < 0$  (prediction decreases)
- 9:     **end if**
- 10: **end for**
- 11: Calculate fraction of features where explanation and prediction change agree
- 12: **Output:** Monotonicity score (agreement ratio)

---

---

**Algorithm 6** Completeness Metric for XAI Explanations

---

- 1: **Input:** Model  $f$ , sample  $x$ , explanation scores  $E$  for features
- 2: Compute model prediction:  $y = f(x)$
- 3: Compute baseline prediction:  $y_{base} = f(\text{baseline input})$
- 4: Sum explanation scores:  $S = \sum_i E_i$
- 5: Compute completeness error:  $|S - (y - y_{base})|$
- 6: **Output:** Completeness score (lower error means higher completeness)

---

The evaluation approach included:

- **Stratified Sampling:** Use balanced stratified sampling to select test cases for evaluation, ensuring comprehensive evaluation in various confidence levels and predicting results.
- **Edge Case Analysis:** Special attention for collecting the "edge cases", such as abnormally high confidence scores or legitimate transactions near decision boundaries, fraud transactions with valid transactions, to ensure that clarifications are reliable in challenging scenarios.
- **Cross-Model Comparison:** Compare to identify model-specific biases in feature importance and explanation stability in both CNN and LSTM models. This helped to understand whether different architecture depends on different features for their predictions.
- **Confidence bin analysis:** It breaks down the performance further, highlighting the strengths and weaknesses of each XAI method in different prediction confidence.

## 3.5 Technical Environment

This section details the implementation tools, hardware specifications and development environment used for this project. It is important to ensure a consistent and well-written technical environment, ensuring reproducibility.

### 3.5.1 Implementation Tools and Frameworks

The project was implemented using the following tools and frameworks:

Table 3.6: Implementation Tools and Frameworks

Component	Specification
Programming Language	Python 3.10.11 (latest version supports tensorflow-gpu on Windows NT)
Deep Learning Framework	TensorFlow 2.10.1
Core Libraries	pandas, numpy, scikit-learn, imblearn
XAI Libraries	shap, lime, anchor-exp
Visualisation Libraries	matplotlib, seaborn
Data Processing	pandas, numpy
Version Control	GitHub

Custom utility modules ('utils.py') provided supportive tasks for data preprocessing, model evaluation and visualisation. This module ensured continuous data handling in all model implementations and XAI techniques.

### 3.5.2 Hardware Specifications

The GPU acceleration was particularly helpful to train the CNN model and LSTM model with attention mechanisms, which required much more computational resources. The hardware configuration was verified on the runtime to ensure optimal performance (Lai, 2025d).

The models were trained and evaluated on the following hardware configuration:

- **GPU Support:** 1 NVIDIA GPU enabled
- **CUDA Version:** 64\_112
- **CUDNN Version:** 64\_8
- **Memory Requirements:** 8GB RAM for better handling of the Sparkov dataset
- **Storage:** SSD storage for data loading performance

### 3.5.3 Project Directory Structure and Development Environment

The development environment was configured to ensure consistency across different implementation phases and team members:

- **Virtual Environment:** Suggest create separate virtual environment when running (example: 'venv\_xai\_fraud\_detection')
- **Random Seed:** Set to 42 for reproducibility across all experiments
- **Project Structure:**
  - Model architectures stored in `architectures/` directory

- All input, output data stored in `data/` directory
- Visualisations, plots and diagrams stored in `visualisation/` directory
- **Version Control:** GitHub repository with branch-based workflow (Lai, 2025a)

## Directory Structure

Project Directory Structure

```

/
├── XAI_methods.ipynb ..... Explainable AI methods implementation
├── Siddhartha_CNN.ipynb..... CNN model implementation
├── Ibtissam_LSTM.ipynb ..... LSTM model implementation
├── utils.py ..... Utility functions
├── requirements.txt ..... Dependencies
├── architectures/ ..... Trained model storage
│   ├── model_1_Siddhartha_CNN_acc99/
│   └── model_2_Ibtissam_LSTM_acc98/
├── data/ ..... Data files and results
│   ├── predictions.csv
│   ├── stratified_samples.csv
│   └── xai_metrics.json
└── visualisation/ ..... Visualisation outputs
├── README.md ..... Project brief
└── Report Documentation.pdf ..... Project Report Documentation

```

## Development Environment Configuration

### Development Environment Configuration:

- **Jupyter Notebook** for interactive development (`.ipynb` files)
- **Dependencies:** Installed via `pip install -r requirements.txt`
- **Data and Models:** Organized in `data/` and `architectures/` folders
- **Utilities:** Helper functions in `utils.py`

Figure 3.4: Development Environment Configuration

I used dependency management through a `requirements.txt` file to ensure consistent package versions in the environments. This approach helped prevent compatibility issues during integration:

Listing 3.1: Dependencies Installation

```

# Create virtual environment
python -m venv .venv_xai_fraud_detection

# Activate environment
# On Windows
.venv_xai_fraud_detection\Scripts\activate
# On Unix/MacOS

```

```
# source .venv_xai_fraud_detection/bin/activate  
  
# Install dependencies  
pip install -r requirements.txt
```

Constant environmental configuration allows for spontaneous transition between development, training and evaluation stages, ensuring that the results are comparable and reproducible throughout the project life cycle.

