A Novel Approach in Financial Fraud Detection: Addressing Data Imbalance with Prompt Engineering, Leveraging Large Language Model Embeddings for Fraud Scoring, and Cost-Effectiveness Optimization

by Jiayi Lin

B.S. in Actuarial Science, December 2012, University of Illinois at Urbana Champaign
M.S. in Statistics, December 2014, University of Illinois at Urbana Champaign
M.B.A in Business Administration, June 2021, University of Chicago

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

January 10, 2025

Praxis directed by

Domingo Rivera
Professorial Lecturer in Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that John Howard Doe has passed the Final Examination for the degree of Doctor of Engineering as of November 19, 2024. This is the final and approved form of the Praxis.

A Novel Approach in Financial Fraud Detection: Addressing Data Imbalance with Prompt Engineering, Leveraging Large Language Model Embeddings for Fraud Scoring, and Cost-Effectiveness Optimization

Jiayi Lin

Praxis Research Committee:

Domingo Rivera (Advisor), Professorial Lecturer in Engineering and Applied Science, Praxis Director

Amir Etemadi, Associate Professor of Engineering and Applied Science, Committee Chair

Buddha Nepal, Professorial Lecturer in Engineering and Applied Science, Committee Member

© Copyright 2025 by Jiayi Lin
All rights reserved

Dedication

I am deeply grateful to my family and many friends for their unwavering support throughout this journey. A special thank you goes to my mother, Weilian, whose steadfast encouragement made this work possible. During the challenging times of unexpected setbacks in 2024, she stood by my side with unwavering faith. Her words, “Don’t give up; you can tackle each task one at a time,” have inspired me time and again to persevere. It is with heartfelt gratitude that I dedicate this work to her.

Acknowledgements

I would like to extend my heartfelt gratitude to all the SEAS faculty members for their invaluable support and guidance. Thank you for your dedication and commitment in helping me complete this independent study. This program has been an incredible journey, allowing me to learn advanced methodologies that I could immediately apply in my work and see impactful results in the industry.

A special thank you to my advisor, Dr. Domingo Rivera. His profound knowledge, patience, encouragement, and thoughtful suggestions guided me step by step toward achieving my goals. His mentorship has been instrumental in the completion of this research, and I am deeply grateful for his support.

Abstract of Praxis

A Novel Approach in Financial Fraud Detection: Addressing Data Imbalance with Prompt Engineering, Leveraging Large Language Model Embeddings for Fraud Scoring, and Cost-Effectiveness Optimization

This research examines novel methodologies that leverage large language models (LLMs) for synthetic data generation and embedding tasks to enhance the effectiveness of financial fraud detection. The study also introduces an optimized approach for balancing false negatives and false positives, with a focus on cost-effectiveness, ultimately supporting users in making practical, data-driven decisions.

Fraudulent transaction data is often highly imbalanced, necessitating specialized data processing methods to address this challenge. This study employs 14 traditional machine learning and deep learning classifiers alongside six data processing methods to develop fraud detection models and identify the most effective traditional approach for detecting fraudulent transactions. Building on this foundation, the research integrates advanced techniques such as OpenAI, BERT, LLaMA, Gemini, and Gemma to create hybrid models that incorporate categorical variable embeddings within machine learning classifiers. The objective is to enhance fraud detection capabilities beyond traditional methods, extending the work of Bakumenko et al. (2024) (Bakumenko, Hlaváčková-Schindler , Plant , & Hubig, 2024). Furthermore, the study introduces a novel hybrid oversampling technique that combines LangChain, GPT-4o-Mini, and advanced oversampling methods to address data imbalance and improve model performance. To

optimize model accuracy, a customized metric is developed to fine-tune hyperparameters by balancing the trade-off between false positives and false negatives. This research advances fraud detection methodologies through three key innovations: 1) Development of hybrid models leveraging LLM-based text embedding, 2) Creation of a hybrid oversampling technique incorporating LLM-based few-shot prompted synthetic data generation, 3) Implementation of a cost-effective optimization strategy to balance false positives and false negatives effectively.

Another aspect of this research focuses on both commercial API and publicly licensed light-weighted large language models. This research aims to find the best-in-class methodology to improve financial fraud detection ability within the cybersecurity domain.

Table of Contents

Dedication iv

Acknowledgements v

Abstract of Praxis vi

List of Figures xii

List of Tables xiii

List of Equations xiv

List of Symbols xv

List of Acronyms xvi

Chapter 1—Introduction 1

1.1 Background 1

1.2 Research Motivation 3

1.3 Problem Statement 5

1.4 Thesis Statement 6

1.5 Research Objectives 7

1.6 Research Questions and Hypotheses 8

1.7 Scope of Research 9

1.8 Research Limitations 9

1.9 Organization of Praxis 10

Chapter 2—Literature Review 12

2.1 Introduction 12

2.2 Machine Learning and Imbalance Class in Detecting Fraud Events 14
2.3 Deep Learning in Detecting Fraud Events 17
2.4 Large Language Models or Transformer in Detecting Fraud Events 18
2.5 Transformer Architecture Overview 20
2.5.1 Positional Encoding 22
2.5.2 Attention 22
2.5.3 Feed Forward Networks 24
2.6 Large Language Models 25
2.6.1 Popular Large Language Models 25
2.6.2 Prompt Engineering 28
2.6.3 Hyperparameter Tuning 29
2.6.4 Retrieval Augmented Generation 30
Chapter 3—Methodology 32
3.1 Introduction 32
3.2 Data Description 34
3.3 Research Design 43
3.4 Traditional Methods to Detect Fraud Events 45
3.4.1 Traditional Oversampling Techniques 45
3.4.2 Binary Classification 48
3.4.3 Anomaly Detection 51
3.4.4 Deep Learning 52
3.5 LLM Base Fraud Event Generation for Oversampling 53
3.5.1 Fine-Tune LLMs 54

3.5.2 Pattern Recognition and Text Summarization with LLMs 54
3.5.3 Addressing Data Imbalance with Synthetic Oversampling 57
3.6 Hybrid Models Combining LLMs 58
3.6.1 Categorical Embeddings 58
3.6.2 Principal Component Analysis of Embeddings Outputs 59
3.6.4 Hybrid Models Utilizing Large Language Models 60
3.7 Evaluating Model Performance 60
3.7.1 Key Performance Metrics for Model Evaluation 61
3.7.2 Training Model and Prediction Process 63
3.8 Fraud Detection Optimization Methodology 63
3.8.1 Analysis of False-Positive Rate Study 64
3.8.2 Analysis of False-Negative Rate Study 65
3.8.3 Optimizing Fraud Risk Management Through Iterative Improvements 65
3.9 Limitations and Challenges 66
3.9.2 System Requirements and Execution Costs 67
3.9.3 OpenAI Data Generator Limitation 67
3.9.4 Assumption to Cover all Fraud Patterns in the Few-Slots Prompt 68
3.9.5 Data Privacy Risk 68
Chapter 4—Results 69
4.1 Introduction 69
4.2 LLM based Categorical Variable Embedding 69
4.2.1 Model Performance using Traditional Methods 69

4.2.2 Hybrid Model with LLM-Based Categorical Embedding Variables 76
4.3 LLM-Based Synthetic Data Generation 81
4.3.1 Synthetic Data 81
4.3.2 Fine-Tuned GPT-4o Mini Model 81
4.3.3 Few-shot Prompt 82
4.3.4 Classifier Result Comparison 84
4.4 Optimizing Cost-Efficiency in Hyperparameter Tuning 85
4.4.1 Customized Metric 85
4.4.2 Bayesian Search Hyperparameter Tuning 86
4.4.3 Final Model Selection 87
Chapter 5—Discussion and Conclusions 89
5.1 Discussion 89
5.2 Conclusions 90
5.3 Contributions to Body of Knowledge 91
5.4 Recommendations for Future Research 92
References 94
Appendix A 103

List of Figures

Figure 1. Fraud Detection Method Evolution 2

Figure 2. Fraud Detection Research 15

Figure 3. Deep Learning Research 18

Figure 4. Transformer Architecture 21

Figure 5. Python Scripts Process Chart 34

Figure 6. Distribution of transaction count by Fraud Status 36

Figure 7. KDE Plot of Transaction Amount by Fraud Status 37

Figure 8. Number of Fraud Transactions Over Time 38

Figure 9. Correlation Heap Map 40

Figure 10. LightGBM Cumulative Feature Importance Plot 42

Figure 11. Fraud Patterns Generated by Fine-Tuned GPT-4o Mini Model 56

Figure 12. Initial Batch of Synthetic Data 57

List of Tables

Table 1. Classification models comparison with imbalanced data 70

Table 2. Classification models comparison with random oversampling data 71

Table 3. Classification models comparison with SMOTE oversampling data 71

Table 4. Classification models comparison with borderline SMOTE oversampling data 71

Table 5. Classification models comparison with ADASYN oversampling data 72

Table 6. Classification models comparison with SMOTETOMEK oversampling data ... 72

Table 7. Classification models comparison with SMOTEEN oversampling data 72

Table 8. Isolation Forest and COPOD Result Comparison 74

Table 9. CNN and RNN Result Comparison 75

Table 10. Hybrid Gemma - Based Classification Comparison 77

Table 11. Hybrid LLaMA 3.2-Based Classification Comparison 77

Table 12. Hybrid Bert-Based Classification Comparison 78

Table 13. OpenAI Embedding Classification Comparison 79

Table 14. Gemini Embedding Classification Comparison 80

Table 15. Few-Slot Prompt 83

Table 16. Synthetic Data Generation 84

List of Equations

Equation 1. Sine Positional Encodings 22

Equation 2. Cosine Positional Encodings 22

Equation 3. Softmax Function 23

Equation 4. Multi-head Attention 24

Equation 5. Feed-forward Network 25

Equation 6. Synthetic Minority Oversampling Technique (SMOTE) Data Generation .. 46

Equation 7. Synthetic Data Generation Example 47

Equation 8. Accuracy 61

Equation 9. Recall 61

Equation 10. precision 62

Equation 11. F1-Score 62

Equation 12. Kappa 62

Equation 13. Matthews Correlation Coefficient 62

Equation 14. Total Loss 66

Equation 15. Customized Metric 85

List of Symbols

F1	F1 Score (Harmonic Mean of Precision and Recall)
FFN	Feed-forward Network
FN	False Negative Count
FP	False Positive Count
K	Key Matrix
PE	positional encodings
Q	Query Matrix
V	Value
α	Cost to correct a false positive case
β	Cost to correct a false negative case
Ω	Customized metric

List of Acronyms

ADASYN	Adaptive Synthetic
AUC	Area under the Curve
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
COPOD	Copula-Based Outlier Detection
DL	Deep Learning
GNN	Graph Neural Networks
GPT	Generative Pre-Trained Transformer
GRU	Gated Recurrent Unit
LLaMA	Large Language Model Meta AI
LLM	Large Language Model
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
PCA	Principal Component Analytics
RNN	Recurrent Neural Network
ROS	Random Oversampling
SMOTE	Synthetic Minority Oversampling Technique
SMOTEEN	SMOTE with Edited Nearest Neighbors
SMOTETomek	SMOTE with Tomek Link

Chapter 1—Introduction

1.1 Background

Payment fraud remains a critical cyber threat in the banking industry, with new trends continuously emerging (Chase Bank, 2021). Fraud detection techniques have evolved significantly, from early statistical methods in 1968 (Lenard & Alam, 2009) to advanced transformer models in 2024 (Hao An, 2024). Over the past 56 years, fraud detection methods have diversified, incorporating machine learning (Dornadula & Geetha, 2019), deep learning techniques like RNNs and CNNs (Sulaiman, Nadher, & Hameed, 2023), Graph Neural Networks (Chen, Zhu, Yuan, & Huang, 2024), and large language models (LLMs) and generative AI solutions (An et al., 2024). Generative AI, in particular, holds significant promise for the financial sector, impacting areas like fraud prevention, anti-money laundering, and risk management (Kalia, 2023). In 2024, Mastercard's cybersecurity team leveraged generative AI to enhance fraud detection accuracy by up to 300% (Browne, 2024).

Payment fraud remains a critical cyber threat in the banking industry, with new trends continuously emerging (Chase Bank, 2021). Fraud detection techniques have evolved significantly, from early statistical methods in 1968 (Lenard & Alam, 2009) to advanced transformer models in 2024 (Hao An, 2024). Over the past 56 years, fraud detection methods have diversified, incorporating machine learning (Dornadula & Geetha, 2019), deep learning like RNN or CNN (Sulaiman, Nadher, & Hameed, 2023), Graph neural network (Chen, Zhu, Yuan, & Huang, 2024), OpenAI (Ahmadi, 2023) and other LLM or generative AI solutions (An, et al., 2024). Among all techniques,

generative AI methods have potential great impact on the financial industry, including fraud protection, anti-money laundering and risk management areas (Kalia, 2023). In 2024, cybersecurity from Mastercard used generative AI to boost fraud detection by up to 300% (Browne, 2024).

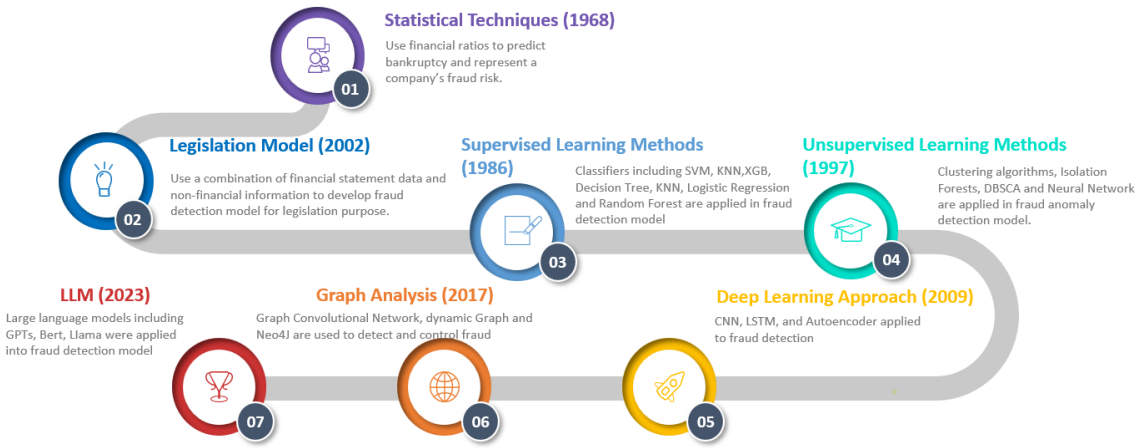

Figure 1. Fraud Detection Method Evolution

In 2024, more new Large Language Models were released, such as Google's Gemini 1.5 Pro, Flash, Gemma (Velloso & Woodward, 2024), Meta's LLaMA 3 (Meta Platform Inc., 2024), GPT-4o (OpenAI, Hello GPT-4o, 2024) . These pretrained models with different parameter sizes, have different advantages on efficiency and accuracy. The application of LLMs in industry production has shown polarizing effects. On the positive side, successful applications of LLMs include clinical diagnoses, mapping cybersecurity regulations to policies and controls, expediting claims processing, analyzing customer feedback, fielding customer services questions, enhancing developer productivity, detecting signs of malware, scraping online data. (Forbes Technology Council, 2024). Conversely, a fraudster could leverage generative AI to create AI-generated selfies, fake bank statements or other documents, cloning voices, or even generating deepfake videos

to perform fraud activity (Persona, 2024). With the advent of new LLMs in 2024, it is an opportune time to apply and compare these LLM models to enhance defenses against cybersecurity fraud attacks in the finance industry.

This research aims to leverage machine learning, large language models, retrieval augment generation to develop a novel method to improve cybersecurity fraud detection performance and compare the performance results with other advanced fraud detection methods using machine learning and deep learning.

1.2 Research Motivation

The banking industry is one of the sectors facing predominantly high fraud risk (Sanusi, Rameli, & Isa, 2015). Data released by the Federal Trade Commission in 2024 shows that Americans lost more than $10 billion to fraud in 2023, with approximately $3.5 billion attributed to payment losses including bank transfers, cryptocurrency, and wire transfers (Federal Trade Commission, 2024). According to the TransUnion 2024 State of Omnichannel Fraud report, cybercriminals steal identity information to commit account takeovers and perpetrate first-party fraud by creating fraudulent synthetic accounts. This has resulted in $3.1 billion in fraud lending exposure for US auto loans, credit cards, and unsecured personal loans (TransUnion, 2024).

Common fraud tactics such as unauthorized transactions, stolen financial information, spam, identity theft, and illegitimate chargebacks have led to a 354% year-over-year increase in account takeover attacks in Q2 2023. The fintech sector has been particularly affected, experiencing an 808% surge (Drenik, 2023). Equipped with generative AI tools such as ChatGPT, FraudGPT, WormGPT, and DarkBert,

cybercriminals can easily scale synthetic fake ID fraud, making it harder to detect (Bussler, 2023).

New fraud tactics using generative AI enable fraudsters to create synthetic identities, perform digital injection attacks, and generate deepfakes, posing a real-time challenge in recognizing new customers. Cybercriminals fabricate fake ID documents that combine stolen, real, and fake information, and use deepfake tools to create images that match legal ID photos. With these synthetic identities, fraudsters can manipulate facial authentication and ID verification systems. The fraud detection process, often reliant on in-house employees, is slow and struggles to keep up with the large scale of synthetic ID fraud (Hanif, 2024).

In addition, these new fraud tactics can be easily scaled up using automated scripted bots. Open forums and hackers offer bot attack services, significantly lowering the barrier to committing fraud for a relatively small fee (Drenik, 2023). Deloitte's Center for Financial Services forecasts that GenAI could drive fraud losses in the United States to soar from $12.3 billion in 2023 to $40 billion by 2027, reflecting a compound annual growth rate of 32% (Lalchand, Srinivas, Maggiore, & Henderson, 2024).

This research aims to leveraging large language models (LLMs) to improve fraud detection classifier beyond traditional techniques, with an emphasis on cost efficiency in selecting a final model that balances false positives and false negatives. As cybercriminals gradually adopt advanced technologies, the financial industry must keep pace, implementing more efficient methods to defend against fraud attacks.

1.3 Problem Statement

Despite the widespread use of machine learning methods in banking fraud protection, significant challenges remain, particularly when dealing with the highly imbalanced nature of financial fraud transaction data. This imbalance can lead to biased results, where models may fail to adequately detect fraudulent activities. Traditionally, two major approaches have been employed to address this issue:

1. **Effective Data Resampling and Engineering**: Techniques such as Random Oversampling (ROS), SMOTE, Borderline SMOTE, Adaptive Synthetic (ADASYN), SMOTETomek, and SMOTEEN have been used to balance the dataset before model training (El-Naby, Hemdan, & El-Sayed, 2022).
2. **Optimizing Deep Learning Models**: Adjusting deep learning models to account for the imbalanced data distribution has also been a common strategy (Alothman, Talib, & Mohammed, 2022).

However, these methods have limitations. They often fail to account for emerging fraud patterns not present in the original dataset, and the data processing techniques can significantly impact the overall performance of fraud detection models by training different classifier algorithms. As new fraud schemes evolve, the ability of these methods to adapt and maintain accuracy becomes increasingly critical, highlighting the need for more advanced and adaptive solutions in fraud detection.

Building on the future research directions outlined in Mienye's paper (Mienye & Jere, 2024), the development of a real-time fraud detection system necessitates an efficient data processing method that can swiftly and accurately respond to fraudulent

activities. This paper proposes the use of prompt engineering to oversample fraudulent events, thereby enhancing the system's ability to capture emerging fraud patterns. Additionally, it explores the integration of hybrid models that combine machine learning or deep learning techniques with different large language models, aiming to improve overall detection performance. To further optimize the system's deployment, the paper introduces a novel cost-efficient approach that leverages the new data processing method and hybrid models to minimize false positives.

1.4 Thesis Statement

There is a pressing need for the financial services sector to embrace emerging AI technologies, particularly large language models (LLMs), to proactively detect and respond to evolving fraud trends.

The first stage of this research focuses on developing fraud detection models to assess the performance of traditional machine learning and deep learning methods, alongside a variety of LLMs such as GPT, BERT, Gemma, LLaMA, and Gemini. Techniques like transfer learning will be employed to construct and refine the pre-trained LLM models.

In the second stage, the research aims to propose innovative solutions to enhance model performance and efficiency. This includes exploring model optimization, refining performance metrics, and recommending strategies for monitoring these models.

This study seeks to tackle the challenges associated with implementing new LLM technologies to strengthen banking fraud detection capabilities, effectively addressing emerging digital fraud trends. Specifically, it aims to investigate a hybrid approach

integrating machine learning and LLMs to enhance classification accuracy and reduce false positives, thereby improving overall customer experience.

1.5 Research Objectives

Objective 1: To enhance model performance, develop hybrid models that incorporate the latest pre-trained LLMs, such as GPT, BERT, Gemini, Gemma, and LLaMA. The objective is to compare their performance against other machine learning models. This approach will utilize cutting-edge techniques like prompt engineering, categorical variable embedding and fine-tuning to optimize the base models by tunning hyperparameters with a customized metric. Model performance will be rigorously evaluated using key performance metrics to assess the effectiveness of these enhancements. Additionally, the comparison will highlight cost-efficient models with reduced training time, ensuring both high performance and resource efficiency.

Objective 2: To improve the model performance and overcome the limitation of lack of historical data, leveraging large language models to develop an oversampling method to resolve data imbalance issues. Specially, this objective will be evaluated by comparing it to multiple data processing methods. Six traditional oversampling techniques along with a hybrid LLM-based synthetic data oversampling data processing will be applied to different classifiers.

Objective 3: Improve fraud detection capabilities on new data, develop and experiment with a novel method aimed at reducing the costs associated with using large language models (LLMs) while. A highly sensitive fraud detection model can lead to a high number of false positives, causing inconvenience for customers and negatively impacting their spending behavior. To address this, our approach involves combining

machine learning techniques with LLMs for fraud detection. The proposed method involves identifying false positive cases from the best-performing model and using them as training data to refine predictions using cost efficient model. This approach has the potential to reduce training costs and enhance the model's overall ability to detect fraud. An efficient strategy may involve using LLM hybrid methods to capture major fraud cases, while deploying traditional machine learning model to further minimize false positives, thereby optimizing both accuracy and cost-efficiency.

1.6 Research Questions and Hypotheses

RQ1: Can hybrid models that integrate machine learning with LLM-based embeddings for categorical variables outperform traditional machine learning utilizing traditional data processing techniques?

RQ2: Can large language models be leveraged to generate synthetic data that replicates fraud patterns, effectively balancing the fraud dataset and improving the performance of binary classifiers?

RQ3: Can the proposed hybrid model with generated synthetic data be further optimized by minimizing false positives and false negatives considering cost-efficiency?

H1: The proposed hybrid model, incorporating categorical embedding techniques from LLaMA3.2, Gemma, OpenAI, BERT, and Gemini, combined with traditional data balancing methods, achieves superior performance in classification tasks compared to other traditional algorithms.

H2: Integrating synthetic data generated by the fine-tuned GPT-4o Mini model for the minority class, followed by oversampling techniques, enhances the performance of the binary classification model.

H3: The proposed method could optimize model performance to further reduce false positive fraud cases and enhance the overall cost effectiveness of fraud detection.

