VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF ECONOMICS AND LAW
FACULTY OF INFORMATION SYSTEMS

------□□□------

FINAL PROJECT REPORT

MACHINE LEARNING IN BUSINESS ANALYTICS

APPLYING MACHINE LEARNING

CREDIT CARD FRAUD DETECTION

Lecturer: Tran Duy Thanh, Ph.D.

Group 1:

1	K224161845	-	Trương Thảo Vy
2	K224161820	-	La Nam Khánh
3	K224161808	-	Nguyễn Trọng Tấn Dũng
4	K224161838	-	Nguyễn Hoàng Bảo Trân
5	K224161817	-	Phạm Nguyễn Gia Huy

Ho Chi Minh City, march 2025

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF ECONOMICS AND LAW
FACULTY OF INFORMATION SYSTEMS

------□□□------

FINAL PROJECT REPORT

MACHINE LEARNING IN BUSINESS ANALYTICS

APPLYING MACHINE LEARNING

CREDIT CARD FRAUD DETECTION

Nguyễn Trọng Tấn Dũng

Ho Chi Minh City, March 2025

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Mr. Tran Duy Thanh for his invaluable guidance and support throughout this project. His enthusiasm and expertise have greatly contributed to its success.

We would also like to extend our thanks to the teachers in the Faculty of Information Systems at the University of Economics and Law, VNU-HCM. Their dedication to imparting knowledge has been instrumental in shaping our understanding and skills during our study days.

Furthermore, we are grateful to the University of Economics and Law, VNU-HCM for including the subject "Machine Learning in Business Analytics" in the curriculum. The knowledge gained from this course has not only deepened our understanding of our field but has also equipped us with valuable tools for future studies and professional endeavors.

Lastly, we would like to wish everyone in the "Machine Learning in Business Analytics" class good health and continued success in their academic pursuits. May you all achieve great accomplishments in your learning journey.

Once again, we express our heartfelt appreciation. Thank you sincerely.

Group 1

COMMITMENT

We hereby declare that this project is the result of our research and research with the dedicated guidance of Mr. Tran Duy Thanh. This project has never been posted before. The reference content, we have consulted, and the content we refer to are all publicly available. If we detect any untrue content, we are ready to take all responsibility.

Ho Chi Minh City, March 2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

Acronyms	Meaning
eKYC	electronic Know Your Customer
ML	Machine Learning
IDE	Integrated Development Environment
APR	Annual Percentage Rate
LightGBM	Light Gradient-Boosting Machine
SMOTE	Synthetic Minority Over-sampling Technique
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
ROC AUC	Area Under the Receiver Operating Characteristic curve

ABSTRACT

Nowadays, credit cards have become an essential payment tool in modern life. With the rapid expansion of the financial sector and advancements in technology, billions of credit card transactions are processed worldwide every day. However, the increasing sophistication of fraudsters has made fraudulent transactions a major concern for financial institutions. Banks and payment providers must continuously enhance their fraud detection systems to minimize financial losses and protect customers.

In response to these challenges, artificial intelligence, particularly machine learning, has emerged as a powerful approach to detecting fraudulent credit card transactions. This project utilizes a credit card fraud detection dataset sourced from Kaggle, containing transaction details such as amount, time, and user behavior patterns. These results demonstrate the model's exceptional ability to identify fraudulent transactions, enhancing security and reducing financial risks for banks and customers

Keywords: Machine Learning, credit card fraud.

CHAPTER 1. OVERVIEW THE TOPIC

In the first chapter of the research paper, the group introduces the topic "Applying Machine Learning to Credit Card Fraud Detection." This chapter discusses the widespread use of credit cards in the global economy, spanning both developed and emerging markets. It highlights the growing challenge that financial institutions face in detecting fraudulent transactions, as fraudsters continuously develop more sophisticated techniques to bypass security measures.