# Chapter 4

# Implementation, Findings, and Analysis

This chapter describes the practical implementation of the credit card fraud detection system, using Deep Learning models along with Explainable AI integration. The implementation follows the methodological structure established in the previous chapter 3, system architecture, model implementations and collecting performance results. Each section examines the technical aspects of implementation, highlighting major decisions and their results.

## 4.1 System Architecture

System architecture includes many interconnected components designed to facilitate efficient data processing, model training and explanation integration and is designed with modularity and extensibility. Mutual helper functions are implemented in the 'utils.py' file module for separate notebooks that are intended for individual development purposes.

### 4.1.1 Data Processing Pipeline

The data processing pipeline converts raw transaction data into a suitable format for feeding into Deep Learning models. This multi-stage process involves feature engineering, preprocessing as shown in Figure 3.2.

The feature engineering phase applies several major stages:

- **Temporal Feature Extraction:** Convert Unix timestamps into cyclic time features such as hour of day, day of week and month.
- **Distance Calculation:** Applying the Haversine formula to calculate the distance between transactions and merchant locations based on their longitudes and latitudes.
- **Demographic Processing:** Counting age from date of birth and creating meaningful demographic groupings.

Listing 4.1: Feature Engineering Implementation

```
def feature_engineering(df):  
    # Calculate age from date of birth  
    df[ 'age' ] = df[ 'dob' ].apply(lambda x: calculate_age(x))
```

```

# Extract temporal features
df[ 'hour' ] = df[ 'timestamp' ].apply(lambda x: datetime.fromtimestamp(x).hour)
df[ 'day' ] = df[ 'timestamp' ].apply(lambda x: datetime.fromtimestamp(x).day)

# Calculate distance using Haversine formula
df[ 'distance' ] = df.apply(
    lambda row: haversine(row[ 'lat' ], row[ 'long' ],
                           row[ 'merch_lat' ], row[ 'merch_long' ]),
    axis=1
)
return df

# Distance calculation
def calculate_distance(lat1, lon1, lat2, lon2):
    # Haversine formula implementation
    R = 6371 # Earth radius in km

    dLat = np.radians(lat2 - lat1)
    dLon = np.radians(lon2 - lon1)

    a = (np.sin(dLat/2) * np.sin(dLat/2) +
          np.cos(np.radians(lat1)) * np.cos(np.radians(lat2)) *
          np.sin(dLon/2) * np.sin(dLon/2))

    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))

return R * c

```

Table 4.1: Feature Description Table After Data Processing

<b>Field Name</b>	<b>Description</b>
merchant	Name of merchant where transaction occurred
category	Category of merchant (e.g., travel, shopping_net, etc.)
amt	Amount of transaction (e.g., \$25.50)
gender	Gender of credit card holder (e.g., M(Male), F(Female))
lat	Latitude location of credit card holder (e.g., 34.0522)
long	Longitude location of credit card holder (e.g., -118.2437)
city_pop	Population of credit card holder's city (e.g., 4000000)
merch_lat	Latitude location of merchant (e.g., 34.0522)
merch_long	Longitude location of merchant (e.g., -118.2437)
age	Age of credit card holder (derived from dob, e.g., 28)
age_group	Age group category of credit card holder (e.g., 20-29, 30-39)
dist	Distance from transaction to merchant (e.g., 5.2 miles)
hour	Hour of day when transaction occurred (e.g., 14)
day_of_week	Day of the week when transaction occurred (e.g., Monday)
month	Month when transaction occurred (e.g., April)

The preprocessing stage includes:

- **Categorical Encoding:** Transforming the categorical variables through ordinal encod-

ing.

- **Handling Class Imbalance:** Applying **SMOTE** from **imblearn** library to address serious imbalance in fraud data.
- **Feature Standardisation:** Using **RobustScaler** from **sklearn** library to keep resistant to outliers when standardise features.

Listing 4.2: Preprocessing Implementation

```
# Import necessary libraries
from sklearn.preprocessing import RobustScaler
from imblearn.over_sampling import SMOTE

def pre_processing(df):
    # Standardise features
    scaler = RobustScaler()
    X_scaled = scaler.fit_transform(X)

    # Apply SMOTE to handle imbalance (only apply for the train set)
    smote = SMOTE(random_state=42)
    X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

This processing pipeline ensures consistent data preparation in all experiments, enhances reproducing ability and enables appropriate comparison between models. These data processing steps share the same pipeline before transferring to the model (which are applied to separate notebooks), so all the mutual functions of data collecting and processing data are implemented in ‘utils.py’ (Lai, 2025a)

#### 4.1.2 XAI Integration Components

The XAI integration structure includes several explanation methods to provide wide insights into model decisions. The implementation focuses on three explainers of three XAI methods:

- **SHAP explainer:** **KernelExplainer** provided in **shap** library was implemented for local explanations through force plots, summary plots and dependence plots.
- **LIME explainer:** **LimeTabularExplainer** provided in **lime** library for integrating local explanation of individual predictions.
- **Anchors explainer:** **AnchorTabularExplainer** provided in **anchor-exp** library for rule-based explanation that provides high-accurate insight.

Listing 4.3: XAI Integration Implementation