1.7 Scope of Research

Domain: This research centers on leveraging large language models (LLMs) such as GPT, BERT, Gemma, LLaMA, and Gemini, as well as exploring hybrid models that combine LLMs with deep learning techniques, to enhance the classification of banking fraud attacks. The primary goal is to improve detection capabilities by integrating these transformer-based models with traditional fraud detection methods. Key aspects of the study include fine-tuning LLMs using Retrieval-Augmented Generation (RAG), optimizing tuning parameters based on evaluation metrics, developing a novel oversampling method to address data imbalance issues, and creating a cost-efficient optimization strategy to enhance overall model performance.

Traditional fraud detection modeling approach: This research compared the existing fraud detection modeling approaches to setup a baseline of fraud detection ability using traditional methods. This focus indicates the limit of fraud detection ability including model performance metrics and running time.

1.8 Research Limitations

The primary limitation of this research lies in the use of synthetic U.S. based Bank data, which, while utilized in this study, offers a restricted number of variables compared to authentic industrial data. This constraint hinders the ability to effectively

identify genuine fraud patterns over extended periods. Moreover, the Banking, Finance, Services, and Insurance (BFSI) sector is subject to rigorous compliance standards, including the Gramm-Leach-Bliley Act (GLBA) and the Fair Credit Reporting Act (FCRA), which impose stringent privacy regulations on financial institutions (Sandepudi, 2024). Smaller financial institutions may also face challenges in acquiring sufficient data to support robust big data-driven LLM solutions.

1.9 Organization of Praxis

This Praxis is structured into five chapters, each addressing different aspects of the research, followed by references and appendices.

Chapter 1 Introduction: This chapter provides the background, motivation, problem statements, thesis statement, research objectives and hypotheses, research domain, and limitations of the study.

Chapter 2 Literature Review: This chapter offers an overview of literature on methodologies in the finance sector, focusing on techniques for fraud detection. It explores recent advancements in transformer, new open-source LLM models, prompt engineering, retrieval augment generation and hyperparameter tuning.

Chapter 3 Methodology: This chapter outlines the methodology employed in this study, encompassing the application of traditional machine learning models, deep learning techniques, and LLMs in fraud detection models. Special topics include fine-tuning, transfer learning, retrieval-augmented generations (RAGs), experimental design, coding development, and novel methods utilized in the research.

Chapter 4 Results: This chapter comprehensively presents the outcomes of our model performance comparisons across various metrics and provides a synthesis of their

strengths and weaknesses. The analysis is structured into three main sections: (1) develop new oversampling methods to resolve imbalance issue and compare the performance result using this method vs existed data processing methods; (2) develop the most efficient and best performance LLM hybrid models and compare the performance result with other existed modeling methods; (3) develop an optimization method to lower false positive events and minimize cost.

Chapter 5 Discussion and Conclusions: This concluding chapter synthesizes the key findings and proposes recommendations for future research, including leveraging federated learning as a decentralized approach to address data privacy concerns in advanced modeling.

Chapter 2—Literature Review

2.1 Introduction

This chapter provides a comprehensive review of the existing literature on cybersecurity fraud risk and the methodologies employed in the financial industry. It also explores the application of various transformer models to address fraud detection challenges. The models examined in this Praxis are not limited to traditional machine learning and deep learning techniques but also encompass several large language models. These include the Generative Pretrained Transformer-4o (OpenAI, Hello GPT-4o, 2024) developed by OpenAI, the Large Language Model Meta AI 3.1 (Meta Platform, Inc., 2024), Bidirectional Encoder Representations from Transformers (BERT) (Google Research, 2018), Google's Exploratory Multimodal Machine Learning Algorithm (GEMMA) (AI Google, 2024), and the GEMINI models (Pichai & Hassabis, 2023) from Google.

The research presented in this Praxis is distinctive for its comparative analysis of machine learning, deep learning, and large language models (LLMs) in detecting fraud events. This study stands out in the field of fraud detection due to its innovative use of machine learning, deep learning and LLMs to initiate and enhance fraud detection scoring responses. It offers a thorough examination of various factors, including model types, model sizes, LLM types, the temperature hyperparameter, and potential areas for LLM adoption. The unique approach involves leveraging LLMs to improve fraud detection models by generating synthetic data to balance input, recognizing fraud suspect patterns, correcting false positive cases and identify potential suspect cases without large

historical samples. There are several literature papers discuss fraud detection models and LLM use cases relating to this Praxis.

Among these papers, the study “Comparing GPT and Traditional Machine Learning in Fraud Detection” (Trozelli & Anton, 2024) compares several machine learning models, including Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), Decision Tree (DT), Naïve Bayes (NB), and Extreme Gradient Boosting (XGBoost), with LLM approaches. It found that LLM approaches can be comparable to traditional machine learning models, offering faster implementation and lower cost. This Praxis adopts an extended approach by comparing LLMs with both machine learning and deep learning methods and proposing a novel approach that combines traditional machine learning or deep learning with LLMs to optimize decision performance and cost.

The paper “Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions” (Mienye & Jere, 2024) reviews numerous studies on fraud detection modeling using machine learning and deep learning techniques. It specifically describes different deep learning architectures, including Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), and evaluates their performance in credit card fraud detection. Additionally, the paper explores the application of BERT transformer models in this context and discusses future research directions that combine deep learning with transformer models to enhance performance. Extending this scope, the Praxis experiments with mixed techniques incorporating data balancing treatments and hybrid methods that blend deep learning and

transformer models. It also tests the latest advanced large language models (LLMs), such as LLaMA 3.1, Gemini, Gemma, and GPT-4o, in addition to BERT, to identify the optimal approach for fraud detection.

Another study, "Spatial-Temporal-Aware Graph Transformer for Transaction Fraud Detection" (Tian & Liu, 2024), introduces a new approach that combines Graph Neural Networks (GNN) and Transformers to address the limitations of the GNN structure. This method connects the target node with long-distance nodes to enhance the fraud detection algorithm. This Praxis focuses not only on the GNN-Transformer algorithm but also extends the performance comparison to various deep learning models, transformer models, and hybrid techniques.

2.2 Machine Learning and Imbalance Class in Detecting Fraud Events

The most common solution for detecting fraud events is using machine learning models to replace the rule-based approach. A prevalent issue with fraud datasets is class imbalance, where fraudulent transactions constitute a very small minority compared to non-fraud transactions (Mondal, Haque, Hassan, & Shatabda, 2021). Numerous studies have discussed solutions to address the class imbalance problem to effectively capture these rare fraud cases. The table below summarizes various methodologies and their limitations in dealing with data imbalance issues.

Authors and Year	Methodology	Results	Future Work or Limitations
(Xuan, et al., 2018)	Uses Random Forest algorithm with random sampling for the whole dataset	High accuracy in fraud detection	Random forest has good results on small set data but has issues on imbalanced data
(Ahammad, Hossain, & Shafiul Ala, 2020)	Uses both over sampling (SMOTE) and under sampling techniques	Enhanced performance of fraud detection models	Use different neural network (convolutional and artificial neutral network) and deep learning method to test a larger volume dataset
(El-Naby, Hemdan, & El-Sayed, 2022)	Comprehensive framework using serval data processing methods including random oversampling (ROS), SMOTE, Borderline SOMTE, Adaptive synthetic (ADASYN), SMOTETomek and SMOTEEN across algorithms KNN, LR, LDA, CART and NB	Borderline SMOTE gives the best performance with the CART algorithm	Tuning optimal parameters in different algorithms could control the sampling process. Apply deep learning after borderline SMOTE could further improve the performance
(Mondal, Haque, Hassan, & Shatabda, 2021)	Evaluates auPR and auROC for several models with GAN-based sampling including KNN, Logistic Regression, Random Forest, Bagging Classifier and ensemble model across two data processing methods SMOTE and ADASYN	Ensemble model after being oversampled by GAN has better performance to handle imbalance data. SMOTE is a more stable option than ADASYN and have better performance on both auROC and auPR	GAN-based sampling technique could be beneficially to apply to other domains
(Isangediok & Gajamannage, 2022)	Evaluates classifiers (LR, DT, RF, XGB) with SMOTE and ENN	Improved fraud detection with optimized tools	Logistic Regression less effective with extreme class imbalance
(Alothman, Talib, & Mohammed, 2022)	Use XGBoost and CatBoost algorithm on unbalanced data to detect credit fraud. Bayesian optimization is used to tune hyperparameters to increase the model's accuracy for the minority class	The proposed result outperformances other traditional machine learning models and data balancing method	Limitation of this study is only test on a single data set and a single optimization method
(Kennedy, Salekshahrezaee, Villanustre, & Khoshgoftaar , 2023)	Uses Autoencoders iteratively to clean and learn high-imbalanced data. Compare performance result of AUC and AUPRC among unsupervised fraud detection approaches like Isolation Forest and COPOD	The proposed method outperforms both isolation forest and COPOD using AUPRC and outperformances the baseline autoencoder in few iterations on two different datasets	Expand the result to other application domains and work on measuring the effectiveness of this approach on large balanced data. Future work can also explore potential early stopping techniques to train autoencoders that could reduce overall training time and keep significant performance

Figure 2. Fraud Detection Research

These studies illustrate that the performance of fraud detection models is closely tied to the data processing methods used to address class imbalances, as well as the chosen machine learning algorithms. Highly imbalanced fraud data can skew models toward the majority class, making it difficult to accurately predict fraudulent cases (Isangediok & Gajamannage, 2022). These papers explore the interactions between machine learning algorithms such as KNN, Logistic Regression, tree-based models, Naive Bayes, Isolation Forest, and COPOD, and different data processing strategies to enhance fraud detection. A common approach involves applying different oversampling strategies, such as Random Oversampling (ROS), SMOTE, Borderline SMOTE, ADASYN, SMOTETomek and SMOTEENN to the training data (Mondal, Haque, Hassan, & Shatabda, 2021), and then comparing the performance of the same machine learning algorithm with different data processing techniques. This approach is extended to various machine learning algorithms to identify the optimal combination of algorithm and oversampling strategy that yields the best performance metrics, such as AUC, AUPRC, and F1-score. Alothman's study concludes that the CatBoost algorithm, when paired with Bayesian hyperparameter optimization, can effectively address class imbalance without the need for special data processing techniques. However, it also finds that XGBoost with oversampling methods produces results similar to CatBoost with optimization. The study's limitation lies in its focus on a single dataset, suggesting that new algorithms could be developed for handling unbalanced data (Alothman, Talib, & Mohammed, 2022). Consequently, there is an opportunity to apply data processing techniques to other algorithms including deep learnings, LLMs and newly proposed hybrid algorithms in order to identify the optimal framework.

2.3 Deep Learning in Detecting Fraud Events

Recent advancements in fraud detection have led to the development of various deep learning algorithms designed to identify fraudulent activities. Commonly used models include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Graph Neural Networks (GNN). To further enhance these models, additional techniques such as reinforcement learning and ensemble methods have been introduced to address challenges like data imbalance and to improve overall performance.

In addition, comparative studies between traditional machine learning and deep learning approaches reveal that, for complex financial datasets, traditional models like Random Forest (RF) can outperform deep learning models, particularly in scenarios with infrequent fraudulent events (Boulsane & Afdel, 2024). Future research in this area includes developing hybrid models, implementing real-time fraud detection systems, optimizing reinforcement learning techniques, addressing data imbalance, and extending model generalization to other domains.

Below are summaries of four 2024 papers that apply deep learning methods to detect fraud events, along with approaches to address data imbalance issues during data processing.

Authors and Year	Methodology	Results	Future Work or Limitations
(Mienye & Jere, 2024)	Reviews deep learning techniques for credit card fraud detection, focusing on architectures like CNN, RNN, LSTM, and GRU. The study also evaluates performance	Deep learning models, especially LSTM and GRU, show high accuracy in detecting fraud.	Exploring hybrid models, improving interpretability, addressing data imbalance, and enhancing scalability and adaptability in real-world applications.

	metrics and discusses challenges like class imbalance.		
(Kong, et al., 2024)	Proposes CFTNet, a credit card fraud detection model using counterfactual data augmentation and a triplet Neural Network and reinforcement learning (P-DQN) to enhance learning of causal relationships.	CFTNet outperforms traditional methods in accuracy, recall, and AUC, demonstrating improved robustness across datasets with different sample ratios.	Extending the model's generalizability to other domains, integrating real-time processing capabilities, and exploring ensemble approaches to further enhance performance.
(Boulsane & Afdel, 2024)	The study compares the effectiveness of machine learning models, such as Random Forest (RF), with deep learning models like Neural Networks (NN), MLP, CNN, GRU, and RNN in detecting financial fraud on the Ethereum blockchain by accuracy and precision	The results indicate that Random Forest (RF) outperforms deep learning models, especially in complex financial datasets with infrequent fraudulent events.	The study emphasizes the importance of selecting the appropriate model for fraud detection. Future research could focus on real-time system integration and further exploration of model generalizability to other domains.
(Cheng, et al., 2024)	Proposes the GNN-CL model combining Graph Neural Networks (GNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks for financial fraud detection. The model incorporates a neighborhood noise purifier using Multi-layer Perceptrons (MLP), a core node intensifier with reinforcement learning, and a relationship summarizer to improve feature representation and reduce noise.	GNN-CL outperforms existing models like GCN, PC-GNN, and CARE-GNN across various metrics, including precision, recall, F1-score, and AUC.	Future work could focus on extending the GNN-CL model to other datasets and domains, exploring real-time processing capabilities, and further optimizing the reinforcement learning mechanism

Figure 3. Deep Learning Research

2.4 Large Language Models or Transformer in Detecting Fraud Events

In 2024, there has been a growing interest in leveraging Large Language Models (LLMs) or transformer architecture for fraud detection in the financial industry. Bakumenko and colleagues focused on comparing five machine learning models, including Logistic Regression, Random Forest, Gradient Boosting Machines, Support Vector Machines, and Neural Networks, with BERT-based LLM embeddings to detect

anomalies in financial journal entries and statement data. Their study demonstrated that LLM embeddings significantly outperformed traditional machine learning methods in detecting anomalies. The paper highlights the effectiveness of LLMs in addressing the challenges of feature heterogeneity and sparsity in financial data. Future work suggested includes applying LLM embeddings to non-semantic, high-dimensional, and sparse transactional data, improving computational efficiency in detecting financial fraud patterns, integrating LLMs with other machine learning and deep learning models, exploring unsupervised strategies to address zero-day anomalies, enhancing data preprocessing strategies with LLM encoding efficiency and prompt engineering, and investigating further uses of LLM embeddings or alternative algorithms to improve anomaly detection capabilities (Bakumenko, Hlaváčková-Schindler , Plant , & Hubig, 2024).

Another research by Tian and Liu introduces a novel algorithm called Spatial-Temporal-Aware Graph Transformer (STA-GT), which addresses the limitations of traditional Graph Neural Networks (GNNs) in financial fraud detection. STA-GT incorporates a temporal encoding strategy within the GNN framework, enhancing its ability to capture temporal dependencies and spatial relationships in transaction data. Additionally, the algorithm overcomes the GNN structure's limitations by integrating a transformer layer, which facilitates interactions between target nodes and distant nodes across the graph. The paper demonstrates that STA-GT significantly outperforms general GNNs and other GNN-based models in financial fraud detection tasks. Future research could focus on improving the robustness of STA-GT against adversarial attacks and

handling noisy data, further enhancing its effectiveness in real-world applications (Tian & Liu, 2024).

Unlike using pre-train LLM, Yu and colleagues explored the use of an advanced Transformer model for credit card fraud detection, incorporating a self-attention mechanism and a feed-forward neural network within the encoder layer. The study demonstrated that the Transformer model significantly outperformed a variety of other models, including Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Decision Trees, Neural Networks, XGBoost, and the deep learning-based TabNet. Future research is encouraged to further investigate the potential of Transformer models in the fraud detection domain, exploring ways to enhance their robustness and applicability across different scenarios (Yu, et al., 2024).

2.5 Transformer Architecture Overview

The Transformer model is a groundbreaking deep learning architecture first introduced in 2017 by Ashish Vaswani and colleagues in the paper "Attention is All You Need." This model revolutionized natural language processing by replacing the traditional recurrent layers in encoder-decoder architectures with multi-headed self-attention mechanisms. This change allows the Transformer to capture dependencies across the entire sequence of data without relying on recurrence or convolution, which significantly improves parallelization and reduces training time. The Transformer model was demonstrated to outperform previous deep learning models, including those using ensembles, particularly in tasks such as English-to-German translation, where it achieved state-of-the-art results with lower computational costs (Vaswani, et al., 2017).

The paper also introduced the Transformer architecture, which revolutionized language translation by embedding input language into vectors that capture its meaning and patterns. These embeddings, combined with positional encoding, are processed through a series of encoder layers to produce a continuous vector representing all the information from the input. The decoder then uses this encoded information, along with previously generated outputs, to generate the next output in the sequence. The model effectively builds the sequence step by step by predicting the next output while incorporating previously generated symbols. Key components of the Transformer architecture include positional encoding, self-attention and multi-head attention mechanisms, feedforward neural networks, embedding layers, and the softmax layer. Together, these elements enhance the model's ability to understand and generate text with remarkable accuracy (Vaswani, et al., 2017)

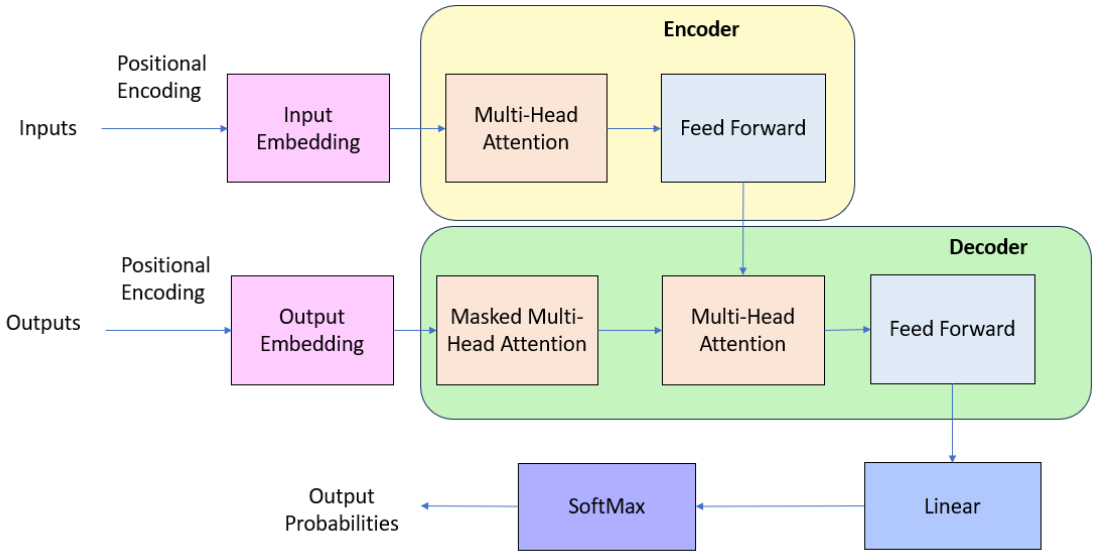

Figure 4. Transformer Architecture

2.5.1 Positional Encoding

After the input or previous output is embedded into vector-based data, Positional Encoding becomes crucial for maintaining the order of the sequence, as the arrangement of words in a text can greatly influence its meaning. To help the model comprehend the relative positions within a sequence, an additional positional encoding vector is incorporated into the original word embeddings. The Transformer model employs sine and cosine functions to create these positional encodings (PE), which are defined as follows:

$$PE(pos, 2i) = sin\ (pos/10000^{2i/d_{model}}) \quad (1)$$

$$PE(pos, 2i+1) = \cos\ (pos/10000^{2i/d_{model}}) \quad (2)$$

Here, “pos” represents the position in the sequence, and “i” denotes the dimension. The sine function is applied to even dimensions, while the cosine function is used for odd dimensions. This approach allows the model to effectively capture the sequential information of the text (Vaswani, et al., 2017).

2.5.2 Attention

The attention function maps a query and a set of key-value pairs to an output vector, enabling the model to focus on specific parts of the input sequence. In the context of the attention layer, self-attention is a mechanism that identifies the relationships and dependencies between a word and all other words within the same sequence. This allows the model to assess the relevance of each word to every other word, effectively capturing the contextual relationships within the sequence. The self-attention mechanism operates using three vectors derived from the input embeddings: Query (Q), Key (K), and Value

(V). Specially, the Query represents the current focus or question, the Key serves as the reference points for all other words in relation to the focus word, and the Value contains the actual information associated with each word (Shastri, 2024).

After self-attention, the Query (Q), Key (K), and Value (V) matrices are processed through the Scaled Dot-Product Attention mechanism. This process begins by computing the dot product between the Query and Key matrices, which measures the similarity between the query and each key. The resulting scores are then scaled by dividing them by the square root of the dimensionality of the Key matrix to ensure numerical stability. A softmax function is subsequently applied to these scaled scores, converting them into a probability distribution that reflects the importance of each word in the sequence relative to the query. Finally, this probability distribution is used to weight the Value matrix, effectively focusing on the most relevant word information (Vaswani, et al., 2017).

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

In real-world applications, a single attention function can be performed in parallel across different parts of the input, leading to the concept of multi-head attention. Multi-head attention allows the model to focus on different aspects of the input sequence simultaneously, capturing a richer set of relationships. Each attention function in this setup is referred to as a "head", and multi-head attention is achieved by concatenating the outputs from multiple attention heads. This enables the model to gather information from various positions in the sequence, enhancing its ability to understand complex patterns and dependencies. The multi-head attention mechanism is mathematically defined as:

$$MultiHead(Q, K, V) = concat(head_1, \dots, head_h)W^O \quad (4)$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Here, "h" denotes the number of parallel attention layers, and $d_k = d_v = \frac{d_{model}}{h}$. The projection matrices are

$$W_i^Q \in R^{d_{model} \times d_k}, W_i^K \in R^{d_{model} \times d_k}, W_i^V \in R^{d_{model} \times d_v}\ and\ W_i^O \in R^{hd_v \times d_{model}}$$

During the decoding process, a masking technique is applied to the scaled dot-product attention by replacing all input value in softmax function with negative infinity. This mask ensures that the model cannot access or consider future information that would otherwise influence the current output. By doing so, the model ensures that the prediction for position i is based solely on the known outputs before position i (Vaswani, et al., 2017).

2.5.3 Feed Forward Networks

After the self-attention layer, which captures relationships between different positions in the sequence, a position-wise feed-forward network is applied independently to each position to further transform the representations. This feed-forward network introduces non-linear transformations, refining the features extracted by the self-attention mechanism. The parallel processing capability of the feed-forward network enables the Transformer model to train faster while efficiently learning complex relationships within the data (BL, 2024). The feed-forward network includes two linear transformations defined as follows (Vaswani, et al., 2017) :

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \quad (5)$$

2.6 Large Language Models

A large language model employs deep learning techniques and large data from wide range of topics to summarize, comprehend, generate and predict new content. LLMs are pretrained on special type of data to understand the complexity language within certain area. Utilizing techniques like fine-tuning and zero-/one-/few-slot learning, the pretrained models could be tailored for special tasks (Marvin, Hellen, Jjingo, & Nakatumba-Nabende, 2024).

Tech companies like OpenAI utilize the Transformer architecture to train extensive datasets, resulting in the development of powerful Large Language Models (LLMs) such as the Generative Pre-Trained Transformer (GPT) series. In addition to GPT, other notable LLMs include BERT, LLaMA, and Gemini.