The objective of this study is to develop a machine learning-based system that automatically detects fraudulent credit card transactions by analyzing user behavior and transaction patterns. This system aims to enhance the efficiency and accuracy of fraud detection, minimizing financial losses for banks and customers. Additionally, the chapter outlines the specific objectives, target subjects, and scope of the research. It describes the research methodology employed, including data collection, feature selection, and model evaluation. The practical significance of the study is also discussed, emphasizing its potential impact on reducing fraud-related losses and improving security in financial transactions. Finally, the chapter presents the structure of the research paper, summarizing the main sections and the expected content of each chapter.

1.1. Rationale for topic selection

Credit card fraud has become a pervasive issue that significantly impacts individuals and organizations worldwide. The rise of e-commerce and digital payments has contributed to an alarming increase in fraudulent activities, creating substantial financial risks and eroding consumer trust. According to the Nilson Group, global losses from payment card fraud exceeded $28 billion in 2022 and are projected to surpass $35 billion by 2025 (Nilson Report, 2023). This rapid escalation underscores the critical need for innovative and effective fraud detection mechanisms.

In Vietnam, the financial landscape mirrors these global trends. By July 2023, the country recorded over 140 million payment cards in circulation, including 36.7 million

international cards (State Bank of Vietnam, 2023). Despite advancements in security protocols, financial institutions face increasing challenges in detecting and preventing fraud. Traditional methods of analyzing fraud patterns remain labor-intensive, error-prone, and inefficient, leaving organizations vulnerable to ever-evolving fraud tactics (Nguyen et al., 2022).

The advent of artificial intelligence, particularly machine learning, has revolutionized fraud detection. Machine learning models offer unparalleled capabilities to analyze vast amounts of transactional data, identify anomalies, and predict fraudulent activities in real time (Kibria & Sevkli, 2021). These systems improve the precision and speed of fraud detection processes, minimizing financial losses and bolstering consumer confidence.

This research project aims to develop a machine learning-based fraud detection system. By leveraging transaction data and behavioral patterns, the system will provide an efficient and scalable solution to mitigate fraud risks. Such advancements align with global efforts to enhance financial security and streamline operational processes (Sangal, 2022).

1.2. Objectives of the Project

The objective of this project is to research, develop software, and integrate a machine learning model to detect credit card fraud. The specific goals include:

- Understanding and applying supporting software and databases in the project, specifically Qt Designer, MySQL, and PyCharm.
- Identifying and analyzing key indicators of fraudulent credit card transactions.
- Exploring the theoretical foundations of credit card fraud detection and applicable machine learning algorithms.
- Addressing data imbalance in the dataset to optimize classification and prevent overfitting during fraud detection.

- Developing a machine learning model to identify fraudulent transactions based on historical data and relevant factors, helping financial institutions enhance security and reduce financial losses.

1.3. Practical significance

The project holds significant value in the application of machine learning models for credit card fraud detection, offering critical advantages for both financial institutions and the broader financial market.

For banks and financial institutions, the project aids in the early identification of fraudulent activities, enabling them to minimize financial losses and reduce the risks associated with credit card fraud. By utilizing machine learning algorithms to analyze transaction patterns and detect anomalies, banks can make more accurate and timely decisions, preventing fraud before it occurs. This not only helps in safeguarding customer trust but also enhances operational efficiency by automating fraud detection processes, reducing the reliance on manual checks.

For the financial market, the project promotes greater security and confidence in digital transactions, which is crucial for the growth of e-commerce and online banking. By improving fraud detection mechanisms, the financial sector can ensure that consumers feel safe when using credit cards, leading to increased transaction volumes and overall market activity. Additionally, the integration of advanced machine learning models for fraud detection supports innovation in financial technologies, fostering a more robust and resilient financial ecosystem.

Overall, the project demonstrates the power of machine learning in transforming credit card fraud detection, offering tangible benefits to banks, consumers, and the financial market as a whole. By enhancing fraud prevention efforts, it helps protect assets, build trust, and contribute to a more stable and dynamic financial environment.

1.4. Subjects and Scope of the Project

(i) Subjects: A machine learning model for detecting credit card fraud. The dataset includes various features such as transaction amount, location, timestamp, user behavior patterns, and other relevant details to identify fraudulent activities.

(ii) Scope:

- Research time: 6 months
- Research based on a dataset retrieved on Kaggle.This dataset is merged, cleaned, and transformed using Pentaho Data Integration to ensure high-quality data for training the fraud detection model.