```
import shap
from lime import lime_tabular
from anchor import anchor_tabular

# SHAP integration
def get_shap_explanation(model, sample, background_size=100):
    background = shap.utils.sample(X_train, background_size)

    kernel_explainer = shap.KernelExplainer(model, background)
    shap_explainer = kernel_explainer(sample)

    # Return dict of feature:importance pairs
```

```

shap_dict = {
    feature: value
    for feature, value in
        zip(shap_explainer.feature_names, shap_explainer.values.squeeze())
}
return shap_dict, shap_explainer

# LIME integration
lime_tabular_explainer = lime_tabular.LimeTabularExplainer(
    training_data=X_train,
    class_names=['non-fraud', 'fraud'],
    feature_names=features,
    categorical_names=categorical_names,
    kernel_width=np.sqrt(len(features)) * 0.75
) # Only need to perform the perturbations once with the train set

def get_lime_explanation(model, sample):
    lime_explainer = lime_tabular_explainer.explain_instance(
        sample,
        model.predict
    )

    # Extract feature importances from LIME explanation
    lime_dict = {
        features[idx]: value
        for idx, value in dict(lime_explainer.as_map()[1]).items()
    }

    return lime_dict, lime_explainer

# Anchors integration
anchor_tabular_explainer = anchor_tabular.AnchorTabularExplainer(
    train_data=X_train,
    class_names=['non-fraud', 'fraud'],
    feature_names=features,
    categorical_names=categorical_names
) # Initialise Anchors explainer with the sample

def get_anchors_explanation(model, sample):
    # Get explanation for the first instance
    anchors_explainer = explainer.explain_instance(sample, model.predict)

    # Create a dictionary of feature binary importance based on rules
    anchor_dict = {
        feature: 1.0 if
            feature in [features[idx] for idx in anchors_explainer.features()]
        else 0.0
        for feature in features
    }