2.6.1 Popular Large Language Models

The first GPT model was introduced by OpenAI in 2018. This initial version, known as GPT-1, was pre-trained on the BooksCorpus dataset, which contains over 7,000 unique unpublished books spanning various genres, including Romance and Fantasy. GPT-1 leveraged a two-stage training process: unsupervised pre-training followed by supervised fine-tuning on specific tasks. This approach allowed GPT-1 to capture a broad understanding of language, leading it to outperform traditional models such as LSTMs and their ensemble variants across various natural language understanding tasks (Radford, Narasimhan, Salimans, & Sutskever, 2018). In 2023, OpenAI released the "GPT-4 Technical Report," highlighting GPT-4's remarkable performance, which reaches human-level proficiency across 57 subjects. This achievement was made possible through

the development of advanced deep learning infrastructure and optimization methods. However, the report also acknowledges certain limitations of GPT-4, such as occasional hallucinations, reasoning errors, and the lack of real-time learning. OpenAI underscores the importance of ongoing efforts to improve safety and alignment, making these areas a key focus for future development. (OpenAI, GPT-4 Technical Report, 2023).

In 2019, Google AI Language introduced the Bidirectional Encoder Representations from Transformers (BERT), a groundbreaking pre-trained model designed to enhance natural language understanding. BERT employs a masked language model (MLM) to enable deep bidirectional training, allowing the model to consider both left and right context simultaneously across all layers. This approach significantly improves performance on various tasks, including question answering and language inference, as demonstrated on datasets like GLUE, SQuAD, and SWAG. Additionally, BERT incorporates the Next Sentence Prediction (NSP) task, which further strengthens its ability to comprehend and model the relationships between sentences, thereby advancing the state of the art in multiple NLP benchmarks (Devlin, Chang, Lee, & Toutanova, 2019).

In 2023, Meta AI introduced the LLaMA series of large language models, ranging from 7 billion to 65 billion parameters, trained exclusively on publicly available data sources such as CommonCrawl, C4, GitHub, and Wikipedia. The LLaMA models continue to leverage the transformer architecture but are designed with several key optimizations, including pre-normalization, SwiGLU activation, and rotary embeddings, which enhance training stability and overall performance. Notably, LLaMA-13B outperforms GPT-3, despite being more than ten times smaller, demonstrating the

effectiveness and efficiency of these models on most benchmarks (Touvron, et al., 2023). In July 2024, LLaMA advanced its model to version 3.1, significantly enhancing its capabilities with an expanded context length, allowing it to handle much longer sequences of text. This update also introduced new security and safety tools, providing developers with greater flexibility for fine-tuning and deploying the model across various environments (Meta, 2024).

In 2023, Google introduced the Gemini family of large language models, which are highly capable multimodal models trained across a diverse range of data types, including images, audio, video, and text. The models underwent extensive large-scale pre-training and subsequent post-training to enhance performance across various domains. Gemini models come in three distinct sizes: Ultra, Pro, and Nano, each optimized for different computational and application needs. These models excel across a wide range of benchmarks, including tasks involving reasoning, language, coding, and multimodal understanding, setting new state-of-the-art performance records (Google Gemini Team , 2023).

In 2024, Google introduced the Gemma models, a family of lightweight, state-of-the-art language models built upon the advancements of Gemini research. These models are available in two size, 2 billion and 7 billion parameters, designed to cater to different computational needs and deployment scenarios. The Gemma models demonstrate superior performance, surpassing similarly sized open models in 11 out of 18 text-based tasks, excelling in areas such as question answering, reasoning, mathematics, and coding. Notably, these models strike an optimal balance between efficiency and size while

undergoing rigorous safety evaluations to ensure responsible deployment (Google Gemma Team, 2024).

2.6.2 Prompt Engineering

One of the key applications of large language models is generating solutions based on a given prompt. A prompt is essentially an instruction or context provided to the model to produce output for accomplishing specific tasks. Prompt engineering refers to the optimized practice of designing and refining prompts to maximize the efficiency and accuracy of language models in producing relevant responses across diverse tasks. To create an effective prompt, the first step is to clearly define the goal and determine the specific questions or instructions that will guide the model to generate the desired response. Understanding the model's capabilities and limitations is crucial, as it helps in selecting the most appropriate prompt format. Next, providing sufficient context enables the model to better understand the task at hand. Finally, it's essential to test and continuously refine the prompt to ensure it generates the desired output, aligning with the set goal. After establishing the prompt creation process, several techniques can further enhance and optimize prompt engineering. One such technique is automatic instruction generation, which automates the creation of guides or instructions for generating prompts. This technology allows for the evaluation of a model's zero-shot performance, often outperforming instructions crafted by human annotators. Another method, program synthesis for prompt engineering, focuses on generating programs from high-level specifications. This approach can be used to create prompts that meet specific criteria, improving both precision and effectiveness (Marvin, Hellen, Jjingo, & Nakatumba-Nabende, 2024).

2.6.3 Hyperparameter Tuning

Fine-tuning hyperparameters is critical to enhancing the performance of Large Language Models (LLMs), as it allows for better alignment of the model's output with specific requirements and expectations. Adjusting key hyperparameters optimizes the model's accuracy, contextual relevance, and overall efficiency (Talamadupula, 2024).

Model Size: Lager size models are more capable to handle complex tasks as having more layers and weights can be learned from training data and understand the relationship between tokens. The down side is larger LLM has higher cost, slower speed and potential overfitting (Talamadupula, 2024).

Number of Epochs: This refers to the number of complete iterations the model makes over the entire training dataset. A higher number of epochs allows the model to better grasp semantic relationships, but too many epochs can lead to overfitting, where the model performs well on training data but poorly on unseen data (Talamadupula, 2024).

Learning Rate: This controls how quickly the model is updated from the loss function when it predicts an incorrect output during training. A higher learning rate could result in overfitting, while a lower learning rate increases stability and long training time (Talamadupula, 2024).

Batch Size: This indicates the size of data processes each epoch. Each batch is a portion of the data. Smaller batch size requires less memory to process in LLM (Talamadupula, 2024).

Max Output Tokens: This determines the maximum number of tokens (sequence length) the model can generate. Increasing the number of output tokens may help the model produce more detailed and comprehensive responses but requires more memory and computational power (Talamadupula, 2024).

Top-k and Top-p Sampling: Top-k sampling samples tokens with the highest probabilities until the selected number of tokens (integer from 1 to 100) is reached. Top-p sampling samples tokens with the highest probability scores until the sum of the scores reaches to the specified value (floating number from 0.0 to 1.0). Higher values lead to more diverse outputs, while lower values produce more conservative results (IBM, 2024).

Temperature: This hyperparameter affects the creativity and variability of the model's output by altering the token probability distribution. Lower temperatures (closer to 0) make the model more deterministic, generating predictable and reliable responses. Higher temperatures (closer to 2) lead to more creative and diverse outputs, as the model is encouraged to explore unusual tokens (Talamadupula, 2024).

Repetition Penalties: This hyperparameter discourages the model from repeating tokens too frequently, thus promoting diversity in the output. By penalizing repeated tokens, the model is encouraged to produce a wider range of responses, helping avoid redundancy and monotony (Talamadupula, 2024).

2.6.4 Retrieval Augmented Generation

Retrieval Augment Generation (RAG) is a technique designed to enhance the knowledge of large language models (LLMs) by incorporating additional external data. Since LLMs are limited to the public data available up to the time they were trained,

RAG allows models to retrieve and integrate new or private information, such as data that became available after the model's training period (LangChain, 2024).

One practical application of RAG is in building chatbots, where it enables the model to better understand and respond to conversations by leveraging updated or domain-specific information. The process of applying RAG to improve LLM responses involves several key steps. First, data is collected, including conversation flows, and the custom dataset is preprocessed. This dataset is then used to fine-tune the LLM. Additionally, the same dataset is converted into vector embeddings and stored in a vector database, enabling the LLM to retrieve relevant information during the generation process. With this approach, the fine-tuned LLM becomes more adept at answering client questions with greater precision. (Vidivelli, Ramachandran, & Dharunbalaji, 2024)

Chapter 3—Methodology

3.1 Introduction

This chapter provides a detailed overview of the methodology employed in this research to conduct the experiment based on the hypotheses outlined in the previous chapter. The financial fraud dataset used for this study is sourced from an open-access repository on GitHub and contains binary-labeled credit card transactions (fraudulent or non-fraudulent).

The dataset used is from the Capital One Data Science Challenge, a public dataset released by Capital One in 2018 as part of a recruitment initiative (CapitalOneRecruiting, 2018). This synthetic dataset was made available for learning and analysis purposes. Since its release, three notable studies have utilized this dataset:

1. **Gradient Boosting vs. Logistic Regression**: The first study focused on building a gradient boosting model and comparing its performance with a logistic regression model (calvinscottforbes, 2021).
2. **Fraud Detection Case Study**: The second study aimed to develop a robust predictive model for fraud detection by evaluating the performance of random forest, gradient boosting, and XGBoost models (abdullahsaka, 2021).
3. **Data Analytics and Model Comparison**: The most recent study included a comprehensive data analysis, a model comparison between logistic regression and gradient boosting classifiers, and a discussion of oversampling techniques to improve model performance (Lee, Hu, Tecimer, & Luo, 2023).

As discussed in Chapter 2, financial fraud datasets are typically highly imbalanced, which can significantly affect model performance. To address this, a Large Language Model (LLM) is applied to generate synthetic fraud transactions by leveraging historical patterns and zero/few-shot learning. This method introduces both high-frequency and novel patterns to oversample the fraudulent transactions, thereby mitigating the imbalance issue.

In this research, 16 Python Jupyter Notebook programs were developed to support various tasks, including data cleaning, synthetic data generation and oversampling, machine learning and deep learning model comparisons, LLM-based analytics on pattern recognition, hybrid LLM-based embedding methodology implementation, performance analytics, and visualization. The research is structured into four distinct phases: (1) traditional data sampling and fraud detection modeling, (2) LLM-based categorical embedding hybrid modeling, (3) Generating synthetic data through fine-tuned OpenAI

modeling, and (4) model optimization. Figure 5 provides a detailed illustration of how these Python programs integrate into the overall data flow for this Praxis.

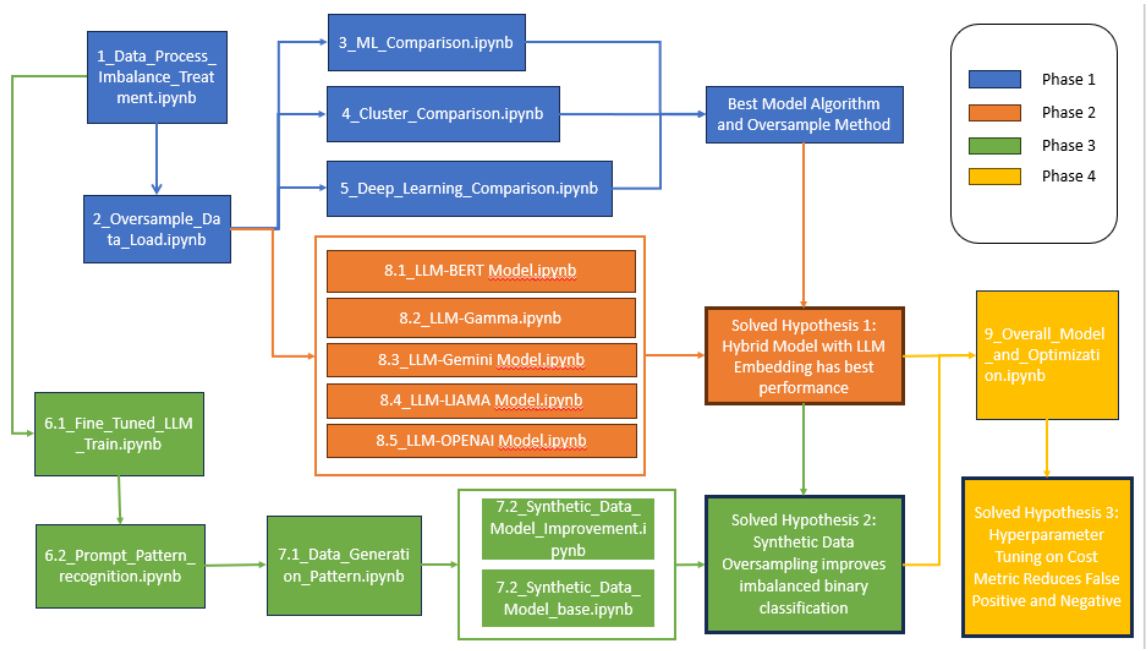

Figure 5. Python Scripts Process Chart

3.2 Data Description

This research utilizes the financial fraud transaction dataset from the Capital One Data Science Challenge, which was made publicly available in a GitHub repository[1] in 2018. The dataset is open for public analysis and exploration, and serves as a foundation for implementing and evaluating different fraud detection methodologies for this study.

The dataset comprises a wide range of transaction-related attributes, including transaction amount, merchant details, cardholder behavioral information, and a binary classification label indicating whether a transaction is fraudulent or legitimate. These

[1] Capital One Data Science Challenge. https://github.com/CapitalOneRecruiting/DS

features offer significant potential for building, testing, and refining machine learning models aimed at accurately identifying fraudulent activities.

The original transaction data is structured as a JSON object containing key-value pairs for each transaction record. To convert this unstructured text data into a structured tabular format, the .txt file is first transformed into a .json file. Next, the json library is used to parse and normalize the data, making it suitable for further analysis. The Pandas library is employed for data manipulation, while seaborn and matplotlib are used for visualizing data patterns and extracting meaningful insights. For a comprehensive evaluation of the dataset, several analytical techniques are applied, including feature distribution analysis by class, correlation matrix heatmaps, time series analysis, and numeric and categorical analysis. Additionally, feature importance analysis is conducted to identify key predictors of fraudulent behavior. The resulting visualizations are presented and discussed in the subsequent sections.

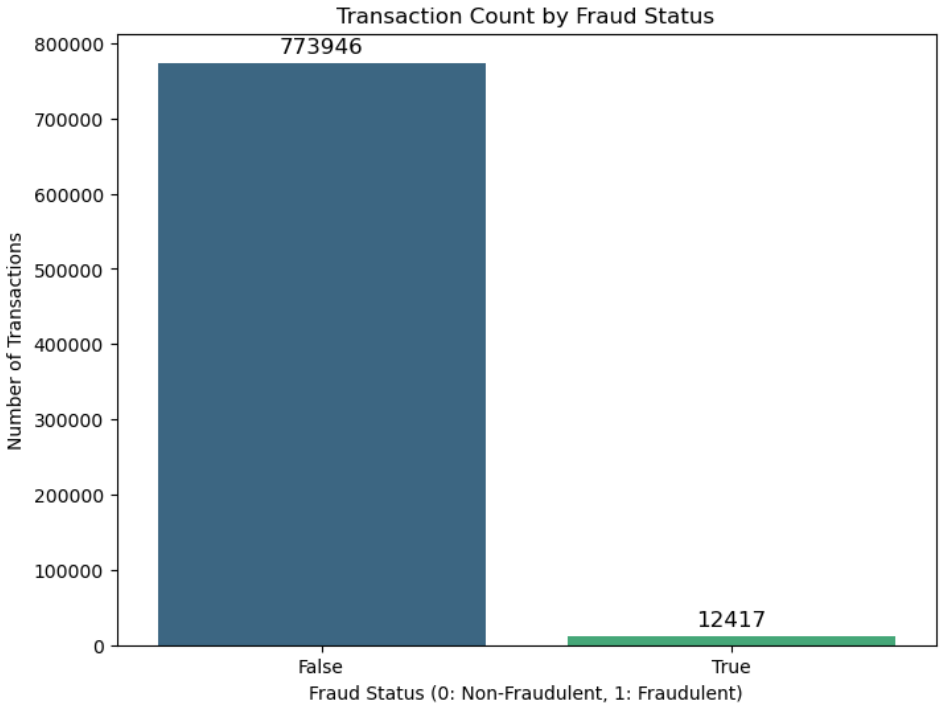

Figure 6. Distribution of transaction count by Fraud Status

The dataset consists of 783,363 rows and 29 variables. The target variable, "isFraud", is a highly imbalanced Boolean indicator, with only 12,417 transactions flagged as fraudulent and 773,946 flagged as legitimate, as show in Figure 6. Distribution of transaction count by Fraud Status. The dataset includes 4 numeric variables, 4 timestamp variables, and 21 categorical variables. To better utilize the timestamp features, 3 new variables are created to capture the time differences between the transaction date and the account open date, the last address change date, and the current card expiration date. Among the categorical variables, 6 are completely missing, and the "customerId" variable is identical to "accountNumber", making these 7 variables redundant and suitable for removal. Additionally, the "CVVMatch" variable was created to consolidate the information from "cardCVV" and "enteredCVV" as a security

measure. The remaining categorical features can be transformed into numeric format using label encoding to prepare them for further modeling.

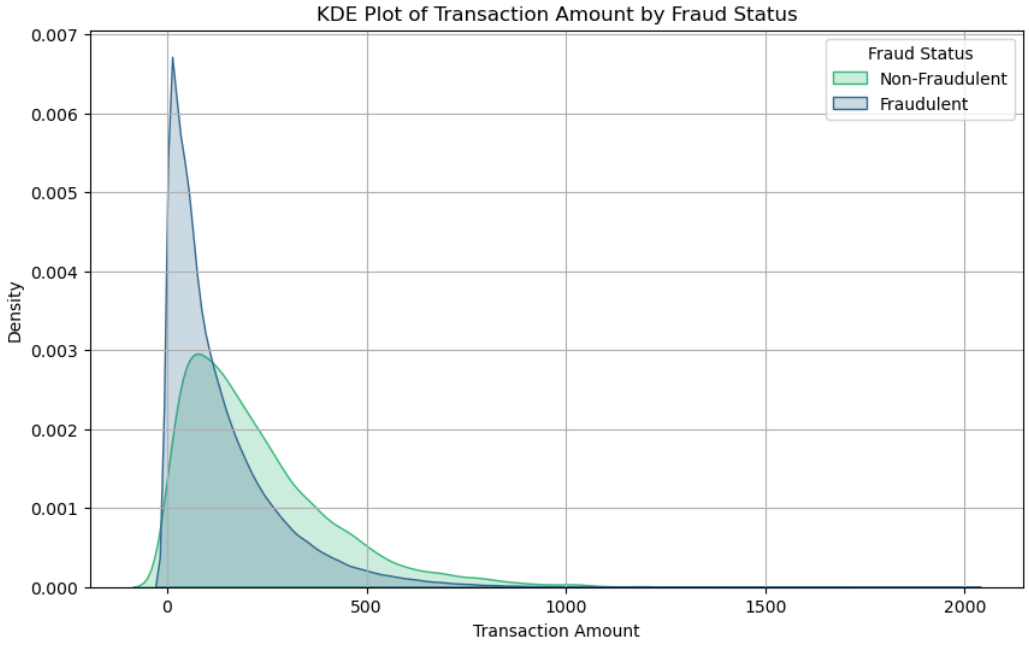

Figure 7. KDE Plot of Transaction Amount by Fraud Status

Figure 7 displays the Kernel Density Estimate (KDE) plot for transaction amounts, segmented by fraud status. The plot indicates that fraudulent transactions are primarily concentrated at lower transaction values compared to legitimate ones. The dataset spans from January 1, 2016, to December 30, 2016, with a relatively stable distribution of fraud incidents per day throughout the year, as shown in Figure 8. To simulate a real-world fraud detection scenario, transactions occurring before November 1, 2016, can be used as the training set, while those from November and December can serve as the test set for model evaluation.

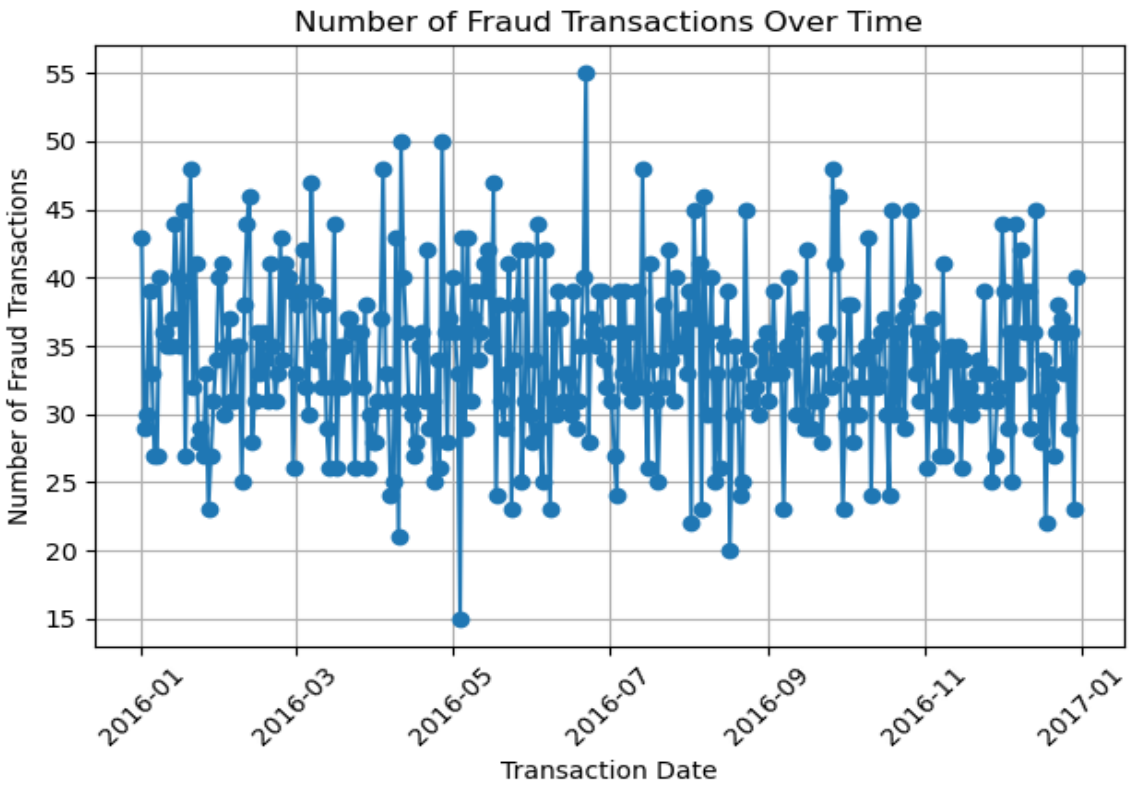

Figure 8. Number of Fraud Transactions Over Time

Due to the introduction of newly created variables designed to capture time gaps during transaction processing, some of these variables exhibit high correlations with the original timestamp variables. As shown in the correlation heatmap in Figure 9, two original timestamp variables display a very strong negative correlation (-0.99) with their corresponding time gap features, which measure the time difference between the transaction date and both the account opening date and the date of the last address change. Given this strong correlation, the original timestamp variables for the account opening date and the last address change can be safely removed without any significant loss of information.

Similarly, the variable “MonthtoExp” shows a perfect correlation (1.0) with “currentExpDate_tfm,” and “DAYsinceopen” has a strong negative correlation (-0.99)

with “accountOpenDate_tfm.” Due to these high correlations, the variables “currentExpDate_tfm” and “accountOpenDate_tfm” can also be eliminated to reduce redundancy.