    return anchor_dict, anchors_explainer

```

This implementation approach is based on the framework and libraries proposed by Lundberg and Lee (2017), Ribeiro et al. (2018), and Ribeiro et al. (2016). Raufi et al. (2024) also emphasised the importance of combining these several XAI techniques for better credit card fraud interpretation.

#### 4.1.3 System Interaction Diagram

The system components interact through a carefully designed workflow that ensures uninterrupted data flow and model integration. Figure 4.1 illustrates these interactions.

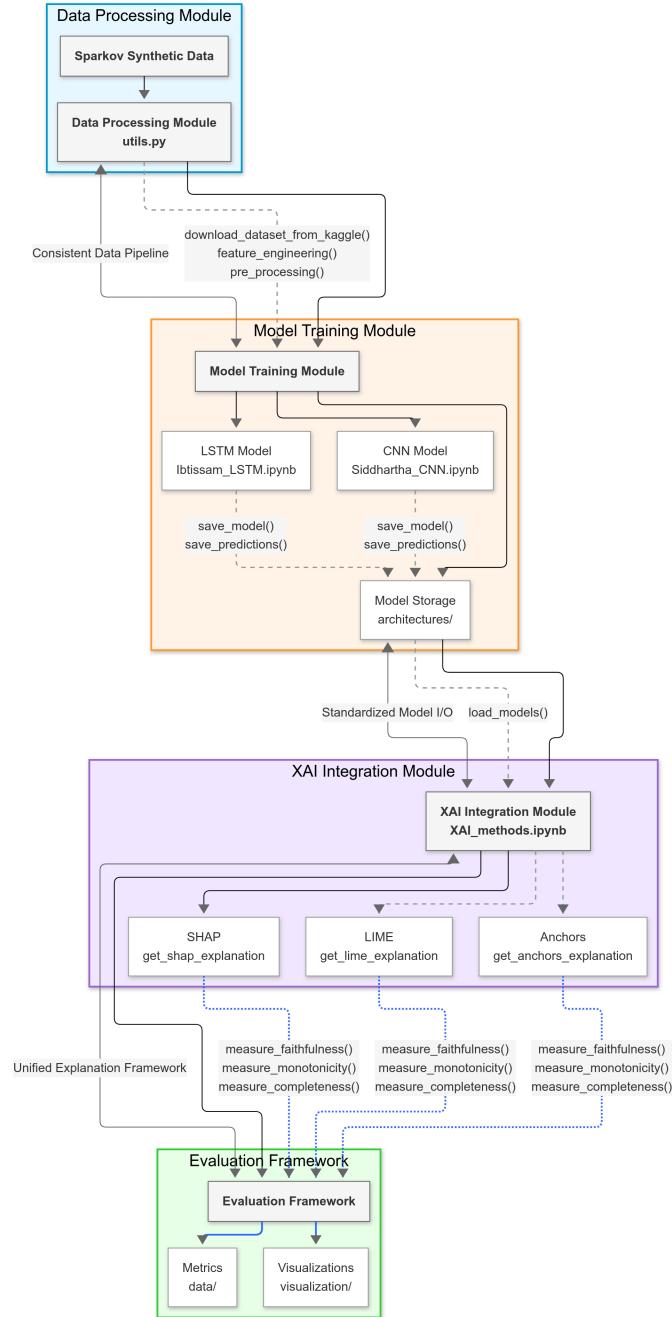


Figure 4.1: System Component Interactions

The system architecture facilitates:

- Modular designs with separation of concerns between data processing, model training and explanation generation
- Consistent data pipelines ensuring compatible transformations between components
- Standardised model input/output interfaces enable the spontaneous integration of the models
- Unified explanation framework allows comparison performance between various XAI methods

## 4.2 CNN Model Implementation

### 4.2.1 Detailed Architecture

The CNN model architecture was specifically designed to process 15 input features through a series of convolutional layers (3.3.1). Figure 4.2 shows the model architecture.

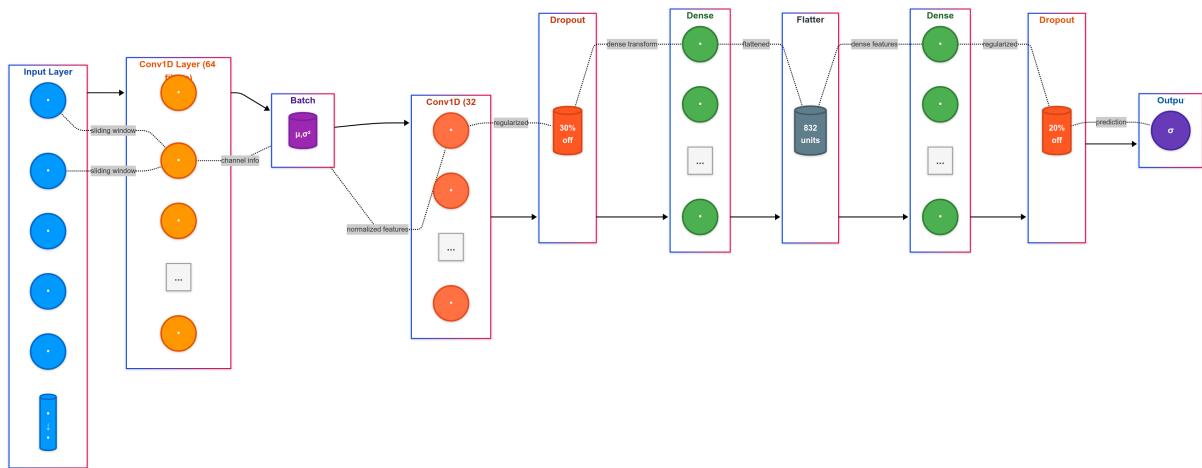


Figure 4.2: CNN Model Architecture

The implemented architecture consists of:

Listing 4.4: CNN Model Architecture Implementation

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, BatchNormalization
from tensorflow.keras.layers import MaxPooling1D, Flatten, Dense, Dropout

# Define CNN model
cnn_model = Sequential([
    # First convolutional block
    Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(15, 1)),
    BatchNormalization(),

    # Second convolutional block
    Conv1D(filters=32, kernel_size=2, activation='elu'),
    Dropout(0.3),
    ...]
```

```

# Fully connected layers
Flatten(),
Dense(64, activation='relu'),
Dropout(0.3),
Dense(1, activation='sigmoid')
])

```

The main design options include:

- **Convolutional Layers:** Two Conv1D layers with 64 and 32 filters respectively using a small kernel size (2) to capture local patterns in sequential features.
- **Activation Functions:** ReLu activation for the first convolutional layer, and ELU (Exponential Linear Unit) for the second layer to address the vanishing gradient problem.
- **Regularisation Techniques:** Batch normalisation after the first convolutional layer and dropout (0.3) to prevent overfitting after the second convolutional and dense layer.
- **Output Layer:** A sigmoid activation function for binary classification.

#### 4.2.2 Training Process and Optimisation

The CNN model training process implemented optimisation techniques to increase performance and convergence:

Listing 4.5: CNN Model Training Implementation

```

from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

# Compile model
cnn_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Define early stopping
callback = EarlyStopping(
    monitor='val_loss',
    patience=4,
    restore_best_weights=True
)

# Train model
history = cnn_model.fit(
    X_train,
    y_train,
    epochs=50,
    batch_size=30000,
    validation_data=(X_test, y_test),
    callbacks=[callback]
)

```

The main optimisation strategies included:

- **Adam Optimiser:** Was implemented with a learning rate of 0.001, combined with AdaGrad and RMSProp advantages.
- **Batch Size:** Define as 30,000 for the balance between computational efficiency and shield estimation accuracy (as this project utilises Tensorflow-GPU performance).
- **Early Stopping:** Applied with a patience of 4 epochs to avoid overfitting, ensuring proper training.
- **Binary Cross-Entropy Loss:** Selected as a proper loss function for this binary fraud classification task.

Figure 4.3 shows the training and validation loss and accuracy curves.

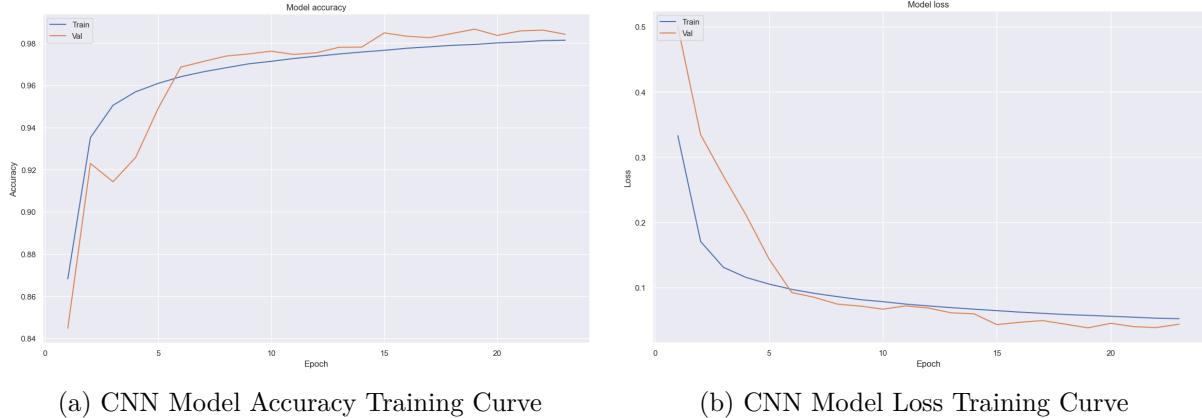


Figure 4.3: CNN Model Training and Validation Curves

#### 4.2.3 Performance Metrics

The CNN model achieved good performance metrics on the test set, Table 4.3 shows summarised metrics obtained by the CNN model.

Table 4.2: CNN Model Performance Metrics

Metric	Value
Accuracy	98.6%
ROC AUC	0.994
<b>Class 0 (Non-fraud)</b>	
Precision	0.999
Recall	0.987
F1-Score	0.993
<b>Class 1 (Fraud)</b>	
Precision	0.213
Recall	0.918
F1-Score	0.346

The performance metrics show that the CNN model performed well in detecting non-fraudulent transactions (high precision and recall for Class 0). For fraud transactions (class 1), the model receives high recall (0.918), showing that it successfully identifies most cases of fraud, although with low precision (0.213). This performance aligns with conclusions by Gambo et al. (2022), who observed the same great pattern in the CNN model to detect fraud in credit card transactions.

## 4.3 LSTM Model Implementation

### 4.3.1 Architecture with Attention Mechanism

The LSTM model involves an attention mechanism to increase its performance on sequential transaction data (Lai, 2025b). Figure 4.4 shows the model architecture.

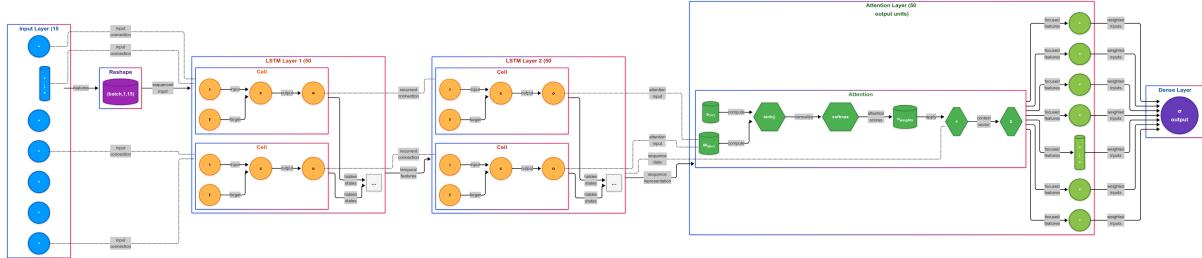


Figure 4.4: LSTM Model Architecture with Attention Mechanism

The implemented architecture consists of:

Listing 4.6: LSTM Model Architecture Implementation

```
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K

# Custom Attention Layer
class attention(Layer):
    def __init__(self, **kwargs):
        super(attention, self).__init__(**kwargs)

    def build(self, input_shape):
        self.W = self.add_weight(shape=(input_shape[-1], 1),
                               initializer='random_normal',
                               trainable=True,
                               name='attention_weight')
        self.b = self.add_weight(shape=(input_shape[1], 1),
                               initializer='zeros',
                               trainable=True,
                               name='attention_bias')
        super(attention, self).build(input_shape)

    def call(self, x):
        # Calculate attention scores
        e = K.tanh(K.dot(x, self.W) + self.b)
        a = K.softmax(e, axis=1)
        output = x * a
        return K.sum(output, axis=1)

    def compute_output_shape(self, input_shape):
        return (input_shape[0], input_shape[-1])
```

```

# Reshape data for LSTM input
X_train_lstm = X_train_resampled.reshape(X_train_resampled.shape[0], 15, 1)
X_test_lstm = X_test.reshape(X_test.shape[0], 15, 1)

# Define LSTM model with attention
input_layer = Input(shape=(15, 1))
lstm_layer1 = LSTM(50, return_sequences=True, dropout=0.3, recurrent_dropout=0.2)
lstm_layer2 = LSTM(50, return_sequences=True, recurrent_dropout=0.2)(lstm_layer1)
attention_layer = attention()(lstm_layer2)
output_layer = Dense(1, activation='sigmoid')(attention_layer)

lstm_model = Model(inputs=input_layer, outputs=output_layer)

# Model summary
lstm_model.summary()