Additionally, the variables “creditLimit” and “availableMoney” are moderately correlated (0.83), reflecting their mathematical relationship: the difference between the credit limit and available money represents the amount of credit used but not yet repaid. Since available money more accurately captures an individual's current purchasing power, it is more practical to retain this variable for a clearer representation of the user’s financial status.

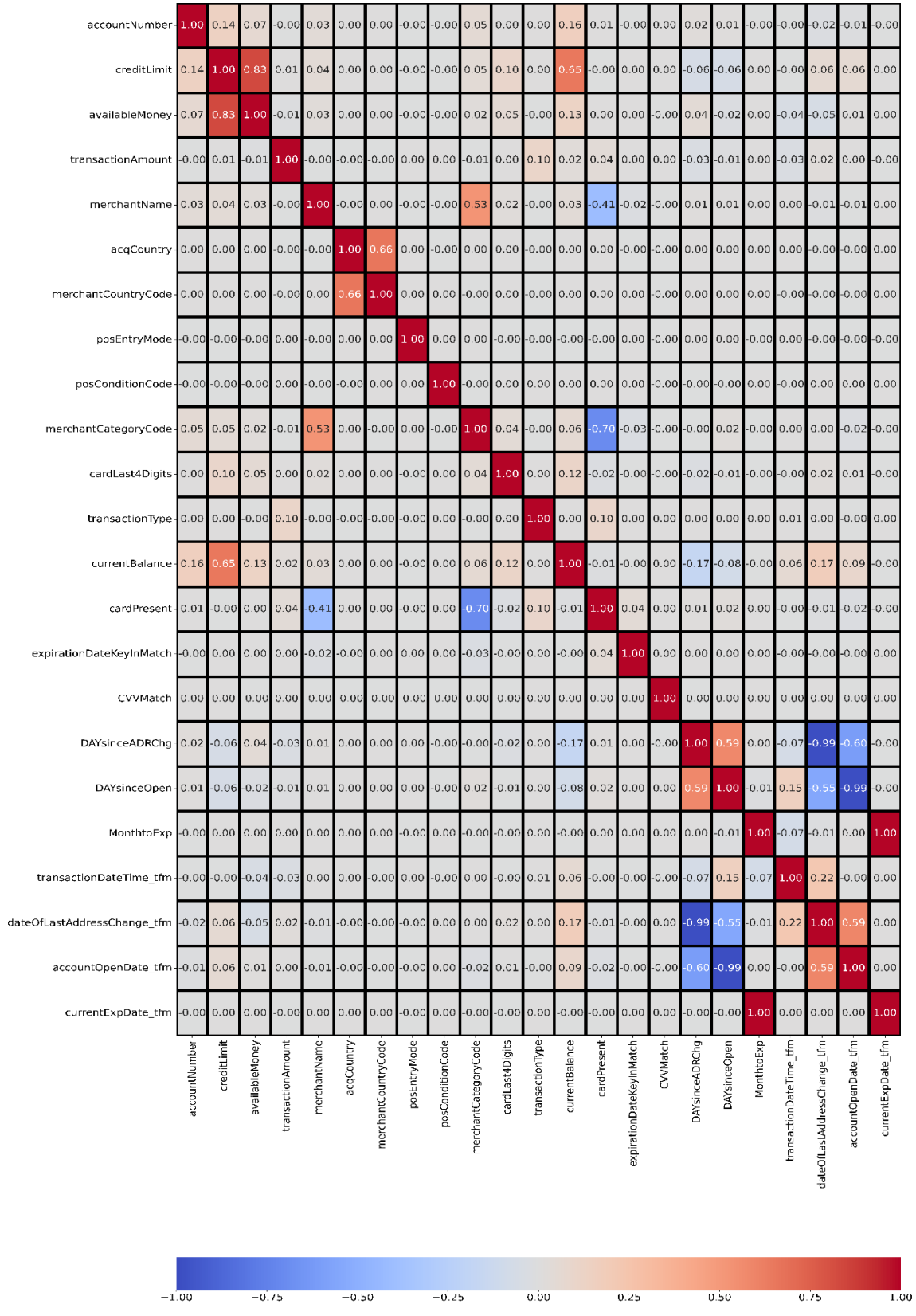

Figure 9. Correlation Heap Map

According to Occam’s Razor, the principle that simpler models are preferable, the general model could be selected by the features that contribute to 80% of the model’s predictive power (PerceptiveAnalytics, 2018). Figure 10 below shows the cumulative feature importance plot for the LightGBM model, which accumulates feature importance scores until a defined threshold (such as 80%) is met. This method helps identify features with low predictive power, allowing them to be safely removed without compromising the model's overall performance. As a result of this analysis, eight key features were selected, while eleven features with minimal contribution were removed to enhance model efficiency.

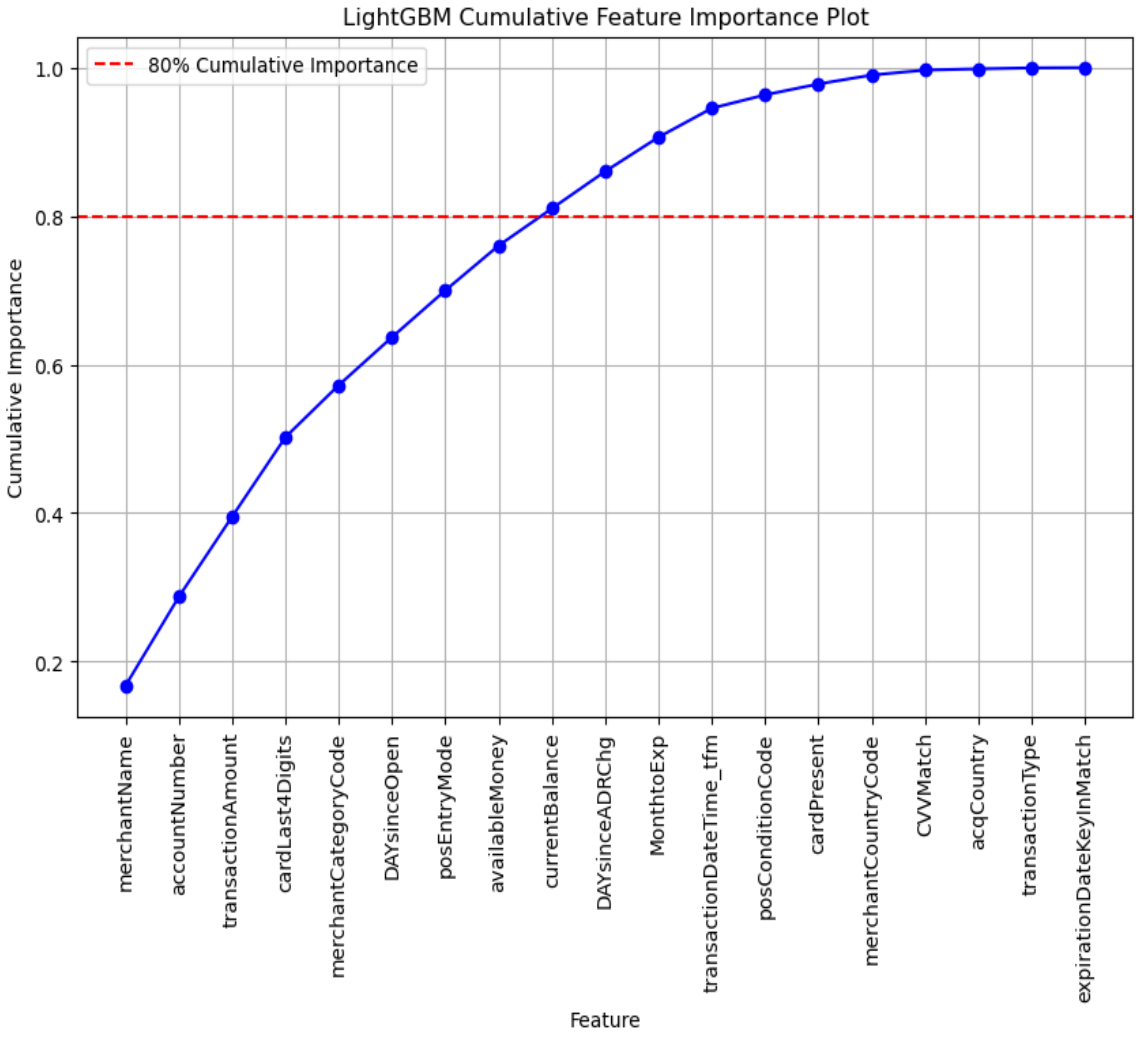

Figure 10. LightGBM Cumulative Feature Importance Plot

3.3 Research Design

This research aims to leverage a variety of pretrained large language models, including GPT, BERT, LLaMA, Gemini, and Gemma, to tackle financial fraud detection using multiple innovative approaches. These models are employed for tasks such as fine-tuned LLM, fraud pattern recognition through few-shot prompting, synthetic data generation to address class imbalance, and hybrid classification using LLMs, with the overarching goal of enhancing the accuracy and robustness of fraud detection systems compared to traditional methods.

The research methodology is organized into three stages, each designed to test a specific hypothesis for improving classification performance. In the first stage, a GPT-4o Mini model is fine-tuned with historical training data to generate synthetic samples that replicate known fraud patterns. The fine-tuned model is also used for pattern recognition to identify potential fraud trends for the forecasted November and December seasonal periods. These patterns are then leveraged to guide few-shot prompting, generating synthetic data that captures novel fraud behaviors not found in the historical dataset. The synthetic data is merged with the original data to create a comprehensive, balanced dataset, effectively addressing class imbalance. This balanced dataset is then fed into a traditional binary classifier and compared against a classifier trained using conventional oversampling methods to evaluate whether the synthetic data improves overall model performance.

In the second stage, various pretrained models, including GPT, BERT, LLaMA, Gemini, and Gemma, are used to generate embeddings for categorical variables. These embeddings provide a richer, more representation of text compared to traditional label

encoding, enabling the model to better capture the underlying semantics of categorical data. To optimize computational efficiency, the embeddings are reduced to 10 principal components using Principal Component Analysis (PCA). These compressed embedding columns are then combined with the original numerical variables, and the best-performing model from the first stage is retrained. The updated model is evaluated to determine whether this method enhances overall performance.

The final stage centers on developing an optimization methodology that balances profit, cost, and the rates of false positives and false negatives to minimize the overall cost of fraud risk. Large Language Models (LLMs) are employed to analyze and summarize fraud patterns linked to both false positives and false negatives. Leveraging these insights, a fine-tuned LLM generates additional synthetic data to help balance false positive and false negative rates. Two parameters, α and β, are introduced to quantify the impact of reducing these rates, enhancing both the cost-efficiency and effectiveness of the proposed fraud detection strategy.

The objective of this research is to leverage Large Language Models (LLMs) to improve the performance of binary classifiers in financial fraud detection by identifying optimal combinations of imbalance treatments, traditional or LLM-enhanced classifiers, and cost optimization strategies to effectively mitigate fraud risk. The study seeks to develop novel methodologies that elevate classification performance in cybersecurity contexts within the financial sector. The research approach introduces LLMs into a traditionally tabular data, enabling more efficient pattern recognition and decision-making. This framework also lays the groundwork for future exploration into how AI

tools can be used to generate more precise and interpretable model explanations, bridging the gap between advanced LLM capabilities and practical financial applications.

The overall research design aims not only to validate the proposed hypotheses but also to provide actionable recommendations for integrating large language models with traditional classifiers to address similar classification challenges in the cybersecurity domain. Furthermore, the findings and methodologies from this research can be applied beyond cybersecurity, offering insights for developing models in various fields such as instant authorization approval, marketing advertisement response prediction, credit risk scoring, and other classification-based applications.

3.4 Traditional Methods to Detect Fraud Events

This study employs a range of traditional techniques for detecting fraud events, including data balancing methods and the application of binary classification models. These methods are implemented using four Python programs, iterating through various models across multiple data balancing methodologies. In this section, six data oversampling techniques will be systematically applied to the top ten performing classification models identified from PyCaret classification model comparison, two unsupervised models' comparison, and two deep learning models. The following sections provide an overview of the data balancing techniques and classifier methods used.

3.4.1 Traditional Oversampling Techniques

In real-world datasets, such as those related to credit card fraud, the minority class—representing rare events—can often be misclassified as noise or outliers, leading to incorrect identification of fraud instances. Oversampling is one approach used to address this issue, aimed at increasing the number of minority class samples to balance

the overall class distribution and improve model performance (Gosain & Sardana, 2017). In this research, six oversampling techniques will be applied during pre-processing and iterated with various modeling algorithms to enhance overall performance.

- **Random oversampling (ROS)**

Random oversampling (ROS) duplicates minority records randomly until the minority class size is equal to majority class. For reproducibility, usually the assigned seeds could retrieve the randomness (El-Naby, Hemdan, & El-Sayed, 2022).

- **Synthetic Minority Oversampling Technique (SMOTE)**

SMOTE (Synthetic Minority Over-sampling Technique) uses the k-nearest neighbors (KNN) algorithm to generate synthetic data for the minority class, addressing class imbalance by increasing the size of the minority class without simply duplicating existing records, as in random oversampling. Synthetic data is generated by selecting the K nearest neighbors of a given data point and creating new samples based on the original feature vector and its neighbors. A random number, t, is drawn from a uniform distribution between 0 and 1 (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), and the new synthetic feature is calculated as follows,

$$X_i' = X_i + (X_{KNN_i} - X_i) * t \qquad (6)$$

For example, if the dataset contains two variables, such as height and weight, and a data point A is (165,50) with its nearest neighbor B at (170,65), SMOTE generates a new synthetic data point using the formula:

$$\begin{bmatrix} X_1' \\ X_2' \end{bmatrix} = \begin{bmatrix} 165 \\ 50 \end{bmatrix} + \begin{bmatrix} t_1 * (170 - 165) \\ t_2 * (65 - 50) \end{bmatrix} = \begin{bmatrix} 165 + 5t_1 \\ 50 + 15t_2 \end{bmatrix} \quad (7)$$

Here t_1 and t_2 are random values from a uniform distribution between 0 and 1. This ensures that the synthetic data point lies between the original data point (165, 50) and its neighbor (170, 65), thus reducing the risk of creating outliers.

- **Borderline Synthetic Minority Oversampling Technique (Borderline SMOTE)**

Borderline SMOTE outperforms the standard SMOTE algorithm when oversampling is focused on the borderline samples of the minority class. These borderline samples are often more difficult to classify correctly compared to those farther from the decision boundary. Borderline SMOTE specifically targets these "dangerous zones," where misclassification is more likely, applying the SMOTE algorithm again to generate synthetic samples in the borderline area. This targeted approach helps improve the classification accuracy (El-Naby, Hemdan, & El-Sayed, 2022).

- **Adaptive Synthetic (ADASYN)**

The Adaptive Synthetic (ADASYN) algorithm extends the SMOTE approach by dynamically adjusting the amount of synthetic data generated for different minority class examples. It assigns weights to each minority class instance based on its level of difficulty to classify—harder-to-learn instances are given higher weights. ADASYN then generates more synthetic samples for these difficult cases using the SMOTE algorithm. This targeted generation of synthetic data shifts the classifier's decision boundary towards the harder-to-classify samples (El-Naby, Hemdan, & El-Sayed, 2022).

- **SMOTE with Tomek Link (SMOTETomek)**

SMOTE with Tomek Links is an enhancement of the SMOTE algorithm that helps prevent overfitting caused by oversampling the minority class. While SMOTE generates synthetic samples to balance the dataset, Tomek Links clean the space by removing borderline cases between the majority and minority classes. Specifically, Tomek Links identify and remove pairs of nearest neighbors from different classes, reducing the risk of introducing noise and misclassification. This combination ensures better class separability and improves the overall quality of the synthetic data (El-Naby, Hemdan, & El-Sayed, 2022).

- **SMOTE with Edited Nearest Neighbors (SMOTEEN)**

SMOTE with Edited Nearest Neighbors (ENN) is similar to SMOTE with Tomek Links in that both aim to remove noisy data. However, ENN goes further by targeting borderline samples and smoothing the decision boundary. It removes any sample whose class label differs from at least two of its three nearest neighbors, leading to a more aggressive data cleaning process. As a result, ENN typically removes more data points than Tomek Links (El-Naby, Hemdan, & El-Sayed, 2022).

3.4.2 Binary Classification

In this study, the PyCaret classification library was employed to streamline the implementation of various supervised machine learning models in Python, significantly accelerating the experimentation process. PyCaret simplifies the coding effort required to apply a wide range of classifiers, including Extra Trees, Random Forest, Decision Tree, Extreme Gradient Boosting, LightGBM, AdaBoost, K-Nearest Neighbors, Ridge Classifier, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Logistic

Regression, and more. The top 10 performing models were selected for comparison, with different data oversampling techniques applied to enhance performance. These models were majorly grouped into three categories listed below.

- **Tree-Based Classifiers**

Tree-based models start with a root node and branches, where each branch represents an intermediate decision and each leaf node provides a final prediction based on feature values. Unlike algorithms like KNN or SVM, tree-based models do not require feature standardization and can handle both categorical and numerical variables, making them well-suited for capturing non-linear relationships. Importantly, tree-based models are highly flexible and can be combined with ensemble methods, leading to more advanced models like Random Forest and boosting techniques (Rendyk, 2024).

A decision tree, the simplest form of a tree-based model for classification, is easy to interpret but prone to overfitting. To address this, **Random Forest** introduces the concept of bagging, where multiple decision trees are trained on different subsamples of the data, and the final prediction is an average of these trees, improving robustness and reducing overfitting (Rendyk, 2024).

Boosting models work sequentially, where each new model is trained to correct the mistakes of the previous one. **AdaBoost** is a type of boosting that adapts by focusing more on misclassified instances, adjusting the weights of these instances in the next iteration to improve the model's performance. Similarly, **Gradient Boosting Machines (GBM)** correct errors from the previous learners, but instead of simply adjusting weights, it fits the new learner to the negative gradient of the loss function, enabling the use of various loss functions for greater flexibility. **Extreme Gradient Boost (XGBoost),** an

advanced form of gradient boosting, builds on GBM by incorporating regularization to control tree complexity and prevent overfitting. XGBoost also handles missing data effectively and allows for fine-tuning hyperparameters like maximum tree depth and learning rate (shrinkage) to enhance model performance and stability, ensuring robust learning (Rendyk, 2024).

- **Linear Classifiers**

Linear models assume that the target value is a linear combination of all input features. In **Linear Regression**, the model finds the optimal coefficients for each feature by minimizing the residual sum of squares between the observed and predicted values. **Ridge Regression** builds on linear regression by adding regularization, penalizing large coefficient values to reduce overfitting and improve generalization (Scikit Learn, 1.1 Linear Models, 2024). **Logistic Regression** applies a logit transformation to the response variable, establishing a linear relationship between the log-odds of the target and the independent variables (Scikit Learn, 1.1.11 Logistic Regression, 2024).

- **Other Classifiers**

Both linear and quadratic **discriminant analysis** model the class-conditional distribution of the features. Classification predictions are made using Bayes' rule, which assigns the most probable class based on these distributions (Scikit Learn, 1.2. Linear and Quadratic Discriminant Analysis, 2024). **K-nearest neighbors (KNN)** classification, on the other hand, identifies the K closest points to a new data point based on distance metrics. The class prediction is determined by majority voting among the nearest neighbors, aligning the predicted class with the most frequent class among them (Sachinsoni, 2023).

3.4.3 Anomaly Detection

In addition to supervised learning for classification, 2 unsupervised learning algorithms will be tested for anomaly detection: Copula-Based Outlier Detection (COPOD) and Isolation Forest. This study assumes the fraud cases are abnormal acting as outliers comparing to normal transactions. By identify the abnormal patterns, fraud transaction could be identified in this way.

- **Copula-Based Outlier Detection (COPOD)**

The COPOD algorithm relies on the empirical cumulative distribution function (CDF) to detect anomalies by assessing tail probabilities. It does not involve traditional learning processes, hyperparameter tuning, or stochastic training, which eliminates the risk of overfitting associated with hyperparameter selection. COPOD quantifies the contribution of each dimension to the overall abnormality, and is highly efficient, capable of handling datasets with over 1 million observations and 10,000 variables on a standard personal laptop (Li, Zhao, Botta, Ionescu, & Hu, 2020).

- **Isolation Forest**

The Isolation Forest algorithm, introduced by Fei Tony Liu and Zhi-Hua Zhou in 2008, is designed to efficiently isolate anomalies by randomly selecting features and partitioning the data, similar to the structure of random forests. In this approach, normal data points typically require more partitions to be isolated, while anomalous points tend to be isolated with fewer cuts, as they are located in sparse regions and thus require shorter branches to separate them from other observations (Akshara, 2024).

3.4.4 Deep Learning

Traditional machine learning requires heavy design on features to improve performance. Deep learning can automatically extract predictive features which gain significant prediction power. The two main deep neural networks (DNN) architectures are convolutional neural network (CNN) and recurrent neural network (RNN) (Cui, Long, Min, Liu, & Li, 2018).

- **Convolutional Neural Network (CNN)**

CNNs are highly effective for learning spatial features, making them particularly well-suited for image classification tasks. The convolution and down-sampling layers capture deep, abstract features that can represent complex patterns or abnormal behaviors. A basic CNN model consists of three main layers (Cui, Long, Min, Liu, & Li, 2018). The convolution layer follows, applying a set of filters to extract localized features from the input, transforming subregions into a smaller matrix of convolved features that capture essential patterns. Finally, the max pooling layer reduces dimensionality by selecting the maximum value from each region in the feature map, which reduces computational requirements for processing the data (Manav, 2024).

- **Recurrent Neural Network (RNN)**

RNNs are widely used for predicting time series or sequential data, as they retain information from the current sequence to compute the current output. RNNs incorporate a hidden layer that connects the current input with information from previous time steps. However, historical information in RNNs tends to decay over time, making it difficult to capture long-term dependencies. To address this limitation, Long Short-Term Memory (LSTM) networks replace the standard hidden layer with more sophisticated memory

blocks, allowing them to coordinate historical information more effectively, thereby enhancing their capacity for learning complex time-dependent patterns (Cui, Long, Min, Liu, & Li, 2018).

3.5 LLM Base Fraud Event Generation for Oversampling

In addition to traditional oversampling methods, this research introduces an innovative three-stage methodology to generate synthetic fraud transactions:

1. **Fine-tuning the GPT-4o Mini Model**: The model is fine-tuned with historical fraud data to further train its ability to generate realistic synthetic transactions.

2. **Synthetic Data Generation**: Using the fine-tuned LLM, generate synthetic data with few-shot prompts, leveraging the OpenAI, LangChain, and Pydantic libraries to structure and validate outputs.

3. **Iterative Batch Processing**: To address token limitations in GPT-4o Mini's input and output, data generation is performed iteratively in batches. Simulated seeds and sample prompts ensure coverage and collection of enough records across batches.

This process aims to identify potential fraud events by generating group of transactions that align with specific fraud patterns. Rather than producing single fraud transactions independent of others—such as bootstrapping individual transactions or simply finding similar ones—this method focuses on generating interrelated transaction groups that exhibit coherent fraud patterns. Details on the pattern recognition methodology are provided in Section 3.5.2.