```

The main design options include:

- **LSTM Layers:** Two stacked LSTM layers with 50 units each, and maintain sequential information for the attention mechanism by using `return_sequences=True`.
- **Attention Mechanism:** A custom attention layer that learns to focus on the most relevant parts of the sequence to detect fraud.
- **Regularisation:** Dropout (0.3) and recurrent dropout (0.2) to avoid overfitting.
- **Output Layer:** A sigmoid activation function for binary classification.

This architecture was inspired by the research of Ibtissam et al. (2021), who demonstrated that the LSTM model with attention mechanisms could effectively capture temporary patterns in transactions data to detect the activities of fraud.

#### 4.3.2 Training Process and Challenges

The LSTM model training process faced several challenges related to sequence handling and convergence:

Listing 4.7: LSTM Model Training Implementation

```

from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

# Compile model
lstm_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Define early stopping
callback = EarlyStopping(
    monitor='val_loss',
    patience=4,
    restore_best_weights=True
)

```

```

# Train model
history = lstm_model.fit(
    X_train,
    y_train,
    epochs=50,
    batch_size=30000,
    validation_data=(X_test, y_test),
    callbacks=[callback]
)

```

The training process involved:

- **Adam Optimiser:** Implemented with a learning rate of 0.001, similar to the CNN model.
- **Batch Size:** Set to 30,000 to maintain consistency with the CNN training approach.
- **Early Stopping:** Implemented with a patience of 4 epochs, which triggered after 26 epochs.

Several challenges emerged during the LSTM training process:

- **Slow Convergence:** The LSTM model required more epochs to reach optimal performance compared to the CNN model.
- **Memory Constraints:** The recurrent nature of LSTM layers increased memory requirements, necessitating optimised batch processing.

Figure 4.5 shows the training and validation loss and accuracy curves.

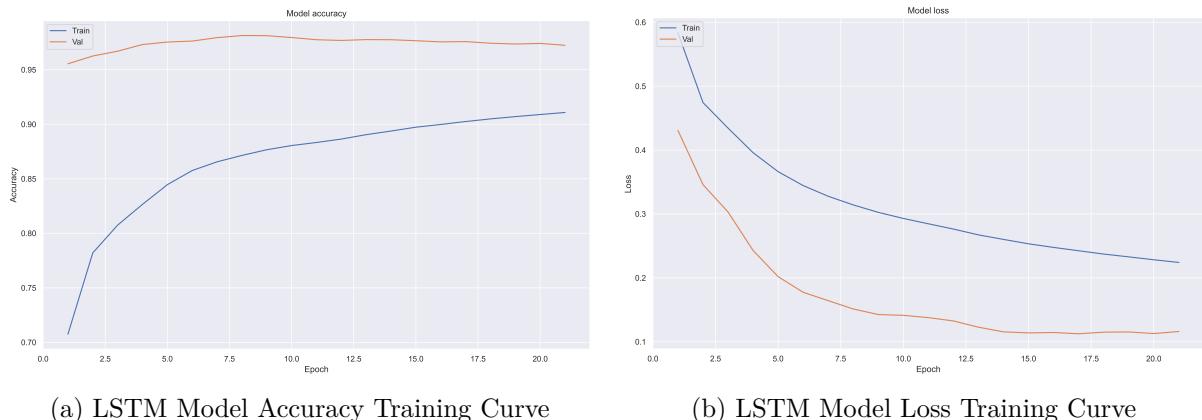


Figure 4.5: LSTM Model Training and Validation Curves

### 4.3.3 Performance Metrics

The LSTM model achieved robust performance metrics on the test set. Table ?? summarises the performance metrics achieved by the LSTM model.

Table 4.3: CNN Model Performance Metrics

Metric	Value
Accuracy	97.6%
ROC AUC	0.971
<b>Class 0 (Non-fraud)</b>	
Precision	0.999
Recall	0.976
F1-Score	0.987
<b>Class 1 (Fraud)</b>	
Precision	0.118
Recall	0.814
F1-Score	0.206

The LSTM model shows strong overall performance, with excellent accuracy (97.6%) and ROC AUC (0.971). For fraudulent transactions, it achieves a recall of 0.814, highlighting the persistent challenge of false positives in fraud detection. This performance is consistent with findings by Ibtissam et al. (2021), who observed that LSTM models excel at capturing temporal patterns.

# Bibliography

- Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N., & Herrera, F. (2023). Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, 99, 101805. <https://doi.org/https://doi.org/10.1016/j.inffus.2023.101805>
- Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58, 82–115. <https://doi.org/https://doi.org/10.1016/j.inffus.2019.12.012>
- Bowyer, K. W., Chawla, N. V., Hall, L. O., & Kegelmeyer, W. P. (2011). SMOTE: synthetic minority over-sampling technique. *CoRR*, *abs/1106.1813*. <http://arxiv.org/abs/1106.1813>
- Breskuvienė, D., & Dzemyda, G. (2024). Enhancing credit card fraud detection: Highly imbalanced data case. *Journal of Big Data*, 11(182). <https://doi.org/10.1186/s40537-024-01059-5>
- Cherif, A., Badhib, A., Ammar, H., Alshehri, S., Kalkatawi, M., & Imine, A. (2023). Credit card fraud detection in the era of disruptive technologies: A systematic review. *Journal of King Saud University - Computer and Information Sciences*, 35(1), 145–174. <https://doi.org/https://doi.org/10.1016/j.jksuci.2022.11.008>
- Du, H., Lv, L., Wang, H., & Guo, A. (2024). A novel method for detecting credit card fraud problems. *PLoS ONE*, 19(3), e0294537. <https://doi.org/10.1371/journal.pone.0294537>
- Financial Conduct Authority. (2024, March). *Using synthetic data in financial services* [Report published 8 March 2024]. Financial Conduct Authority. Retrieved April 24, 2025, from <https://www.fca.org.uk/publication/corporate/report-using-synthetic-data-in-financial-services.pdf>
- Găbudeanu, L., Brici, I., Codruta, M., Mihai, I.-C., & Scheau, M. (2021). Privacy intrusiveness in financial-banking fraud detection. *Risks*, 9, 104. <https://doi.org/10.3390/risks9060104>
- Gambo, M. L., Zainal, A., & Kassim, M. N. (2022). A convolutional neural network model for credit card fraud detection. *2022 International Conference on Data Science and Its Applications (ICoDSA)*, 198–202. <https://doi.org/10.1109/ICoDSA55874.2022.9862930>
- Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 80–89. <https://doi.org/10.1109/DSAA.2018.00018>
- Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision-making and a “right to explanation”. *AI Magazine*, 38(3), 50–57. <https://doi.org/10.1609/aimag.v38i3.2741>
- Grover, P., Xu, J., Tittelfitz, J., Cheng, A., Li, Z., Zablocki, J., Liu, J., & Zhou, H. (2023). Fraud dataset benchmark and applications. <https://arxiv.org/abs/2208.14417>

- Hafez, I. Y., Hafez, A. Y., Saleh, A., et al. (2025). A systematic review of ai-enhanced techniques in credit card fraud detection. *Journal of Big Data*, 12(6). <https://doi.org/10.1186/s40537-024-01048-8>
- Harris, B. (n.d.). *Generate fake credit card transaction data, including fraudulent transactions*. GitHub. Retrieved April 23, 2025, from [https://github.com/namebrandon/Sparkov\\_Data\\_Generation](https://github.com/namebrandon/Sparkov_Data_Generation)
- Ibtissam, B. (2021). *Lstm-attention-fraudetection*. GitHub. Retrieved April 22, 2025, from <https://github.com/bibtissam/LSTM-Attention-FraudDetection>
- Ibtissam, B., Samira, D., Bouabid, E. O., & Jaafar, J. (2021). Enhanced credit card fraud detection based on attention mechanism and lstm deep model. *Journal of Big Data*, 8(151). <https://doi.org/10.1186/s40537-021-00541-8>
- Kadir, M. A., Mosavi, A., & Sonntag, D. (2023). Evaluation metrics for xai: A review, taxonomy, and practical applications. *2023 IEEE 27th International Conference on Intelligent Engineering Systems (INES)*, 000111–000124. <https://doi.org/10.1109/INES59282.2023.10297629>
- Kamalaruban, P., Pi, Y., Burrell, S., Drage, E., Skalski, P., Wong, J., & Sutton, D. (2024). Evaluating fairness in transaction fraud models: Fairness metrics, bias audits, and challenges. <https://arxiv.org/abs/2409.04373>
- Lai, T. M. (2025a). *Credit card transaction fraud detection using explainable ai*. Retrieved April 25, 2025, from <https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI>
- Lai, T. M. (2025b). *Ibtissam's lstm-attention model implementation*. Retrieved April 22, 2025, from [https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI/blob/main/Ibtissam\\_LSTM.ipynb](https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI/blob/main/Ibtissam_LSTM.ipynb)
- Lai, T. M. (2025c). *Siddhartha's cnn model implementation*. Retrieved April 22, 2025, from [https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI/blob/main/Siddhartha\\_CNN.ipynb](https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI/blob/main/Siddhartha_CNN.ipynb)
- Lai, T. M. (2025d). *Xai methods implementation in deep learning models for credit card fraud detection*. Retrieved April 22, 2025, from [https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI/blob/main/XAI\\_methods.ipynb](https://github.com/ThongLai/Credit-Card-Transaction-Fraud-Detection-Using-Explainable-AI/blob/main/XAI_methods.ipynb)
- Lundberg, S. M., & Lee, S. (2017). A unified approach to interpreting model predictions. *CoRR*, *abs/1705.07874*. [http://arxiv.org/abs/1705.07874](https://arxiv.org/abs/1705.07874)
- Ma, S. (2024). Towards human-centered design of explainable artificial intelligence (xai): A survey of empirical studies. <https://arxiv.org/abs/2410.21183>
- Mienye, I. D., & Jere, N. (2024). Deep learning for credit card fraud detection: A review of algorithms, challenges, and solutions. *IEEE Access*, 12, 96893–96910. <https://doi.org/10.1109/ACCESS.2024.3426955>
- Paulraj, B. (2024). Machine learning approaches for credit card fraud detection: A comparative analysis and the promise of 1d convolutional neural networks. *2024 7th International Conference on Information and Computer Technologies (ICICT)*, 82–92. <https://doi.org/10.1109/ICICT62343.2024.00020>
- Pawlicki, M., Pawlicka, A., Uccello, F., Szelest, S., D'Antonio, S., Kozik, R., & Choraś, M. (2024). Evaluating the necessity of the multiple metrics for assessing explainable ai: A critical examination. *Neurocomputing*, 602, 128282. <https://doi.org/https://doi.org/10.1016/j.neucom.2024.128282>
- Pozzolo, A. D., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. *2015 IEEE Symposium Series on Computational Intelligence*, 159–166. <https://doi.org/10.1109/SSCI.2015.33>
- Raufi, B., Finnegan, C., & Longo, L. (2024). A comparative analysis of shap, lime, anchors, and dice for interpreting a dense neural network in credit card fraud detection. In L.