Another important consideration in this process is the cost of fine-tuning an existing LLM and using prompt responses to generate synthetic data. Compared to GPT-4o, which costs $1.50 per million input tokens and $5 per million output tokens with the batch API, GPT-4o Mini offers a more affordable pricing structure at $0.075 per million input tokens and $0.30 per million output tokens (OpenAI, Pricing, 2024). Further details on cost and associated data risks will be discussed in Section 3. 9.

3.5.1 Fine-Tune LLMs

On September 23, 2024, OpenAI announced the availability of fine-tuning for GPT-4o and its cost-efficient small model GPT-4o Mini, providing developers with enhanced customization capabilities. Fine-tuning allows the model to adapt its response structure and tone to match specific requirements or follow complex, domain-specific instructions using just a few dozen examples in the training dataset (OpenAI, Fine-tuning now available for GPT-4o, 2024).

Another great benefits for having the fine-tuning with users' data to further train the pretrain model are significantly improve the few-slot learning in the prompt. Once the model has been fine-tined with training data, the model does not require to have many examples in the prompt and guidance to do the prompting task, which save costs and enable lower latency request (OpenAI, Fine-tuning, n.d.).

3.5.2 Pattern Recognition and Text Summarization with LLMs

Pretrained LLMs have shown the ability to recognize patterns through in-context learning without additional fine-tuning. By providing few-shot input-output examples as prompts, these models can extend a given pattern by predicting the next logical context (Mirchandani, et al., 2023).

In this study, the input data is in a tabular format lacking fully contextual text, so using a pretrained LLM alone would not yield well-formatted outputs. Additionally, due to the token limit in the prompt, the LLM's capacity to learn patterns is restricted by the prompt length. To address this, the fraud training data was further used to fine-tune the GPT-4o Mini model, equipping the LLM with enhanced knowledge of fraud patterns across the entire dataset. Below is a demonstration of how the LLM interprets the prompt and extracts underlying fraud patterns to generate synthetic data:

1. **Setup**: Configure a few-shot prompt using 2% of the fraud training examples (200 transactions) and adjust the fine-tuned LLM's hyperparameters to minimize output variation, ensuring consistency in pattern recognition.
2. **Pattern Identification**: Use the fine-tuned LLM to identify distinctive fraud patterns by summarizing the top 10 patterns, accompanied by explanations and illustrative examples in text (see Figure 11).
3. **Synthetic Data Generation**: Apply the same setup to generate 20 synthetic fraud transactions and validate these to ensure they match the fraud patterns identified in Step 2 (see Figure 12).
4. **Iteration**: Repeat Steps 1–3 with varied sets of fraud training examples to produce additional synthetic fraud transactions, enriching the dataset.

The 20 generated fraud transactions align with the major fraud patterns identified. This demonstration represents the initial iteration, with detailed results of the final generated data discussed in Section 4.

```
Assistant: Here are 10 rules that encapsulate key fraud patterns from the provided transaction data:

1. **Online Retail Transactions**:
   - **Description**: Transactions made through online retail platforms (e.g., ebay.com, alibaba.com, etc.) are often fraudulen
t.
   - **Reason**: The nature of online retail allows for anonymity and less stringent verification processes.
   - **Example**: A transaction on ebay.com for $102.82 isFraud: True.

2. **High Transaction Amounts**:
   - **Description**: Transactions with high amounts (e.g., above $100) are more likely to be fraudulent.
   - **Reason**: Fraudsters tend to make larger transactions to maximize their gains.
   - **Example**: A transaction of $342.73 at EZ Putt Putt isFraud: True.

3. **Recent Account Activity**:
   - **Description**: Transactions from accounts that are relatively new (e.g., less than 500 days since opening) are often fra
udulent.
   - **Reason**: New accounts are less likely to have established trust and may be set up specifically for fraud.
   - **Example**: A transaction from an account that is 285 days old on ebay.com isFraud: True.

4. **Low Available Money**:
   - **Description**: Transactions where the account has low available funds (e.g., below $1000) are more likely to be fraudule
nt.
   - **Reason**: Fraudsters may attempt to exploit accounts with insufficient funds, leading to chargebacks or disputes.
   - **Example**: A transaction with available money of $195.56 isFraud: True.

5. **Frequent Transactions from the Same Merchant**:
   - **Description**: Multiple transactions in a short time frame from the same merchant indicate potential fraud.
   - **Reason**: Repeated attempts to charge the same merchant suggest a pattern of abuse.
   - **Example**: Two transactions from Uber within the same day areFraud: True.

6. **Use of Rideshare Services**:
   - **Description**: Transactions involving rideshare services (e.g., Uber, Lyft) are frequently fraudulent.
   - **Reason**: These services often involve quick, high-risk transactions that can be exploited easily.
   - **Example**: A transaction of $694.09 with Uber isFraud: True.

7. **Multiple Transactions with Similar Amounts**:
   - **Description**: Transactions that have similar amounts (e.g., within $10 of each other) are indicative of fraud.
   - **Reason**: This can indicate attempts to test the system or exploit vulnerabilities in payment processing.
   - **Example**: Two transactions of $131.63 and $131.63 from American Airlines areFraud: True.

8. **Transactions with Zero Amounts**:
   - **Description**: Any transaction with a zero amount is considered fraudulent.
   - **Reason**: Zero transactions do not represent legitimate purchases and are often used as a testing mechanism by fraudster
s.
   - **Example**: A transaction of $0 from Boston Fries isFraud: True.

9. **Transactions with High Frequency in a Short Time Frame**:
   - **Description**: High-frequency transactions within a short time period (e.g., multiple within a day) indicate potential f
raud.
   - **Reason**: Rapid transactions can overwhelm detection systems and signal malicious intent.
   - **Example**: Three transactions from target.com on the same day areFraud: True.

10. **Use of Fast Food Merchants**:
    - **Description**: Transactions made at fast food establishments (e.g., McDonald's, Five Guys) are often fraudulent.
    - **Reason**: These transactions tend to be small but frequent, making them ideal for fraudsters looking for quick gains.
    - **Example**: A transaction of $223.51 at McDonald's isFraud: True.

These rules highlight patterns that can help in identifying fraudulent transactions effectively.
```

Figure 11. Fraud Patterns Generated by Fine-Tuned GPT-4o Mini Model

	accountNumber	availableMoney	transactionAmount	merchantName	posEntryMode	merchantCategoryCode	cardLast4Digits	DAYsinceOpen	isFraud
0	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
1	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
2	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
3	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
4	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
5	1234567890	-200.75	95.50	walmart.com	1	online_retail	6789	300	True
6	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
7	1234567890	-500.75	112.34	walmart.com	1	online_retail	6789	150	True
8	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
9	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
10	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
11	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
12	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
13	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
14	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
15	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
16	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
17	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
18	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
19	7408401263	-250.75	150.75	oldnavy.com	1	online_retail	1234	200	True

Figure 12. Initial Batch of Synthetic Data

3.5.3 Addressing Data Imbalance with Synthetic Oversampling

As highlighted in Section 2.2, numerous studies indicate that data imbalance can negatively impact the performance of machine learning and deep learning models. This issue is particularly prevalent in credit card fraud detection, where the fraudulent transactions are often significantly outnumbered by legitimate ones. A common solution is to oversample the minority class to enhance model performance and prediction accuracy (Mondal, Haque, Hassan, & Shatabda, 2021). To address the data imbalance issue in financial fraud transaction datasets, this research utilizes the GPT-4o-mini fine-tuned model to identify fraud transaction patterns and generate synthetic data to oversample the minority fraud class.

The original training dataset contains 10,461 fraud transactions and 641,051 non-fraud transactions. Using the fine-tuned GPT-4o Mini model, generating 1,000 synthetic transactions across 9 variables costs approximately $10. If the LLM were used to

generate synthetic data for all fraud transactions to achieve full oversampling, the cost would be around $6,306. However, the goal here is to improve prediction performance by selectively generating additional synthetic fraud transactions to enhance training data quality. By focusing on high-quality fraud patterns, this study aims to generate approximately 1,000 (or 10% of fraud training data) additional synthetic fraud data, which will then be applied to the best-performing traditional oversampling methodology (mentioned in section 3.4.1) to assess whether the balanced data improves overall model performance on the test dataset.

3.6 Hybrid Models Combining LLMs

A key advantage of LLMs is their capacity to recognize and interpret language patterns. In previous sections, LLMs were utilized to enhance fraud pattern recognition and improve data quality through synthetic data generation. Here, LLMs are employed to gain a deeper understanding of the relationships between categorical variables and the target fraud variable, aiming to boost overall model performance.

3.6.1 Categorical Embeddings

Traditional machine learning models require categorical attributes to be transformed into numerical representations to enable arithmetic operations. However, for categorical variables, unlike ordinal variables, the numerical values often fail to capture the true meaning of the categories. In 2024 September, Huesmann and Linsen proposed using Large Language Models (LLMs) to transform categorical variables into feature vectors. These vectors provide a more interpretable and contextually meaningful representation of the data compared to traditional numerical transformations (Huesmann & Linsen, 2024).

In this study, the final selected variables included two key categorical variables with textual significance: merchant name and merchant category code. These variables carry distinct potential for indicating fraudulent activity. Based on the sample fraud patterns generated in Section 3.5.2, merchant categories such as online retail, fast food, and rideshare services exhibit a higher likelihood of fraud occurrences. By using text understanding vectors, rather than arbitrary numerical representations, the model can gain deeper insights into these categorical variables, which may enhance its predictive power and improve fraud detection accuracy.

To enhance model performance by embedding categorical variables using LLMs, five different models will be tested: fine-tuned GPT-4o Mini, Gemini, Gemma, BERT, and LLaMA 3.2. The GPT and Gemini models will be accessed via API, connecting directly to the large language models. For Gemma, BERT, and LLaMA, OLLaMA library will be used to download and run the models locally for embedding tasks. Since each LLM has been pre-trained on different datasets, as discussed in Section 2.6.1, this comparison will not only assess model performance but also evaluate the varying text understanding between merchant categories and fraud patterns across different models, aiming to identify the one that offers the most significant performance improvement.

3.6.2 Principal Component Analysis of Embeddings Outputs

After gaining the feature vector representing the categorical variables, the number of dimensions will be significantly increased based on the complexity of the text. Huesmann and Linsen's paper mentioned to use multiple dimensionality reduction such as Principal Component Analytics (PCA), Uniform Maniford approximation and Projection (UMAP), Multidimensional Scaling (MDS) or t-distributed stochastic

neighbor embedding(t-SNE) (Huesmann & Linsen, 2024). Principal Component Analysis (PCA) is a linear dimensionality reduction technique that uses singular value decomposition to project data into a lower-dimensional space while preserving as much of the original information as possible. Although individual components may be difficult to interpret, collectively they retain the underlying structure of the high-dimensional space (Scikit Learn, PCA, 2024). In this experiment, PCA will be applied to reduce the dimensionality of the categorical embeddings, compressing each categorical variable to 10 dimensions. This reduction is intended to simplify the comparison process while maintaining the essential information.

3.6.4 Hybrid Models Utilizing Large Language Models

The hybrid model incorporates three key novel methodologies: first, it utilizes rebalanced training data enriched with LLM-generated synthetic data; second, it selects the optimal categorical embeddings produced by the LLM for enhanced representation; and third, it combines numerical variables with dimensionally reduced categorical embeddings to refit the best-performing model algorithms as outlined in Section 3.4. This approach is expected to yield the best performance compared to all other tested combinations.

3.7 Evaluating Model Performance

To evaluate model performance, two main approaches will be used: (1) comparing key performance metrics to assess model effectiveness, and (2) simulating the training and prediction processes using various combinations of data preprocessing methods and modeling algorithms, followed by comparing performance metrics on the test dataset.

3.7.1 Key Performance Metrics for Model Evaluation

In this research, 8 key performance metrics below will be evaluated the model efficiency and cost.

Accuracy: measures the proportion of correct classifications out of the total number of records (Ahmadi, 2023).

$$Accuracy = \frac{True\ positive + True\ Negative}{Total\ number\ of\ records} \quad (8)$$

AUC (Area Under the Curve): AUC measures a binary classifier's ability to distinguish between positive and negative classes. If AUC = 1, this indicates perfect classification, where the model correctly identifies all positive and negative class points. An AUC between 0.5 and 1 suggests that the classifier has a strong likelihood of distinguishing between positive and negative classes, effectively detecting true positives and true negatives more often than false positives and false negatives. An AUC of 0.5 means the classifier performs no better than random chance, lacking the ability to differentiate between positive and negative classes (Bhandari, 2024).

Recall: measures the proportion of true positives among all actual positive samples (Ahmadi, 2023).

$$Recall = \frac{True\ positive}{True\ Positive + False\ Negative} \quad (9)$$

Precision: measures the proportion of true positives among all predicted positive samples (Ahmadi, 2023).

$$Precision = \frac{True\ positive}{True\ positive + False\ Positive} \quad (10)$$

F1: is the harmonic mean of precision and recall (Ahmadi, 2023).

$$F1\ Score = \frac{2\ \times Recall\ \times Precision}{Recall + Precision} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (11)$$

Kappa: Cohen's Kappa statistic measures of agreement between two dependent categorical samples. Interpretation of Kappa values is as follows: below 0 indicates poor agreement, 0–0.20 represents slight agreement, 0.21–0.40 indicates fair agreement, 0.41–0.60 signifies moderate agreement, 0.61–0.80 reflects substantial agreement, and 0.81–1 indicates almost perfect agreement (Cohen's Kappa, n.d.).

$$K = \frac{P_o - P_e}{1 - P_e}, where\ P_{o\ is\ the\ observed\ agreement,} P_{e\ is\ the\ expected\ agreement} \quad (12)$$

MCC: The Matthews Correlation Coefficient (MCC) assesses model performance by taking the entire confusion matrix into account, offering a balanced evaluation especially useful for imbalanced datasets. MCC values range from -1 to 1, where 1 signifies perfect prediction, 0 indicates performance equivalent to random guessing, and -1 represents complete disagreement between observed and predicted values (Pradhan, 2024).

$$MCC = \frac{TP\ \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (13)$$

Where TP is true positive, TN is true negative, FP is false positive and FN is False negative.

Training Time: measures the time cost to train the model. The longer training time will have higher cost of computation power.

3.7.2 Training Model and Prediction Process

The second performance evaluation approach involves comparing performance metrics on the test dataset by simulating various combinations of data imbalance treatments and model algorithms. This process helps assess whether LLMs can effectively enhance model performance. The study follows these steps to compare performance metrics on the test data and validate the first two hypotheses:

1. **Traditional Model and Oversampling Evaluation**: Iterate 14 traditional model algorithms with 6 traditional oversampling techniques, comparing 8 performance metrics to identify the best-performing model and oversampling combination.
2. **LLM-Based Synthetic Oversampling Comparison**: Apply the novel LLM-based synthetic oversampling methodology to all 14 traditional models and compare the 8-performance metrics against the best results from Step 1. This step tests the first hypothesis, evaluating whether LLM-generated synthetic data can improve model performance by addressing data imbalance.
3. **LLM-Based Categorical Embedding Application**: Integrate 5 different LLM-based categorical embeddings across all 14 traditional model algorithms combined with 7 oversampling techniques, then compare performance metrics to the best result from Step 2. This tests the second hypothesis, determining if the LLM-based hybrid model can outperform traditional methods and identifying the optimal model for testing the third hypothesis.

3.8 Fraud Detection Optimization Methodology

Optimizing model performance in fraud detection involves directly reducing false predictions, including both false positives and false negatives. This section introduces a

novel methodology aimed at maximizing profit by balancing associated risks, costs, and enhanced model performance, offering analytical insights to support effective decision-making.

3.8.1 Analysis of False-Positive Rate Study

False positives occur when a model incorrectly flags a legitimate transaction as fraudulent, indicating an overly sensitive model. Reducing false positives requires building a more robust model, potentially through techniques like regularization or penalty adjustments. Additionally, accurately measuring the gains from reducing false positives can guide decisions on the extent of reduction needed.

High false-positive rates can lead to frequent transaction blocks, causing inconvenience for customers. In the worst case, frustrated customers may abandon their credit cards and switch to competitors, resulting in significant lifetime value loss for the company. Additionally, fraud investigation involves semi-manual processes, where legal protocols often necessitate manual reviews, adding operational costs. Reducing false positives thus benefits both customer retention and operational efficiency by lowering these costs.

In this study, regularization or penalty terms will be incorporated into the final model in section 3.7.2, assuming that correcting a false positive yield a cost represented by α. This approach will allow us to assess the financial impact of reducing false positives within a structured decision-making framework.

3.8.2 Analysis of False-Negative Rate Study

False negatives occur when a model fails to identify a fraudulent transaction, incorrectly classifying it as normal. This often indicates that the model is overly conservative and unable to capture certain fraud patterns. Reducing false negatives may require additional training or further hyperparameter tuning to improve sensitivity. The economic benefit of reducing false negatives lies in preventing actual fraud losses.

In this study, hyperparameter tuning will be applied to the final model in the section 3.7.2, assuming that the cost from correcting a single false negative is represented by β. This approach aims to quantify the financial impact of reducing false negatives while optimizing the model's performance.

3.8.3 Optimizing Fraud Risk Management Through Iterative Improvements

In fraud detection, reducing false positives often leads to an increase in false negatives, and vice versa. Achieving an optimal model performance with minimal loss requires a carefully balanced approach. An iterative optimization process can help find the ideal balance between minimizing false positives and false negatives, with predefined values of α and β representing the losses associated with each type of error, respectively. This approach aims to reduce overall loss by fine-tuning the model to achieve the best trade-off between these errors. The following steps outline this optimization process, which also addresses the third hypothesis:

1. **Hyperparameter Tuning**: Utilize Bayesian optimization to fine-tune critical hyperparameters, such as pruning, minimum samples per leaf, maximum features, and maximum depth. Each hyperparameter configuration impacts the model's sensitivity and robustness, influencing false positive and false

negative rates. After each adjustment, the resulting false prediction rates are used to calculate the associated loss.

2. **Loss Calculation for Each Iteration**: For each set of hyperparameters, calculate loss using the formula:

$$Total\ Loss_{\ i} = FP_i \times \alpha + FN_i \times \beta\ , \quad (14)$$

where:

- FP_i is the count of false positives after tuning hyperparameters.
- FN_i is the count of false negatives after tuning hyperparameters.

3. **Iterative Optimization**: Continue iterating through different hyperparameter combinations, calculating the loss for each to refine the model's robustness. Rather than optimizing for standard metrics like MCC or F1-score, this approach focuses on minimizing the loss function. By selecting hyperparameters that yield the most favorable balance between false positives and false negatives, the model aligns with cost-efficient objectives and minimizes overall risk.

3.9 Limitations and Challenges

The methodologies proposed in this research include LLM-based generation of synthetic tabular data to enhance model performance, LLM-based embeddings for categorical variables to better capture class distinctions in language, and a new hyperparameter tuning criterion centered on profit maximization. However, these tools and processes encounter specific limitations and challenges, as outlined below:

3.9.1 Novel Methodology Concerns

The LLM-based synthetic data generation process encounters challenges with large datasets, particularly those exceeding 10,000 dimensions, as it becomes impractical to upload the entire dataset to OpenAI for fine-tuning due to loading limitations. Additionally, few-shot prompting cannot capture the full scope of information within each variable. For LLM-based categorical embeddings after PCA, interpreting the transformed feature vectors can be difficult, limiting their transparency. Lastly, the new profit-maximization stop criterion may lead to overfitting in cases where a small company has concentrated fraud in specific types, as profit-driven criteria might favor highly specialized models rather than generalized solutions.

3.9.2 System Requirements and Execution Costs

This study employed small, efficient models, but testing larger LLMs requires GPUs and expanded memory capacity to handle the increased computational demands. For API-connected LLMs, like OpenAI's models, larger LLMs come at a substantially higher cost; for instance, GPT-4o is 16 times more expensive than GPT-4o Mini, while the latest OpenAI o1-Preview model is 200 times more expensive, priced at $15 per million input tokens and $60 per million output tokens (OpenAI, Pricing, 2024). For companies needing continuous production updates via LLMs, the cost of utilizing these larger models may be prohibitive.

3.9.3 OpenAI Data Generator Limitation

In this study, OpenAI was utilized as the primary tool for data generation. However, the existing create_openai_data_generator function only supports temperature as a hyperparameter, limiting the ability to experiment with other impactful

hyperparameters, such as top_k or top_p. This restricts the capacity to observe how adjustments in these parameters might influence the data generation process.

3.9.4 Assumption to Cover all Fraud Patterns in the Few-Slots Prompt

During the Few-Slots prompt, the study on purposely generates 20 data by every 200 transaction examples due to the limitation of token input in the prompt. The assumption is all transactions were broken into different batches to generate the data so all fraud patterns could be covered. However, there is no effective way to tell whether all patterns were recognized.

3.9.5 Data Privacy Risk

For API-based LLMs like OpenAI, uploading data to an external server for model fine-tuning poses potential data privacy risks. Specifically, once data is uploaded, it could be accessed or downloaded by logging into the OpenAI platform, raising concerns about data leakage, especially if the data is sensitive.

Chapter 4—Results

4.1 Introduction

This chapter presents a comprehensive analysis of the experimental results from this research. It begins with an examination of the generated synthetic data, analyzing variations resulting from different hyperparameter settings and assessing whether incorporating synthetic data enhances model performance. Subsequently, it investigates the impact of various pretrained large language model (LLM)-based categorical embedding techniques on model outcomes. Finally, a customized metric is developed to guide hyperparameter tuning, focusing on optimizing cost-efficiency by balancing false positives and false negatives.

4.2 LLM based Categorical Variable Embedding

This process incorporates 5 different pretrained large language model (LLM) to support the following research hypothesis:

H1: The proposed hybrid model, incorporating categorical embedding techniques from LLaMA3.2, Gemma, OpenAI, BERT, and Gemini, combined with traditional data balancing methods, achieves superior performance in classification tasks compared to other traditional algorithms.

4.2.1 Model Performance using Traditional Methods

This section explores model performance using traditional machine learning and deep learning techniques, along with pre-processing techniques such as oversampling. The objective is to identify the optimal combination of classifiers and oversampling

methods to serve as a baseline. This baseline will provide a foundation for evaluating whether novel methods can further enhance model performance.

4.2.1.1 Classification Models Performance

This research begins by evaluating model performance using traditional data preprocessing and binary classification methods. As described in Section 3.4, the experiment simulates 14 traditional classifiers paired with 7 different data preprocessing techniques. Model performance is assessed on both the training and test datasets using evaluation metrics such as recall, precision, F1-score, Matthew's correlation coefficient (MCC), kappa, and model training time. This simulation aims to identify the best-performing model configuration using conventional methods.

The tables below present the simulation results, highlighting the top 10 classification models combined with 7 different data preprocessing techniques. For reference, "Original" represents the training set, while "New" represents the test set.

Result 1:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9996	0.9868	1.0000	0.8129	1.0000	0.1074	0.9991	0.8400	0.9996	0.1904	0.9991	0.1877	0.9991	0.2976	18.575
1	rf	0.9994	0.9862	1.0000	0.8210	1.0000	0.0613	0.9988	0.8163	0.9994	0.1141	0.9988	0.1123	0.9988	0.2215	25.528
2	dt	0.9902	0.9752	0.9902	0.5691	1.0000	0.1508	0.9808	0.1487	0.9903	0.1497	0.9804	0.1371	0.9806	0.1371	1.044
3	knn	0.9708	0.9513	0.9897	0.6130	1.0000	0.2188	0.9448	0.0783	0.9716	0.1153	0.9416	0.0960	0.9432	0.1097	2.140
4	xgboost	0.8443	0.8289	0.9239	0.8414	0.8674	0.6764	0.8291	0.0557	0.8478	0.1029	0.6886	0.0782	0.6894	0.1593	1.856
5	lightgbm	0.7771	0.7830	0.8643	0.8434	0.7892	0.7439	0.7706	0.0481	0.7798	0.0904	0.5543	0.0650	0.5544	0.1512	1.877
6	gbc	0.7196	0.7541	0.7952	0.7965	0.7086	0.6881	0.7245	0.0397	0.7165	0.0751	0.4391	0.0490	0.4392	0.1222	30.446
7	ada	0.6887	0.6883	0.7579	0.7597	0.7100	0.6994	0.6810	0.0320	0.6952	0.0611	0.3774	0.0343	0.3777	0.0995	7.855
8	lr	0.6296	0.7300	0.6709	0.6726	0.5504	0.5348	0.6540	0.0286	0.5978	0.0543	0.2592	0.0276	0.2625	0.0720	2.869
9	ridge	0.6291	0.7375	0.0000	0.6334	0.5416	0.5261	0.6565	0.0290	0.5935	0.0549	0.2582	0.0282	0.2622	0.0724	0.179

Table 1. Classification models comparison with imbalanced data

Result 2:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9996	0.9868	1.0000	0.8129	1.0000	0.1074	0.9991	0.8400	0.9996	0.1904	0.9991	0.1877	0.9991	0.2976	17.998
1	rf	0.9994	0.9862	1.0000	0.8210	1.0000	0.0613	0.9988	0.8163	0.9994	0.1141	0.9988	0.1123	0.9988	0.2215	25.823
2	dt	0.9902	0.9752	0.9902	0.5691	1.0000	0.1508	0.9808	0.1487	0.9903	0.1497	0.9804	0.1371	0.9806	0.1371	1.083
3	knn	0.9708	0.9513	0.9897	0.6130	1.0000	0.2188	0.9448	0.0783	0.9716	0.1153	0.9416	0.0960	0.9432	0.1097	2.107
4	xgboost	0.8443	0.8289	0.9239	0.8414	0.8674	0.6764	0.8291	0.0557	0.8478	0.1029	0.6886	0.0782	0.6894	0.1593	1.809
5	lightgbm	0.7771	0.7830	0.8643	0.8434	0.7892	0.7439	0.7706	0.0481	0.7798	0.0904	0.5543	0.0650	0.5544	0.1512	1.768
6	gbc	0.7196	0.7541	0.7952	0.7965	0.7086	0.6881	0.7245	0.0397	0.7165	0.0751	0.4391	0.0490	0.4392	0.1222	30.085
7	ada	0.6887	0.6883	0.7579	0.7597	0.7100	0.6994	0.6810	0.0320	0.6952	0.0611	0.3774	0.0343	0.3777	0.0995	7.492
8	lr	0.6296	0.7300	0.6709	0.6726	0.5504	0.5348	0.6540	0.0286	0.5978	0.0543	0.2592	0.0276	0.2625	0.0720	2.861
9	ridge	0.6291	0.7375	0.0000	0.6334	0.5416	0.5261	0.6565	0.0290	0.5935	0.0549	0.2582	0.0282	0.2622	0.0724	0.187

Table 2. Classification models comparison with random oversampling data

Result 3:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9836	0.9769	0.9986	0.8061	0.9903	0.2418	0.9773	0.2253	0.9837	0.2333	0.9673	0.2216	0.9674	0.2217	20.560
1	rf	0.9774	0.9744	0.9974	0.8009	0.9846	0.2301	0.9705	0.1877	0.9775	0.2068	0.9547	0.1939	0.9548	0.1949	33.284
2	dt	0.9452	0.9357	0.9452	0.6158	0.9581	0.2863	0.9341	0.0714	0.9459	0.1144	0.8905	0.0933	0.8908	0.1183	1.519
3	knn	0.9383	0.9034	0.9769	0.6468	0.9928	0.3241	0.8953	0.0514	0.9415	0.0887	0.8767	0.0653	0.8819	0.0979	2.165
4	xgboost	0.8745	0.8701	0.9510	0.7852	0.8740	0.5220	0.8749	0.0580	0.8744	0.1044	0.7490	0.0804	0.7490	0.1410	1.870
5	lightgbm	0.8279	0.8320	0.9139	0.7700	0.8301	0.5588	0.8265	0.0478	0.8283	0.0880	0.6558	0.0630	0.6559	0.1258	2.026
6	gbc	0.7370	0.7117	0.8177	0.7156	0.7808	0.6053	0.7179	0.0301	0.7480	0.0574	0.4739	0.0306	0.4758	0.0838	32.393
7	ada	0.7062	0.6500	0.7794	0.6833	0.7809	0.6242	0.6794	0.0256	0.7266	0.0492	0.4124	0.0219	0.4171	0.0687	8.539
8	qda	0.6420	0.7140	0.6874	0.6371	0.5896	0.5051	0.6586	0.0256	0.6222	0.0487	0.2840	0.0217	0.2856	0.0588	0.310
9	lr	0.6409	0.7170	0.6867	0.6462	0.5882	0.5184	0.6574	0.0265	0.6209	0.0505	0.2817	0.0235	0.2833	0.0632	2.955

Table 3. Classification models comparison with SMOTE oversampling data

Result 4:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9913	0.9851	0.9981	0.8102	0.9915	0.1922	0.9911	0.4636	0.9913	0.2718	0.9827	0.2655	0.9827	0.2922	20.028
1	rf	0.9904	0.9851	0.9980	0.8132	0.9898	0.1467	0.9909	0.4599	0.9904	0.2225	0.9808	0.2170	0.9808	0.2540	32.113
2	dt	0.9716	0.9574	0.9716	0.5948	0.9798	0.2214	0.9639	0.0931	0.9718	0.1311	0.9431	0.1129	0.9432	0.1242	1.348
3	knn	0.9699	0.9499	0.9869	0.6300	0.9945	0.2393	0.9479	0.0816	0.9706	0.1217	0.9398	0.1022	0.9410	0.1183	2.242
4	xgboost	0.9160	0.9112	0.9747	0.7847	0.9155	0.4494	0.9165	0.0746	0.9160	0.1280	0.8321	0.1058	0.8321	0.1556	1.913
5	lightgbm	0.8636	0.8614	0.9421	0.7788	0.8656	0.5082	0.8622	0.0531	0.8639	0.0962	0.7272	0.0718	0.7272	0.1296	2.090
6	gbc	0.7671	0.7513	0.8507	0.7340	0.7900	0.5777	0.7555	0.0334	0.7723	0.0631	0.5343	0.0367	0.5349	0.0914	32.647
7	ada	0.7090	0.6630	0.7833	0.7017	0.7585	0.6452	0.6901	0.0274	0.7227	0.0526	0.4179	0.0255	0.4200	0.0778	8.885
8	qda	0.6434	0.6661	0.6858	0.6127	0.6258	0.5046	0.6487	0.0219	0.6370	0.0420	0.2869	0.0146	0.2871	0.0439	0.336
9	lr	0.6227	0.6978	0.6710	0.6258	0.5596	0.5015	0.6404	0.0241	0.5973	0.0459	0.2454	0.0188	0.2474	0.0527	3.547

Table 4. Classification models comparison with borderline SMOTE oversampling data

Result 5:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9831	0.9768	0.9985	0.8071	0.9908	0.2357	0.9757	0.2197	0.9832	0.2274	0.9661	0.2157	0.9663	0.2158	20.586
1	rf	0.9769	0.9735	0.9973	0.7971	0.9853	0.2275	0.9691	0.1771	0.9771	0.1991	0.9539	0.1859	0.9540	0.1874	34.356
2	dt	0.9457	0.9382	0.9457	0.6193	0.9587	0.2909	0.9343	0.0757	0.9464	0.1201	0.8914	0.0994	0.8917	0.1243	1.571
3	knn	0.9376	0.9015	0.9763	0.6466	0.9941	0.3241	0.8932	0.0503	0.9410	0.0871	0.8753	0.0636	0.8809	0.0962	2.383
4	xgboost	0.8716	0.8643	0.9488	0.7887	0.8742	0.5358	0.8696	0.0569	0.8719	0.1028	0.7432	0.0786	0.7432	0.1409	1.844
5	lightgbm	0.8243	0.8277	0.9106	0.7647	0.8267	0.5455	0.8226	0.0456	0.8246	0.0841	0.6485	0.0589	0.6485	0.1191	2.693
6	gbc	0.7402	0.7198	0.8217	0.7117	0.7798	0.5925	0.7224	0.0304	0.7500	0.0578	0.4803	0.0311	0.4819	0.0834	31.907
7	ada	0.7164	0.6614	0.7904	0.6818	0.7859	0.6166	0.6899	0.0261	0.7348	0.0502	0.4329	0.0230	0.4372	0.0702	8.591
8	qda	0.6530	0.7243	0.6983	0.6390	0.6023	0.4918	0.6701	0.0259	0.6344	0.0492	0.3060	0.0223	0.3075	0.0588	0.351
9	lr	0.6515	0.7211	0.6993	0.6507	0.6067	0.5204	0.6663	0.0270	0.6351	0.0514	0.3030	0.0245	0.3043	0.0651	2.910

Table 5. Classification models comparison with ADASYN oversampling data

Result 6:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9831	0.9768	0.9985	0.8071	0.9908	0.2357	0.9757	0.2197	0.9832	0.2274	0.9661	0.2157	0.9663	0.2158	20.940
1	rf	0.9769	0.9735	0.9973	0.7971	0.9853	0.2275	0.9691	0.1771	0.9771	0.1991	0.9539	0.1859	0.9540	0.1874	33.372
2	dt	0.9457	0.9382	0.9457	0.6193	0.9587	0.2909	0.9343	0.0757	0.9464	0.1201	0.8914	0.0994	0.8917	0.1243	1.488
3	knn	0.9376	0.9015	0.9763	0.6466	0.9941	0.3241	0.8932	0.0503	0.9410	0.0871	0.8753	0.0636	0.8809	0.0962	2.143
4	xgboost	0.8716	0.8643	0.9488	0.7887	0.8742	0.5358	0.8696	0.0569	0.8719	0.1028	0.7432	0.0786	0.7432	0.1409	1.876
5	lightgbm	0.8243	0.8277	0.9106	0.7647	0.8267	0.5455	0.8226	0.0456	0.8246	0.0841	0.6485	0.0589	0.6485	0.1191	1.875
6	gbc	0.7402	0.7198	0.8217	0.7117	0.7798	0.5925	0.7224	0.0304	0.7500	0.0578	0.4803	0.0311	0.4819	0.0834	31.109
7	ada	0.7164	0.6614	0.7904	0.6818	0.7859	0.6166	0.6899	0.0261	0.7348	0.0502	0.4329	0.0230	0.4372	0.0702	8.088
8	qda	0.6530	0.7243	0.6983	0.6390	0.6023	0.4918	0.6701	0.0259	0.6344	0.0492	0.3060	0.0223	0.3075	0.0588	0.305
9	lr	0.6515	0.7211	0.6993	0.6507	0.6067	0.5204	0.6663	0.0270	0.6351	0.0514	0.3030	0.0245	0.3043	0.0651	2.888

Table 6. Classification models comparison with SMOTETOMEK oversampling data

Result 7:

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	et	0.9869	0.9684	0.9992	0.8123	0.9923	0.2868	0.9826	0.1640	0.9875	0.2087	0.9736	0.1938	0.9737	0.2017	20.560
1	rf	0.9802	0.9636	0.9982	0.8010	0.9876	0.2786	0.9749	0.1350	0.9812	0.1818	0.9603	0.1655	0.9604	0.1770	29.726
2	knn	0.9760	0.8772	0.9968	0.6560	0.9978	0.3875	0.9579	0.0470	0.9774	0.0838	0.9518	0.0595	0.9527	0.1002	1.958
3	dt	0.9494	0.9256	0.9487	0.6268	0.9630	0.3190	0.9413	0.0670	0.9520	0.1107	0.8984	0.0888	0.8987	0.1195	1.398
4	xgboost	0.8855	0.8489	0.9579	0.7874	0.8945	0.5573	0.8870	0.0529	0.8907	0.0967	0.7704	0.0721	0.7705	0.1364	1.802
5	lightgbm	0.8377	0.8026	0.9221	0.7649	0.8504	0.5798	0.8404	0.0421	0.8454	0.0785	0.6746	0.0529	0.6747	0.1153	2.017
6	gbc	0.7503	0.6781	0.8300	0.7171	0.8084	0.6467	0.7380	0.0288	0.7716	0.0551	0.4975	0.0281	0.5000	0.0829	28.677
7	ada	0.7200	0.6326	0.7922	0.6892	0.8017	0.6462	0.7033	0.0252	0.7493	0.0486	0.4354	0.0212	0.4402	0.0689	7.381
8	lr	0.6582	0.6691	0.7039	0.6512	0.6530	0.5711	0.6795	0.0249	0.6660	0.0477	0.3163	0.0204	0.3166	0.0613	3.615
9	ridge	0.6545	0.6696	0.0000	0.6183	0.6459	0.5654	0.6771	0.0247	0.6611	0.0473	0.3092	0.0201	0.3096	0.0600	0.169

Table 7. Classification models comparison with SMOTEEN oversampling data

Based on the results above, several key findings emerge:

Training Time: The Extra Trees, Random Forest, and Gradient Boost models exhibit significantly longer training times compared to other models.

Effectiveness of Borderline SMOTE: Models that use the Borderline SMOTE oversampling technique demonstrate higher AUC, accuracy, F1-score, MCC, and Kappa scores compared to other oversampling methods. Notably, Borderline SMOTE consistently outperforms standard SMOTE across all 10 classifiers. As discussed in Section 3.4.1, this is because Borderline SMOTE focuses on oversampling minority class samples near the decision boundary, which enhances model performance on imbalanced data.

Best Model on Test Set: The Extra Trees classifier with Borderline SMOTE achieves the highest F1, Kappa, and MCC scores on the test set, along with relatively high accuracy and AUC. However, the metrics reveal a trade-off between precision and recall. The Extra Trees classifier has high precision but lower recall, making it highly sensitive and potentially prone to high false positives. In a fraud detection context, excessive false positives could lead to customer dissatisfaction, as legitimate transactions may be frequently blocked, prompting customers to switch to other providers. Thus, the most accurate model may not always be the most practical.

Implications for Model Selection: This experiment suggests that the SMOTE oversampling method generally improves model performance metrics. The next step is to explore ways to further enhance model performance and identify the best model that balances predictive accuracy with practical applicability.

4.2.1.2 Comparison of Cluster-Based Models

An alternative approach to improving model performance is to explore methods beyond traditional classification for identifying fraud. In fraud detection, unsupervised anomaly detection models can be effective, as they do not rely on labeled data and can detect unusual patterns that may indicate fraudulent activity. In this experiment, two unsupervised models—**Copula-Based Outlier Detection (COPOD)** and **Isolation Forest**—are applied to the training data. The results are summarized below.

Model	Pre-Data Processing	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
Isolation Forest	No sampling	0.7928	0.7926	0.5382	0.5311	0.2751	0.2618	0.0221	0.0189	0.0409	0.0353	0.0115	0.0085	0.0240	0.0186	2.9920
	ROS	0.5147	0.7957	0.5147	0.5047	0.2147	0.2050	0.5367	0.0152	0.3067	0.0283	0.0293	0.0013	0.0367	0.0028	6.6642
	SMOTE	0.4835	0.7728	0.4835	0.5142	0.1835	0.2480	0.4586	0.0164	0.2621	0.0307	-0.0331	0.0036	-0.0414	0.0082	6.6586
	Borderline SMOTE	0.4877	0.7865	0.4877	0.5111	0.1877	0.2275	0.4692	0.0161	0.2681	0.0300	-0.0246	0.0030	-0.0308	0.0066	6.6011
	ADASYN	0.4836	0.7681	0.4835	0.5139	0.1835	0.2520	0.4585	0.0163	0.2621	0.0306	-0.0331	0.0034	-0.0413	0.0080	5.6093
	SMOTETOMEK	0.4836	0.7681	0.4835	0.5139	0.1835	0.2520	0.4585	0.0163	0.2621	0.0306	-0.0331	0.0034	-0.0413	0.0080	6.0014
	SMOTEEN	0.4750	0.7747	0.4880	0.5157	0.1885	0.2490	0.4918	0.0166	0.2725	0.0311	-0.0234	0.0040	-0.0300	0.0091	6.1595
COPOD	No sampling	0.4927	0.1025	0.4927	0.4910	0.8927	0.8911	0.4960	0.0142	0.6377	0.0280	-0.0146	-0.0006	-0.0243	-0.0075	5.2208
	ROS	0.4927	0.1025	0.4927	0.4910	0.8927	0.8911	0.4960	0.0142	0.6377	0.0280	-0.0146	-0.0006	-0.0243	-0.0075	2.7576
	SMOTE	0.5184	0.1238	0.5184	0.4885	0.9184	0.8640	0.5102	0.0141	0.6560	0.0278	0.0368	-0.0008	0.0614	-0.0087	3.0741
	Borderline SMOTE	0.5097	0.1158	0.5097	0.4857	0.9097	0.8666	0.5054	0.0140	0.6498	0.0276	0.0194	-0.0009	0.0324	-0.0112	2.7374
	ADASYN	0.5205	0.1259	0.5207	0.4885	0.9207	0.8620	0.5113	0.0141	0.6575	0.0278	0.0414	-0.0008	0.0689	-0.0086	2.5674
	SMOTETOMEK	0.5205	0.1259	0.5207	0.4885	0.9207	0.8620	0.5113	0.0141	0.6575	0.0278	0.0414	-0.0008	0.0689	-0.0086	2.5183
	SMOTEEN	0.5363	0.1227	0.5189	0.4876	0.9181	0.8635	0.5322	0.0141	0.6738	0.0278	0.0390	-0.0008	0.0629	-0.0094	2.6061

Table 8. Isolation Forest and COPOD Result Comparison

The experiment results indicate that both Copula-Based Outlier Detection (COPOD) and Isolation Forest models struggle to achieve high accuracy in detecting fraud. The AUC scores on the test data are consistently low, with a maximum of around 53%, which is only marginally better than random guessing. This suggests that these unsupervised anomaly detection methods are not well-suited for this specific fraud detection task when applied directly.

Additionally, various oversampling techniques were tested to examine their effects on model performance. While oversampling marginally improves the recall for

some models, other key metrics like accuracy, precision, F1-score, Kappa, and MCC remain low, particularly on the test set, highlighting the limitations of these models in handling imbalanced data effectively.

Notably, these models tend to exhibit high recall but low precision, meaning they capture more actual fraud cases but also generate a high rate of false positives. This lack of precision suggests that the models are not adequately sensitive and may fail to accurately detect fraudulent transactions, limiting their practical applicability for fraud detection.

4.2.1.3 Deep Learning Models Performance Comparison

In addition to traditional machine learning methods, deep learning approaches like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) offer alternative options for detecting fraudulent transactions. The table below presents the performance metrics for these two deep learning models across various oversampling techniques.

Model	Pre-Data Processing	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
CNN	No sampling	0.9839	0.9855	0.5000	0.6721	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	335.95
	ROS	0.6792	0.8612	0.6792	0.7675	0.4944	0.4842	0.7844	0.0507	0.6065	0.0919	0.3585	0.0674	0.3858	0.1215	666.24
	SMOTE	0.6609	0.8479	0.6609	0.7061	0.4669	0.3891	0.7628	0.0379	0.5793	0.0691	0.3217	0.0438	0.3491	0.0819	699.49
	Borderline SMOTE	0.7340	0.8799	0.7340	0.7246	0.5843	0.3850	0.8340	0.0478	0.6872	0.0851	0.4680	0.0609	0.4905	0.1014	701.06
	ADASYN	0.6708	0.8181	0.6708	0.6955	0.5246	0.4346	0.7412	0.0350	0.6144	0.0648	0.3416	0.0390	0.3572	0.0804	679.62
	SMOTETOMEK	0.6384	0.8691	0.6383	0.6910	0.4011	0.3246	0.7630	0.0374	0.5258	0.0671	0.2766	0.0422	0.3142	0.0727	682.36
	SMOTEEN	0.7093	0.8103	0.7147	0.7210	0.5905	0.4974	0.7999	0.0380	0.6794	0.0707	0.4243	0.0449	0.4407	0.0953	634.04
RNN	No sampling	0.9839	0.9855	0.5000	0.7776	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	987.77
	ROS	0.8788	0.8594	0.8788	0.7972	0.9008	0.5399	0.8629	0.0552	0.8814	0.1002	0.7577	0.0759	0.7584	0.1385	1841.77
	SMOTE	0.8363	0.8484	0.8363	0.7797	0.8272	0.5383	0.8426	0.0511	0.8348	0.0934	0.6727	0.0687	0.6728	0.1301	1829.68
	Borderline SMOTE	0.9142	0.9083	0.9142	0.7636	0.9191	0.3916	0.9102	0.0642	0.9146	0.1103	0.8284	0.0875	0.8285	0.1294	1794.55
	ADASYN	0.7904	0.8550	0.7903	0.7445	0.7222	0.4361	0.8361	0.0442	0.7750	0.0802	0.5807	0.0554	0.5862	0.1015	1764.15
	SMOTETOMEK	0.7887	0.8519	0.7887	0.7459	0.7254	0.4663	0.8305	0.0460	0.7744	0.0837	0.5774	0.0588	0.5821	0.1093	1754.42
	SMOTEEN	0.8584	0.8269	0.8584	0.7840	0.8587	0.5573	0.8685	0.0462	0.8636	0.0854	0.7165	0.0602	0.7166	0.1222	1649.80

Table 9. CNN and RNN Result Comparison

The comparison of model performance suggests that oversampling techniques help deep learning models, such as CNN and RNN, to better capture patterns in the data.

Additionally, model performance on the test set often appears higher than on the training set, possibly due to certain patterns overfitting to the test data, which may not be fully representative or may indicate underfitting in the training set.

Despite the benefits of oversampling, deep learning models (CNN and RNN) generally do not perform as well as traditional classification models for this fraud detection task. Additionally, training times for these deep learning models are significantly higher compared to traditional machine learning classifiers, highlighting a trade-off between computational cost and model performance.

4.2.2 Hybrid Model with LLM-Based Categorical Embedding Variables

From the comparison of traditional methods above, supervised classification combined with the Borderline SMOTE oversampling technique yielded the best performance. To further enhance model accuracy, the next experiment incorporates a large language model (LLM) to process categorical text data. By embedding text descriptions in categorical variables, the LLM converts them into meaningful numeric vectors, capturing more nuanced information. These vectors are then compressed into 10 principal components using PCA, optimizing computational efficiency and improving the model's predictive capabilities.

4.2.2.1 Gemma, LLaMA 3.2 and BERT Categorical Embedding from OLLaMA

This experiment leverages the OLLaMA library, which offers seamless access to a variety of pretrained models for categorical embedding. Three models—Gemma, LLaMA 3.2, and BERT—are utilized to embed categorical variables, transforming them into detailed numerical vectors. These vectors are then further compressed via PCA,

reducing dimensionality while preserving essential information for improved model efficiency and predictive accuracy.