- Longo, S. Lapuschkin, & C. Seifert (Eds.), *Explainable artificial intelligence* (pp. 365–383). Springer Nature Switzerland.
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). <https://doi.org/10.1609/aaai.v32i1.11491>
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. *CoRR, abs/1602.04938*. <http://arxiv.org/abs/1602.04938>
- Ryman-Tubb, N. F., Krause, P., & Garn, W. (2018). How artificial intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. *Engineering Applications of Artificial Intelligence*, 76, 130–157. <https://doi.org/https://doi.org/10.1016/j.engappai.2018.07.008>
- Saeed, W., & Omlin, C. (2023). Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 263, 110273. <https://doi.org/https://doi.org/10.1016/j.knosys.2023.110273>
- Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services*, 1, 1–10. <https://doi.org/10.48550/arXiv.1708.08296>
- Seth, P., & Sankarapu, V. K. (2025). Bridging the gap in xai-why reliable metrics matter for explainability and compliance. <https://arxiv.org/abs/2502.04695>
- Shenoy, K. (2021). *Credit card transactions fraud detection dataset*. Kaggle. Retrieved April 22, 2025, from <https://www.kaggle.com/datasets/kartik2112/fraud-detection>
- Siddhartha, P. (2023). *Credit card fraud detection ml*. GitHub. Retrieved April 22, 2025, from [https://github.com/siddharthapramanik771/CreditCardFraudDetectionML/blob/main/Credit\\_card\\_Fraud\\_Detection.ipynb](https://github.com/siddharthapramanik771/CreditCardFraudDetectionML/blob/main/Credit_card_Fraud_Detection.ipynb)
- Sundararajan, M., Taly, A., & Yan, Q. (2017, June). Axiomatic attribution for deep networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 3319–3328, Vol. 70). PMLR. <https://proceedings.mlr.press/v70/sundararajan17a.html>
- Sundararamaiah, M., Nagarajan, S. K. S., Mudunuru, K. R., & Remala, R. (2024). Unifying ai and rule-based models for financial fraud detection. *International Journal of Computer Trends and Technology*, 72, 61–68. <https://doi.org/10.14445/22312803/IJCTT-V72I12P107>
- Tomy, A., & Ojo, I. (2025). Explainable ai for credit card fraud detection: Bridging the gap between accuracy and interpretability. *World Journal of Advanced Research and Reviews*, 25, 1246–1256. <https://doi.org/10.30574/wjarr.2025.25.2.0492>
- UK Finance. (2022). *Annual fraud report 2022*. UK Finance. <https://www.ukfinance.org.uk/policy-and-guidance/reports-and-publications/annual-fraud-report-2022>
- Uwaezuoke, E. C., & Swart, T. G. (2024). An explainable deep learning model for credit card fraud detection. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.5015497>
- Vallarino, D. (2025, March). Ai-powered fraud detection in financial services: Gnn, compliance challenges, and risk mitigation. <https://doi.org/10.2139/ssrn.5170054>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need. <https://arxiv.org/abs/1706.03762>
- Vihurskyi, B. (2024). Credit card fraud detection with xai: Improving interpretability and trust. *2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 1–6. <https://doi.org/10.1109/ICDCECE60827.2024.10548159>
- West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, 57, 47–66. <https://doi.org/https://doi.org/10.1016/j.cose.2015.09.005>

- Zhou, Y., Li, H., Xiao, Z., & Qiu, J. (2023). A user-centered explainable artificial intelligence approach for financial fraud detection. *Finance Research Letters*, 58, 104309. <https://doi.org/https://doi.org/10.1016/j.frl.2023.104309>
- Zhu, M., Zhang, Y., Gong, Y., Xu, C., & Xiang, Y. (2024). Enhancing credit card fraud detection a neural network and smote integrated approach. <https://arxiv.org/abs/2405.00026>