Hybrid Gemma-Based Supervised Classifiers Result

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	rf	0.9931	0.9873	0.9975	0.8283	0.9897	0.1503	0.9966	0.8473	0.9931	0.2553	0.9863	0.2520	0.9863	0.3538	47.513
1	et	0.9931	0.9868	0.9976	0.8141	0.9898	0.1876	0.9963	0.6601	0.9931	0.2922	0.9862	0.2876	0.9862	0.3474	45.316
2	dt	0.9843	0.9732	0.9843	0.6134	0.9874	0.2428	0.9813	0.1823	0.9844	0.2082	0.9686	0.1949	0.9686	0.1970	3.844
3	knn	0.9607	0.9471	0.9825	0.6124	0.9857	0.1973	0.9387	0.0648	0.9616	0.0976	0.9214	0.0774	0.9225	0.0904	99.508
4	xgboost	0.9582	0.9641	0.9912	0.8073	0.9468	0.3415	0.9690	0.1582	0.9578	0.2163	0.9165	0.2004	0.9167	0.2161	2.485
5	lightgbm	0.9242	0.9446	0.9781	0.8039	0.8996	0.3681	0.9460	0.1035	0.9222	0.1615	0.8483	0.1421	0.8493	0.1736	4.332
6	gbc	0.8194	0.8202	0.9025	0.7514	0.8278	0.5061	0.8142	0.0408	0.8209	0.0755	0.6388	0.0500	0.6389	0.1030	82.469
7	ada	0.7582	0.7349	0.8435	0.7101	0.7831	0.5475	0.7459	0.0298	0.7641	0.0565	0.5164	0.0298	0.5170	0.0771	19.638
8	ridge	0.6707	0.6443	0.0000	0.6329	0.7146	0.6212	0.6570	0.0251	0.6846	0.0482	0.3415	0.0209	0.3428	0.0663	0.411
9	lda	0.6707	0.6443	0.7260	0.6905	0.7146	0.6212	0.6570	0.0251	0.6846	0.0482	0.3415	0.0209	0.3428	0.0663	1.410

Table 10. Hybrid Gemma - Based Classification Comparison

Hybrid LLaMA 3.2-Based Supervised Classifiers Result

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	rf	0.9931	0.9869	0.9975	0.8239	0.9895	0.1329	0.9966	0.8025	0.9931	0.2281	0.9862	0.2249	0.9862	0.3234	48.801
1	et	0.9931	0.9868	0.9975	0.8162	0.9898	0.1820	0.9964	0.6704	0.9931	0.2863	0.9862	0.2818	0.9862	0.3449	42.960
2	dt	0.9829	0.9696	0.9829	0.5937	0.9862	0.2065	0.9797	0.1371	0.9829	0.1648	0.9657	0.1500	0.9658	0.1533	3.867
3	knn	0.9602	0.9470	0.9823	0.6087	0.9857	0.1933	0.9379	0.0636	0.9612	0.0957	0.9204	0.0755	0.9216	0.0882	98.162
4	xgboost	0.9574	0.9640	0.9909	0.8026	0.9459	0.3257	0.9682	0.1525	0.9569	0.2078	0.9148	0.1918	0.9151	0.2064	2.441
5	lightgbm	0.9215	0.9435	0.9770	0.7966	0.8938	0.3349	0.9463	0.0939	0.9193	0.1467	0.8431	0.1270	0.8444	0.1551	4.361
6	gbc	0.8200	0.8191	0.9051	0.7530	0.8231	0.5020	0.8180	0.0403	0.8206	0.0745	0.6400	0.0490	0.6401	0.1012	83.216
7	ada	0.7588	0.7339	0.8469	0.7119	0.7898	0.5593	0.7436	0.0303	0.7660	0.0575	0.5175	0.0308	0.5185	0.0799	19.711
8	ridge	0.6726	0.6408	0.0000	0.6352	0.7211	0.6293	0.6573	0.0252	0.6877	0.0484	0.3451	0.0211	0.3468	0.0672	0.422
9	lda	0.6726	0.6408	0.7286	0.6948	0.7211	0.6293	0.6573	0.0252	0.6877	0.0484	0.3451	0.0211	0.3468	0.0672	1.383

Table 11. Hybrid LLaMA 3.2-Based Classification Comparison

Hybrid Bert-Based Supervised Classifiers Result

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	rf	0.9934	0.9870	0.9975	0.8278	0.9895	0.1380	0.9972	0.8133	0.9934	0.2360	0.9868	0.2328	0.9868	0.3319	45.581
1	et	0.9934	0.9869	0.9976	0.8157	0.9897	0.1840	0.9970	0.6754	0.9933	0.2893	0.9867	0.2848	0.9868	0.3482	41.562
2	dt	0.9832	0.9727	0.9832	0.5945	0.9867	0.2050	0.9799	0.1589	0.9833	0.1790	0.9664	0.1654	0.9665	0.1668	3.756
3	xgboost	0.9597	0.9671	0.9920	0.8070	0.9464	0.3149	0.9723	0.1659	0.9592	0.2174	0.9195	0.2022	0.9198	0.2131	2.441
4	knn	0.9596	0.9479	0.9807	0.5881	0.9849	0.1600	0.9375	0.0550	0.9606	0.0818	0.9192	0.0616	0.9204	0.0711	99.378
5	lightgbm	0.9260	0.9494	0.9789	0.8029	0.8993	0.3354	0.9500	0.1061	0.9240	0.1612	0.8520	0.1423	0.8532	0.1680	4.153
6	gbc	0.8192	0.8237	0.9052	0.7547	0.8148	0.4888	0.8220	0.0403	0.8184	0.0744	0.6384	0.0489	0.6385	0.0996	82.106
7	ada	0.7582	0.7621	0.8458	0.7002	0.7675	0.4872	0.7536	0.0297	0.7604	0.0561	0.5164	0.0295	0.5166	0.0712	19.400
8	ridge	0.6747	0.6518	0.0000	0.6342	0.7064	0.6161	0.6643	0.0254	0.6847	0.0488	0.3495	0.0216	0.3502	0.0672	0.439
9	lda	0.6747	0.6518	0.7351	0.6964	0.7064	0.6161	0.6643	0.0254	0.6847	0.0488	0.3495	0.0216	0.3502	0.0672	1.386

Table 12. Hybrid Bert-Based Classification Comparison

Based on the results from the classification models with Borderline SMOTE oversampling, several key observations emerge:

Extra Trees Model: The Extra Trees (ET) model consistently achieves the highest F1 score across all models on the test data, demonstrating an excellent balance between precision and recall. This suggests it effectively manages both false positives and false negatives.

XGBoost Model: The XGBoost model exhibits notable improvements across multiple metrics, including F1 score, accuracy, AUC, Kappa, and MCC. The significant enhancement in precision suggests a reduction in false positives, making this model particularly reliable for classification tasks where minimizing false positives is essential.

Best Hybrid Model: Among the three hybrid models evaluated, the performance metrics display similar patterns and trends across the board. Notably, the Gemma variant shows relatively higher performance across models, indicating that it may offer a superior approach in leveraging hybrid modeling with categorical embeddings.

4.2.2.2 OpenAI Categorical Embedding

OpenAI has new small, reduced cost and highly efficient embedding model text-embedding-3-small that make it easy for machine learning models to understand the text content with the sequence of numbers (OpenAI, New Embedding Models and API Updates, 2024). Using the OpenAI API, categorical text variables can be converted into numerical vectors, making them ready for input into classification models. This approach allows for comparison with embeddings generated by locally deployed large language models (LLMs), such as those from OLLaMA.

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	rf	0.9934	0.9872	0.9975	0.8320	0.9895	0.1457	0.9972	0.8285	0.9934	0.2478	0.9868	0.2445	0.9868	0.3443	46.163
1	et	0.9934	0.9869	0.9976	0.8190	0.9897	0.1881	0.9970	0.6740	0.9934	0.2942	0.9868	0.2897	0.9868	0.3517	43.648
2	dt	0.9840	0.9720	0.9840	0.5972	0.9872	0.2111	0.9809	0.1563	0.9840	0.1796	0.9680	0.1657	0.9680	0.1676	3.925
3	knn	0.9594	0.9478	0.9805	0.5874	0.9847	0.1600	0.9372	0.0548	0.9604	0.0816	0.9188	0.0613	0.9199	0.0709	119.454
4	xgboost	0.9589	0.9672	0.9915	0.8033	0.9461	0.3103	0.9711	0.1651	0.9584	0.2155	0.9179	0.2004	0.9182	0.2109	2.557
5	lightgbm	0.9267	0.9496	0.9790	0.7991	0.8992	0.3354	0.9516	0.1066	0.9246	0.1618	0.8534	0.1429	0.8547	0.1684	4.532
6	gbc	0.8201	0.8244	0.9040	0.7518	0.8199	0.4698	0.8202	0.0390	0.8200	0.0720	0.6402	0.0465	0.6402	0.0943	89.917
7	ada	0.7682	0.7680	0.8526	0.6999	0.7710	0.4632	0.7667	0.0291	0.7688	0.0547	0.5364	0.0282	0.5364	0.0669	20.732
8	ridge	0.6760	0.6578	0.0000	0.6362	0.7104	0.6140	0.6647	0.0258	0.6868	0.0495	0.3520	0.0223	0.3528	0.0685	0.445
9	lda	0.6760	0.6578	0.7297	0.6942	0.7105	0.6140	0.6647	0.0258	0.6868	0.0495	0.3520	0.0223	0.3528	0.0685	1.521

Table 13. OpenAI Embedding Classification Comparison

4.2.2.3 Gemini Categorical Embedding

Similar to OpenAI's embedding model, the Gemini API utilizes the embedContent method to convert text columns into representative numerical vectors, making it well-suited for a range of natural language processing tasks, such as text classification and semantic search, and more (Google, 2024). These embeddings can also be used to compare model performance against embeddings generated by locally deployed LLMs and OpenAI's embedding model, providing insights into the relative effectiveness of each

approach.

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	rf	0.9934	0.9869	0.9975	0.8293	0.9895	0.1242	0.9973	0.8020	0.9934	0.2151	0.9868	0.2121	0.9868	0.3126	47.486
1	et	0.9933	0.9868	0.9975	0.8183	0.9897	0.1794	0.9970	0.6686	0.9933	0.2830	0.9867	0.2785	0.9867	0.3420	42.366
2	dt	0.9836	0.9733	0.9836	0.5930	0.9871	0.2014	0.9803	0.1617	0.9837	0.1794	0.9672	0.1660	0.9672	0.1670	4.209
3	knn	0.9594	0.9478	0.9805	0.5874	0.9847	0.1600	0.9372	0.0548	0.9604	0.0816	0.9188	0.0613	0.9199	0.0709	97.021
4	xgboost	0.9589	0.9636	0.9917	0.8060	0.9464	0.3277	0.9707	0.1513	0.9584	0.2070	0.9178	0.1910	0.9181	0.2061	2.492
5	lightgbm	0.9261	0.9457	0.9789	0.7995	0.8977	0.3497	0.9517	0.1015	0.9239	0.1574	0.8521	0.1380	0.8535	0.1669	4.556
6	gbc	0.8267	0.8357	0.9094	0.7505	0.8178	0.4504	0.8326	0.0401	0.8252	0.0737	0.6534	0.0483	0.6535	0.0945	85.059
7	ada	0.7617	0.7537	0.8510	0.7011	0.7763	0.5082	0.7543	0.0299	0.7651	0.0565	0.5234	0.0299	0.5237	0.0736	20.532
8	ridge	0.6771	0.6602	0.0000	0.6334	0.7093	0.6058	0.6663	0.0256	0.6871	0.0492	0.3541	0.0220	0.3548	0.0672	0.434
9	lda	0.6770	0.6602	0.7348	0.6914	0.7093	0.6058	0.6663	0.0256	0.6871	0.0492	0.3541	0.0220	0.3548	0.0672	1.456

Table 14. Gemini Embedding Classification Comparison

4.2.2.5 Hybrid Model Result Comparison

This study assesses the performance of five LLM-based embedding techniques on categorical variables by comparing their respective performance metrics. Among these, the hybrid model combining OpenAI embeddings with an Extra Trees classifier achieved the highest F1 score of 0.2942 on the test data, emerging as the best-performing combination. OpenAI's embedding algorithm also notably increased the F1 score of the XGBoost classifier from 0.128 to 0.2155. For recall and precision, XGBoost recorded a recall of 0.3103 and precision of 0.1651, while the Extra Trees classifier had a lower recall of 0.1881 but a significantly higher precision of 0.6740. Depending on whether the user prioritizes reducing false positives or false negatives, the model selection can be tailored accordingly. Section 4.4 explores a novel methodology to balance false positives and false negatives to optimize profit. In the upcoming Section 4.3, this research examines the impact of synthetic data oversampling methods to assess potential improvements in model performance.

4.3 LLM-Based Synthetic Data Generation

The generation of synthetic data and associated analytics supports the following research hypothesis:

H2: Integrating synthetic data generated by the fine-tuned GPT-4o Mini model for the minority class, followed by oversampling, will enhance the performance of the binary classification model.

4.3.1 Synthetic Data

To generate synthetic data, the GPT-4o Mini model is fine-tuned on training data prior to November 2016 to capture historical fraud patterns effectively. Once the model has learned these patterns, prompt-based techniques can be applied to generate synthetic data efficiently. Previous sections identified that borderline SMOTE most improves model performance, likely because the additional oversampled points for the minority class are located near challenging-to-classify boundary areas. Building on this insight, if the LLM can effectively learn fraud patterns and generate synthetic data that is sufficiently distanced from the classification boundary, it could further enhance model performance.

4.3.2 Fine-Tuned GPT-4o Mini Model

To fine-tune the GPT-4o Mini model, the hyperparameters were configured with 3 epochs, a batch size of 3, and a learning rate multiplier of 0.3. The goal was not to exhaustively test hyperparameters but to achieve a balanced, generalized fine-tuning. The fine-tuned model results were stored on the OpenAI Platform for future reuse.

4.3.3 Few-shot Prompt

This study leverages OpenAI and LangChain to implement few-shot prompting techniques for generating synthetic data. To ensure consistency, several validation steps and constraints were applied to create a final table that aligns with the required format. Key considerations are outlined below:

Generate Samples: Transform tabular data into a text-based format for prompt input.

```
[{'example': 'merchantName: Uber, accountNumber: 737265056, transactionAmount: 98.55, cardLast4Digits: 1803, merchantCategoryCo
de: rideshare, DAYsinceOpen: 518, posEntryMode: 1, availableMoney: 5000.0, isFraud: False'},
 {'example': 'merchantName: AMC #191138, accountNumber: 737265056, transactionAmount: 74.51, cardLast4Digits: 767, merchantCate
goryCode: entertainment, DAYsinceOpen: 577, posEntryMode: 3, availableMoney: 5000.0, isFraud: False'}]
```

Rule Description: Ensure inclusion of all unique values for categorical variables and establish formatting constraint.

```
rules_description = """
You are tasked with generating synthetic data for a financial transaction dataset, particularly focusing on fraudulent
transactions during the holiday season (November to December).
The dataset will include the following variables:

1. **merchantName** (string): Name of the merchant. Must be chosen from the provided list of categories: {mer_cat}.
2. **accountNumber** (int): A unique identifier for the account. Format: 10 digits number (e.g., '7372650567').
3. **transactionAmount** (float): The amount of the transaction. Range: 0 to 2,500.
4. **cardLast4Digits** (int): The last four digits of the card number. Range: 0 to 9999
5. **merchantCategoryCode** (string): Category of the merchant. Must be chosen from the provided list of categories: {mcc_cat}
6. **DAYsinceOpen** (float): Number of days since the card was opened.
7. **posEntryMode** (int): Method of card entry. Must be chosen from the provided list of categories: {PEM_cat}.
8. **availableMoney** (float): The amount of money available for use. Range: -1,500 to 50,000.
9. **isFraud** (boolean): Indicates whether this transaction is suspected to be fraudulent.

Examples of transactions are as {examples}.

Search existed knowledge about credit card fraudulent behavior related to the variables above and generate 20 synthetic
transaction records that is fraudu transaction. List below:
"""
```

Table 15. Few-Slot Prompt

Validation: Use the Pydantic library to enforce format constraints for each variable. Any generated data that does not comply with these constraints is discarded.

```python
from pydantic import BaseModel, Field
class frauddata(BaseModel):
    accountNumber:int
    availableMoney:float
    transactionAmount:float
    merchantName:str
    posEntryMode:int
    merchantCategoryCode:str
    cardLast4Digits:int
    DAYsinceOpen:int
    isFraud: bool

    class Config:
        arbitrary_types_allowed = True
```

Final Result: Each prompt generation produces 20 rows, iterated 50 times. Due to the inability to set a fixed seed with the OpenAI API, the synthetic data cannot be reproduced with 100% consistency.

	accountNumber	availableMoney	transactionAmount	merchantName	posEntryMode	merchantCategoryCode	cardLast4Digits	DAYsinceOpen	isFraud
0	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
1	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
2	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
3	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
4	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
5	1234567890	-200.75	95.50	walmart.com	1	online_retail	6789	300	True
6	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
7	1234567890	-500.75	112.34	walmart.com	1	online_retail	6789	150	True
8	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True
9	1234567890	-300.50	150.75	oldnavy.com	1	online_retail	1234	200	True

Table 16. Synthetic Data Generation

4.3.4 Classifier Result Comparison

In this section, the study evaluates whether hybrid oversampling with synthetic data generation can improve model performance. First, a few-shot prompt fraud case is combined with the original dataset. Then, borderline SMOTE oversampling is applied to the combined data. Two experiments are conducted as follows:

1. The first experiment applies hybrid oversampling to the data using various machine learning models. The performance metrics for each model are then evaluated.

2. The second experiment uses hybrid oversampling technique, enhanced by LLM-based categorical variables, to determine whether this approach further improves performance metrics over the initial best model results.

	Model	Original Accuracy	New Accuracy	Original AUC	New AUC	Original Recall	New Recall	Original Prec.	New Prec.	Original F1	New F1	Original Kappa	New Kappa	Original MCC	New MCC	Training Time
0	rf	0.9935	0.9872	0.9974	0.8258	0.9888	0.1472	0.9981	0.8395	0.9934	0.2505	0.9870	0.2473	0.9870	0.3485	45.356
1	et	0.9934	0.9870	0.9975	0.8193	0.9890	0.1799	0.9978	0.7117	0.9934	0.2872	0.9868	0.2830	0.9869	0.3537	43.846
2	dt	0.9845	0.9729	0.9845	0.5998	0.9873	0.2156	0.9818	0.1662	0.9846	0.1877	0.9690	0.1742	0.9690	0.1757	3.768
3	xgboost	0.9671	0.9716	0.9934	0.8008	0.9526	0.2970	0.9810	0.1912	0.9666	0.2326	0.9342	0.2188	0.9346	0.2244	2.645
4	knn	0.9665	0.9451	0.9846	0.5698	0.9937	0.1256	0.9424	0.0414	0.9673	0.0623	0.9329	0.0414	0.9343	0.0483	97.596
5	lightgbm	0.9439	0.9643	0.9851	0.7976	0.9156	0.2879	0.9705	0.1414	0.9422	0.1897	0.8878	0.1736	0.8892	0.1852	4.187
6	gbc	0.8444	0.8549	0.9236	0.7461	0.8306	0.3985	0.8541	0.0407	0.8422	0.0738	0.6888	0.0488	0.6890	0.0891	87.245
7	ada	0.7889	0.7634	0.8768	0.6983	0.7954	0.4744	0.7852	0.0292	0.7903	0.0550	0.5778	0.0284	0.5779	0.0682	20.988
8	ridge	0.6746	0.6590	0.0000	0.6378	0.7053	0.6161	0.6645	0.0260	0.6843	0.0498	0.3493	0.0226	0.3499	0.0694	0.411
9	lda	0.6746	0.6590	0.7329	0.6962	0.7054	0.6161	0.6645	0.0260	0.6843	0.0498	0.3493	0.0226	0.3500	0.0694	1.548

4.4 Optimizing Cost-Efficiency in Hyperparameter Tuning

In the previous experiment, a key question remained: how can we better balance false positives and false negatives? Metrics like F1-score, MCC, and Kappa were used to account for these errors with varying weights. However, to optimize for cost efficiency, this study introduces a novel metric that considers the specific costs associated with both false positives and false negatives. This cost-sensitive metric is applied to guide hyperparameter tuning for the model. While the final selected model may not yield the absolute best performance metrics, it offers a cost-effective solution that can be implemented swiftly, minimizing overall operational costs.

4.4.1 Customized Metric

As discussed in Section 3.7, traditional performance metrics such as AUC, accuracy, F1-score, MCC, and Kappa offer a balanced view by accounting for both false positives and false negatives. However, the model that performs best on these metrics may not be the most cost-efficient due to the different financial impacts of false positives versus false negatives. As noted in Section 3.8, if users can estimate the costs associated with each false positive and false negative, this information can be incorporated to create a customized metric. By defining this metric as the total loss from false positives and false negatives, an optimal, cost-effective balance can be achieved. The metric can be formalized as:

$$Total\ Loss\ \Omega_i = FP_i \times \alpha + FN_i \times \beta\ , \qquad (15)$$

where α and β represent the estimated costs of false positives and false negatives, respectively. Through iterative hyperparameter tuning, this final metric can be minimized

to enhance both performance and cost efficiency.

```
from pycaret.classification import add_metric
import numpy as np

# Define the cost of false positives and false negatives
alpha = 50  # Cost for each false positive
beta = 5   # Cost for each false negative

# Define a function for custom loss metric
def custom_loss(y_true, y_pred):
    # Calculate confusion matrix components
    fp = np.sum((y_pred == 1) & (y_true == 0))  # False Positives
    fn = np.sum((y_pred == 0) & (y_true == 1))  # False Negatives

    # Calculate the custom loss
    loss = (fp * alpha) + (fn * beta)
    return -loss  # PyCaret maximizes metrics, so we negate to minimize
```

Assume that the cost for each false positive (α) is set at 50, and the cost for each false negative (β) is set at 5. These values will be used as tuning parameters in the next section.

4.4.2 Bayesian Search Hyperparameter Tuning

There are three primary techniques for hyperparameter tuning: grid search, random search, and Bayesian optimization. According to Medium.com, grid search, while thorough, is computationally intensive and can be costly. Random search is more efficient, often finding good results with fewer evaluations. Bayesian optimization, meanwhile, requires fewer evaluations overall but has higher per-evaluation costs and may not always yield the optimal solution. This study opts for Bayesian search, which iteratively selects points by evaluating inputs and refining a surrogate model to predict the next best point. Bayesian optimization effectively balances exploration of unexplored regions with exploitation of high-reward areas, optimizing the search process (Linmarsirait, 2023).

Tuning the most performing algorithm extra trees classifier on the test data set using the customized metric of total loss of false positive and false negative as shown below. Both recall (from 0.1799 to 0.1799) and precision (from 0.7117 to 0.6667) are reduced. But F1-score was decreased from 0.2873 to 0.2833.

```
]: predictions = predict_model(tuned_model, data=new)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Custom Loss
0	Extra Trees Classifier	0.9868	0.8107	0.1799	0.6667	0.2833	0.2789	0.3419	-17110

```
: save_model(tuned_model, 'best_loss_minimizing_model')
```

```
Transformation Pipeline and Model Successfully Saved
```

```
: (Pipeline(memory=Memory(location=None),
            steps=[('numerical_imputer',
                    TransformerWrapper(exclude=None,
                                       include=['accountNumber',
                                                'transactionAmount',
                                                'cardLast4Digits', 'DAYsinceOpen',
                                                'posEntryMode', 'availableMoney',
                                                'embed_merchantCategoryCode_0',
                                                'embed_merchantCategoryCode_1',
                                                'embed_merchantCategoryCode_2',
                                                'embed_merchantCategoryCode_3',
                                                'embed_merchantCat...
                    ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0,
                                         class_weight=None, criterion='gini',
                                         max_depth=None, max_features='sqrt',
                                         max_leaf_nodes=None, max_samples=None,
                                         min_impurity_decrease=0.0,
                                         min_samples_leaf=1, min_samples_split=2,
                                         min_weight_fraction_leaf=0.0,
                                         monotonic_cst=None, n_estimators=100,
                                         n_jobs=-1, oob_score=False,
                                         random_state=123, verbose=0,
                                         warm_start=False))],
            verbose=False),
 'best_loss_minimizing_model.pkl')
```

4.4.3 Final Model Selection

After hyperparameter tuning based on the new cost-sensitive metric, the table below presents the final performance of the best model identified in previous sessions,

using assumed values for α and β. Both false positives and false negatives have been reduced, with only a minor decrease in F1-score.

In terms of cost efficiency, it is essential to account for the number of false positives and false negatives to accurately estimate potential operational costs. This new metric facilitates automated hyperparameter tuning to minimize these costs. While the selected model may not achieve the highest overall performance, it is optimized to reduce costs associated with fraud detection, leading to a more cost-effective and practical solution.

.

Chapter 5—Discussion and Conclusions

5.1 Discussion

This research initially explored traditional classifiers and oversampling techniques for fraud detection to find out the best traditional model. Then applying examination on two novels methods aiming to address the core question of how to further enhance model performance by two novel methodologies. The study tested five Large Language Models (LLMs)—LLaMA 3.2, Gemma, BERT, the OpenAI embedding model, and the Gemini model—to convert text-based categorical variables into numerical vectors, creating a more meaningful representation of the original data. These transformed vectors, combined with other numerical features, were used to train machine learning models that outperformed traditional methods, increasing the F1 score from 0.2718 to 0.2942.

An additional finding emerged from comparing multiple transaction classification methods and oversampling approaches without further fine-tuning. The Extra Trees model consistently performed best across different oversampling methods. Furthermore, when LLM-based embeddings were incorporated, the performance of the XGBoost model showed notable improvement.

The second experiment introduced novel methods to improve financial fraud detection rates by addressing data imbalance. Synthetic data generation was implemented using a fine-tuned Generative Pretrained Transformer (GPT4o-Mini), producing approximately 10% additional data relative to the minority class in the training set. Due to cost considerations, this study recommends using Borderline SMOTE to further enhance data balance. Results from testing traditional techniques showed that Borderline

SMOTE outperformed standard SMOTE, likely because it places additional data near decision boundaries, where false predictions are more likely to occur. The hybrid approach—combining synthetic data generation with Borderline SMOTE—yielded improved results across all tested machine learning classifiers, elevating the highest F1 score from 0.2718 to a new benchmark.

One key finding here is due to cost issue by using OpenAI tool, the LLM-based oversampling could not generate all samples using few-slot prompting to complete the oversampling test. The improvement of model performance is not as significant as using LLM-based embedding method.

The final experiment centered on developing a custom metric, **Ω**, aimed at achieving an optimal balance between false positives and false negatives with a focus on cost efficiency. This balance was realized by using **Ω** as an innovative optimization metric, paired with Bayesian optimization to fine-tune the model's hyperparameters. Although the resulting model may not achieve the highest scores on traditional performance metrics, it effectively navigates the trade-off between false positives and false negatives, significantly reducing costs associated with both types of errors and enhancing the model's practical value in cost-sensitive fraud detection. This metric offers an additional recommendation for users to consider, prioritizing practical application of the final model to minimize potential operational and fraud-related costs.

5.2 Conclusions

- Combining synthetic data with borderline SMOTE enhances the performance of the original model, addressing data imbalance issues effectively.
- The OpenAI embedding model outperforms LLaMA, Gemini, Gemma, and BERT, significantly boosting machine learning model performance.

- Embedding-based hybrid models from OpenAI, LLaMA, Gemini, Gemma, and BERT provide a more meaningful representation of categorical variables, contributing to improved model performance overall.
- The model with the highest performance may not always be the most practical choice. The novel metric, Ω, aids in tuning hyperparameters to balance false positives and false negatives, minimizing the costs associated with incorrect predictions.

5.3 Contributions to Body of Knowledge

This Praxis contributes to the body of knowledge by advancing model development, performing statistical analysis on model performance, tuning machine learning hyperparameters, and leveraging large language models to better represent text-based categorical variables, thus improving overall model accuracy. The study extends the application of large language models to enhance tabular data generation—a complex task given that tabular data has different format requirements compared to generating textual responses. In this context, careful prompt setup, example provision, and validation of generated data are crucial to ensuring relevance and usability.

In financial fraud detection, the goal extends beyond building the highest-performing model; practical implications must also be considered. High false positives, for instance, increase customer complaints and risk customer attrition, while also consuming substantial time and effort from operational agents. On the other hand, false negatives occur when the model lacks sensitivity to fraud, allowing unauthorized transactions to go undetected, potentially leading to customer disputes and financial losses for the company. Achieving a balance between false positives and false negatives is essential, and this requires minimizing associated costs.

To address this, a novel metric, Ω, was introduced to optimize the trade-off between false positive and false negative rates. By incorporating Ω into the hyperparameter tuning process, the study aims to reduce overall costs linked to fraud detection errors, creating a more cost-effective and practically viable model.

5.4 Recommendations for Future Research

Large language models (LLMs) are a relatively new and rapidly evolving technology with extensive potential for application across various industries. These models can be leveraged to summarize and generate text, recognize patterns, and potentially support real-time decision-making. Future research opportunities include:

- **Fine-tuning LLMs for Specific Business Needs**: Companies can continue training LLMs to enhance their understanding of specific business problems, enabling them to automate tasks and reduce manual labor. For instance, LLMs could be fine-tuned to power customer service bots that resolve real-time inquiries autonomously. Future studies could focus on improving these bots' decision accuracy and enabling real-time self-correction for enhanced customer satisfaction and efficiency.
- **Addressing Data Sensitivity in Finance and Healthcare**: Access to sensitive data is often restricted in industries like finance and healthcare, which limits the data available for analytics and can delay fraud detection. A potential area of research is the use of LLMs to detect suspicious patterns across multiple organizations, with anonymized, shared insights to publicly highlight new patterns. Additionally, few-shot or zero-shot data generation could be explored to

simulate unseen fraud scenarios, helping prevent fraud attacks that have not been previously encountered but may emerge in the future.

- **Enhancing Compliance and Governance Monitoring:** LLMs offer potential for monitoring and validating the performance and accuracy of traditional models, especially in compliance and internal auditing. Many manual processes remain in these areas, and while automation has been established for certain reports, new reports are often required with each regulatory change. Research could explore how LLMs might automate these tasks and quickly adapt to evolving compliance needs.
- **Developing a Centralized "Data Brain":** Companies invest heavily in architecting, maintaining, and operationalizing data across the organization. Outdated mappings and undetected bugs can limit data effectiveness. Leveraging LLMs to centralize, synthesize, and update enterprise data in real time could greatly improve efficiency, reduce costs, and enhance decision-making.
- **Advancing Algorithmic Performance:** Investigating the synergy between LLMs and advanced algorithms could be a valuable research direction. LLMs may offer ways to address complex deep learning problems or improve the efficiency of classification and pattern recognition algorithms. Studying how LLMs can be integrated to enhance traditional and deep learning models could unlock new possibilities in machine learning.

References

abdullahsaka. (2021). *Fraud Detection Case Study*. Retrieved from Github: https://github.com/abdullahsaka/Capital_One-Data_Challenge

Ahammad, J., Hossain, N., & Shafiul Ala, M. (2020). Credit Card Fraud Detection using Data Pre-processing on Imbalanced Data - both Oversampling and Undersampling. *International Conference on Computing Advancements.* ACM Digital Library.

Ahmadi, S. (2023). Open AI and its Impact on Fraud Detection in Financial Industry. *Journal of Knowledge Learning and Science Technology*.

Ahmed, N. A. (2023, 11 30). *What is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained*. Retrieved from Datacamp: https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning

AI Google. (2024, 6 27). *Gemma Models Overview*. Retrieved from https://ai.google.dev/gemma/docs

Aidasani, C. (2023, 10 21). *The Art of Prompt Engineering- Part 2*. Retrieved from Browsee: https://browsee.io/blog/the-art-of-prompt-engineering-part-2/

Akshara. (2024, 9 10). *Anomaly Detection using Isolation Forest - A Complete Guide*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/

Alothman, R., Talib, H. A., & Mohammed, M. S. (2022). Fraud detection under the unbalanced class based on gradient boosting. *Eastern European Journal of Enterprise Technologies*.

An, H., Ma, R., Yan, Y., Chen, T., Zhao, Y., Fan, D., & Lv, C. (2024). Finsformer: A Novel Approach to Detecting Financial Attacks Using Transformer and Cluster-Attention. *MDPI*.

Bakumenko, A., Hlaváčková-Schindler , K., Plant , C., & Hubig, N. C. (2024). Advancing Anomaly Detection: Non-Semantic Financial Data Encoding with LLMs. *AeXiv*.

Bhandari, A. (2024, 10 11). *Guide to AUC ROC Curve in Machine Learning*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

BL, P. (2024, 4 29). *Understanding Feed Forward Networks in Transformers*. Retrieved from medium.com: https://medium.com/@punya8147_26846/understanding-feed-forward-networks-in-transformers-77f4c1095c67

Boulsane, H. A., & Afdel, K. (2024). Can Machine Learning Outperform Deep Learning in Financial Fraud Detection?: A Comprehensive Look at Improving Fraud Prevention Strategies. *Elsevier*.

Browne, R. (2024, 2 1). *Mastercard jumps into generative AI race with model it says can boost fraud detection by up to 300%*. Retrieved from CNBC: https://www.cnbc.com/2024/02/01/mastercard-launches-gpt-like-ai-model-to-help-banks-detect-fraud.html

Bussler, F. (2023, 9 20). *Guide: Large Language Models-Generated Fraud, Malware, and Vulnerabilities*. Retrieved from Fingerprint: https://fingerprint.com/blog/large-language-models-llm-fraud-malware-guide/#:~:text=discourse%20and%20transparency.-,Financial%20Fraud%20and%20Identity%20Theft,%2C%20letters%2C%20and%20phone%20transcripts.

calvinscottforbes. (2021). *capital-one-fraud-detection*. Retrieved from Github: https://github.com/mandyshen/capital-one-fraud-detection

CapitalOneRecruiting. (2018). *Capital One Data Science Challenge*. Retrieved from Github: https://github.com/CapitalOneRecruiting/DS

Chase Bank. (2021, 05 10). Retrieved from J.P.Morgan.com: https://www.jpmorgan.com/insights/payments/fraud-and-risk-management/payments-fraud-and-cybersecurity-trends-to-watch

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*.

Chen, J., Zhu, G., Yuan, C., & Huang, Y. (2024). Boosting Graph Anomaly Detection with Adaptive Message Passing. *International Conference on Learning Representations (ICLR).* Vienna, Austria.

Cheng, Y., Guo , J., Long, S., Wu, Y., Sun, M., & Zhang, R. (2024). Advanced Financial Fraud Detection Using GNN-CL Model. *arXiv*.

Cohen's Kappa. (n.d.). Retrieved from Datatab: https://datatab.net/tutorial/cohens-kappa

Cui, J., Long, J., Min, E., Liu, Q., & Li, Q. (2018). Comparative Study of CNN and RNN for Deep Learning Based Intrusion Detection System. *International Conference on Cloud Computing and Security* (pp. 159-170). Springer Nature.

Datacamp. (2024, 1 19). *Fine-Tuning GPT-3 Using the OpenAI API and Python*. Retrieved from datacamp.com: https://www.datacamp.com/tutorial/fine-tuning-gpt-3-using-the-open-ai-api-and-python

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*.

Dornadula, V. N., & Geetha, S. (2019). Credit Card Fraud Detection using Machine Learning Algorithms,. *Procedia Computer Science*.

Drenik, G. (2023, 10 11). *Generative AI Is Democratizing Fraud. What Can Companies And Their Consumers Do To Prevent Being Scammed?* Retrieved from Forbes: https://www.forbes.com/sites/garydrenik/2023/10/11/generative-ai-is-democratizing-fraud-what-can-companies-and-their-consumers-do-to-prevent-being-scammed/?sh=7b22d22d13e7

El-Naby, A. A., Hemdan, E. E.-D., & El-Sayed, A. (2022). An efficient fraud detection framework with credit card imbalanced data in financial services. *Springer*.

Federal Trade Commission. (2024, 2 9). *As Nationwide Fraud Losses Top $10 Billion in 2023, FTC Steps Up Efforts to Protect the Public*. Retrieved from Federal Trade Commission: https://www.ftc.gov/news-events/news/press-releases/2024/02/nationwide-fraud-losses-top-10-billion-2023-ftc-steps-efforts-protect-public

Forbes Technology Council. (2024, 03 07). *Successful Real-World Use Cases For LLMs (And Lessons They Teach)*. Retrieved from Forbes: https://www.forbes.com/sites/forbestechcouncil/2024/03/07/successful-real-world-use-cases-for-llms-and-lessons-they-teach/

Google. (2024, 8 8). *Embeddings in the Gemini API*. Retrieved from Googal AI for Developers: https://ai.google.dev/gemini-api/docs/embeddings

Google Gemini Team . (2023). Gemini: A Family of Highly Capable Multimodal Models. *Arvix*.

Google Gemma Team. (2024). Gemma: Open Models Based on Gemini Research and Technology. *ArXiv*.

Google Research. (2018, 11 2). Retrieved from Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing: https://research.google/blog/open-sourcing-bert-state-of-the-art-pre-training-for-natural-language-processing/

Gosain, A., & Sardana, S. (2017). Handling Class Imbalance Problem using Oversampling Techniques: A Review. *IEEE*.

Hanif, M. S. (2024, 5 29). *Synthetic Identities: The Darker Side Of Generative AI*. Retrieved from Forbes: https://www.forbes.com/sites/forbestechcouncil/2024/05/29/synthetic-identities-the-darker-side-of-generative-ai/

Hao An, R. M. (2024). Finsformer: A Novel Approach to Detecting Financial Attacks. *MDPI*.

Huesmann, K., & Linsen, L. (2024). Large Language Models for Transforming Categorical Data to Interpretable Feature Vectors. *IEEE Xplore*.

IBM. (2024, 6 19). *Foundation model parameters: decoding and stopping criteria*. Retrieved from IBM Watsonx: https://www.ibm.com/docs/en/watsonx/saas?topic=lab-model-parameters-prompting

Isangediok, M., & Gajamannage, K. (2022). Fraud Detection Using Optimized Machine Learning Tools Under Imbalance Classes . *2022 IEEE International Conference on Big Data.* IEEE Xplore.

Kalia, S. (2023). Potential Impact of Generative Artificial Intelligence on the Financial Industry. *Cybernetics & Informatics*.

Kennedy, R. K., Salekshahrezaee, Z., Villanustre, F., & Khoshgoftaar , T. M. (2023). Iterative cleaning and learning of big highly-imbalanced fraud data using unsupervised learning. *Springer*.

Kong, M., Li, R., Wang, J., Li, X., Jin, S., Xie, W., . . . Cao, C. (2024). CFTNet: a robust credit card fraud detection model enhanced by counterfactual data augmentation. *Springer Link*.

Lalchand, S., Srinivas, V., Maggiore, B., & Henderson, J. (2024, 5 29). *Generative AI is expected to magnify the risk of deepfakes and other fraud in banking*. Retrieved from Deloitte Center for Financial Services: https://www2.deloitte.com/us/en/insights/industry/financial-services/financial-services-industry-predictions/2024/deepfake-banking-fraud-risk-on-the-rise.html

LangChain. (2024). *Build a Retrieval Augmented Generation (RAG) App*. Retrieved from LangChain: https://python.langchain.com/v0.2/docs/tutorials/rag/

Lee, J., Hu, S., Tecimer, K., & Luo, I. (2023, 12 1). *Fraudulent Card Transaction Classifier Based On Capital One Synthetic Data*. Retrieved from ubc-mds.github.io: https://ubc-mds.github.io/fraud_detection/fraud_detection_full.html

Lenard, M. J., & Alam, P. (2009). An Historical Perspective on Fraud Detection From Bankruptcy Models to Most Effective Indicators of Fraud in Recent Incidents. *Journal of Forensic & Investigative Accounting*.

Li, Z., Zhao, Y., Botta, N., Ionescu, C., & Hu, X. (2020). COPOD: Copula-Based Outlier Detection. *arViv*.

Linmarsirait. (2023, 8 11). *Hyperparameter Tuning in Machine Learning Using Bayesian Optimization*. Retrieved from Medium: https://medium.com/@linmarsirait2/hyperparameter-tuning-in-machine-learning-using-bayesian-optimization-8ee522ef6d99

Manav. (2024, 10 7). *Introduction to Convolutional Neural Networks (CNN)*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Marvin, G., Hellen, N., Jjingo, D., & Nakatumba-Nabende, J. (2024). Prompt Engineering in Large Language Models. *Springer Link*.

Meta. (2024, 7 23). *Introducing LLaMA 3.1: Our most capable models to date*. Retrieved from Meta: https://ai.meta.com/blog/meta-LLaMA-3-1/

Meta Platform Inc. (2024, 04 18). *Introducing Meta LLaMA 3: The most capable openly available LLM to date*. Retrieved from Meta: https://ai.meta.com/blog/meta-LLaMA-3/

Meta Platform, Inc. (2024, 07 23). Retrieved from Introducing LLaMA 3.1: Our most capable models to date: https://ai.meta.com/blog/meta-LLaMA-3-1/

Mienye, I. D., & Jere, N. (2024). Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions. *IEEE Xplore*.

Mirchandani, S., Xia, F., Florence, P., Ichter, B., Driess, D., Montserrat, A. G., . . . Zeng, A. (2023). Large Language Models as General Pattern Machines. *Arxiv*.

Mondal, I. A., Haque, M. E., Hassan, A.-M., & Shatabda, S. (2021). Handling Imbalanced Data for Credit Card Fraud Detection. *2021 24th International Conference on Computer and Information Technology*. IEEE Xplore.

OpenAI. (2023). GPT-4 Technical Report. *OpenAI*.

OpenAI. (2024, 8 20). *Fine-tuning now available for GPT-4o*. Retrieved from OpenAI.com: https://openai.com/index/gpt-4o-fine-tuning/

OpenAI. (2024, 7 18). *GPT-4o mini: advancing cost-efficient intelligence*. Retrieved from OpenAI: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

OpenAI. (2024, 05 13). *Hello GPT-4o*. Retrieved from Openai: https://openai.com/index/hello-gpt-4o/

OpenAI. (2024, 1 25). *New Embedding Models and API Updates*. Retrieved from OpenAI: https://openai.com/index/new-embedding-models-and-api-updates/

OpenAI. (2024). *Pricing*. Retrieved from OpenAI: https://openai.com/api/pricing/

OpenAI. (n.d.). *Fine-tuning*. Retrieved from OpenAI Platform: https://platform.openai.com/docs/guides/fine-tuning/fine-tuning-integrations

PerceptiveAnalytics. (2018, 08 22). *Feature Selection Techniques with R*. Retrieved from PerceptiveAnalytics: https://www.perceptive-analytics.com/feature-selection-techniques-r/

Persona. (2024, 3 13). *LLMs + fraud: How criminals use large language models to commit fraud*. Retrieved from Persona: https://withpersona.com/blog/llm-fraud#toc-0

Pichai, S., & Hassabis, D. (2023, 12 6). Retrieved from Introducing Gemini: our largest and most capable AI model: https://blog.google/technology/ai/google-gemini-ai/

Pradhan, D. (2024, 1 14). *The Best Performance Metric for Classification: Matthews Correlation Coefficient*. Retrieved from Medium:

https://medium.com/@pradhandebasish2046/the-best-performance-metric-for-classification-matthews-correlation-coefficient-15bdd77df17d

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. *OpenAI*.

Rendyk. (2024, 9 28). *Distinguish Between Tree-Based Machine Learning Models*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/04/distinguish-between-tree-based-machine-learning-algorithms/

Sachinsoni. (2023, 6 11). *K Nearest Neighbours - Introduction to Machine Learning Algorithms*. Retrieved from Medium.com: https://medium.com/@sachinsoni600517/k-nearest-neighbours-introduction-to-machine-learning-algorithms-9dbc9d9fb3b2

Sanusi, Z. M., Rameli, M. N., & Isa, Y. M. (2015). Fraud Schemes in the Banking Institutions: Prevention Measures to Avoid Severe Financial Loss. *Procedia Economics and Finance*.

Scikit Learn. (2024). *1.1 Linear Models*. Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/linear_model.html

Scikit Learn. (2024). *1.1.11 Logistic Regression*. Retrieved from Scikit-Learn: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Scikit Learn. (2024). *1.2. Linear and Quadratic Discriminant Analysis*. Retrieved from Scikit-Learn: https://scikit-learn.org/stable/modules/lda_qda.html#

Scikit Learn. (2024). *1.6. Nearest Neighbors*. Retrieved from Scikit-Learn: https://scikit-learn.org/stable/modules/neighbors.html

Scikit Learn. (2024). *PCA*. Retrieved from Scikit-Learn: https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.PCA.html

Shastri, Y. (2024, 4). *Attention Mechanism in LLMs: An Intuitive Explanation*. Retrieved from Datacamp.com: https://www.datacamp.com/blog/attention-mechanism-in-llms-intuition

Sulaiman, S. S., Nadher, I., & Hameed, S. M. (2023). Credit Card Fraud Detection Using Improved Deep LearningModels. *Computers, Materials & Continua*.

Talamadupula, K. (2024, 3 4). *A Guide to LLM Hyperparameters*. Retrieved from Symbl.ai: https://symbl.ai/developers/blog/a-guide-to-llm-hyperparameters/

Tian, Y., & Liu, G. (2024). Spatial-Temporal-Aware Graph Transformer for Transaction Fraud Detection. *IEEE*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., . . . Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *ArXiv*.

TransUnion. (2024). *TransUnion 2024 State of Omnichannel Fraud Report.* TransUnion.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *ArXiv*.

Velloso, M., & Woodward, J. (2024, 05 14). *Gemini 1.5 Pro updates, 1.5 Flash debut and 2 new Gemma models*. Retrieved from Google: https://blog.google/technology/developers/gemini-gemma-developer-updates-may-2024/

Vidivelli, S., Ramachandran, M., & Dharunbalaji, A. (2024). Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG,andPerformance-Optimized LLM Fusion. *Tech Science Press*.

Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random Forest for Credit Card Fraud Detection. *IEEE*.

Yu, C., Xu, Y., Cao, J., Zhang, Y., Jin, Y., & Zhu, M. (2024). Credit Card Fraud Detection Using Advanced Transformer Model. *ArXiv*.

Appendix A

Github: https://github.com/qqshannon88/Jiayi-Lin-Research