1.5. Methodology

The methods employed in this study include:

(i) Analysis and synthesis of theoretical frameworks: Research scientific literature and articles related to models, technologies and knowledge required to implement credit card fraud detection systems.

(ii) Information gathering method: Credit card transaction data will be collected from Kaggle, including important information such as user ID, transaction country, transaction amount, geographic location (coordinates or city name), transaction time, and successful or failed transaction code. This data will play a core role in analyzing user behavior and detecting unusual transactions.Statistical methods: Analyzing personal data, income, marital status, etc., using appropriate key variables for the research.

(iii) Statistical methods:Analyze user transaction data based on key factors such as spending history, usual transaction location, and transaction frequency.

(iv) Experimental method: The process of building a Machine Learning model to detect fraudulent transactions consists of several key steps. First, data preprocessing is performed by cleaning and extracting features such as average spending history, common transaction locations or countries, and transaction frequency over time. Next, the system analyzes user behavior by calculating

deviations from the average spending history and comparing the current transaction location with frequently used locations. Then, Machine Learning models such as Isolation Forest, Random Forest, or Logistic Regression are applied to detect fraudulent transactions, using a threshold of 20-30% deviation from normal behavior to identify suspicious transactions. Finally, the system implements response actions, including requesting user authentication or blocking the account after three failed attempts to ensure transaction security.

(v) Quantitative method: Applying historical transaction data to infer and implement Machine Learning models for real-time fraud detection. The system will utilize **Behavior and Location Analysis (BLA)** – an advanced method that identifies abnormal spending patterns without requiring predefined fraud signatures, minimizing false alerts and improving accuracy in fraud detection.

1.6. Structure of the project

With this project, the summary report is divided into specific sections as follows:

(i) Chapter 1: OVERVIEW THE TOPIC

The first chapter provides a broad overview of the research subject, detailing the usage and current state of credit card fraud detection. It explains the rationale behind selecting this topic, the research objectives, target audience, and scope. Additionally, it outlines the research methods, structure, and significance of studying credit card fraud detection.

(ii) Chapter 2: THEORETICAL FOUNDATIONS

Chapter 2 delves into the fundamental theories and methodologies used in the study. Specifically, it covers:

- An overview of theoretical foundations, including concepts of credit cards and essential factors for fraud detection.
- Insights into machine learning, the algorithms applied, data training models, data mining, and analysis techniques.

- An understanding of Python programming, with a focus on interface programming using PyQt6.

(iii) Chapter 3: PROPOSAL MODEL DEVELOPMENT, ANALYSIS, AND DESIGN OF SOFTWARE SYSTEM FOR PREDICTING

In this chapter, the team will develop the proposed model by following steps such as data collection, preprocessing, model training, and evaluation. Additionally, this chapter includes the analysis and design of the software system for fraud detection.

(iv) Chapter 4: EXPERIMENTAL RESULTS AND DISCUSSION

Based on the model developed in Chapter 3, the team will evaluate the performance and outcomes of the model through various evaluation metrics, and apply these to the fraud detection system.

(v) Chapter 5: CONCLUSION AND FUTURE WORK

This chapter summarizes the findings and achievements of the project, discusses the contributions made, and suggests future directions and potential for further development in this field.

CHAPTER 2. THEORETICAL FOUNDATIONS

Chapter 2 presents the research background and relevant theoretical foundations related to credit card fraud detection. This chapter consists of two main sections: theoretical foundations and applied software. The theoretical foundations section provides concepts, definitions, principles, and methodologies related to credit card transactions, fraud detection, machine learning, and applicable algorithms. It establishes a knowledge base for understanding the key elements of fraud detection and the underlying principles behind machine learning techniques. An essential aspect of this project is the application of software tools and frameworks. The team utilized the Python programming language, PyCharm as the integrated development environment (IDE), MySQL as the database management system, and Qt Designer for creating the user interface. These software tools played a crucial role in implementing and executing the fraud detection system. Chapter 2 serves as the foundation for building the analytical model and detecting fraudulent transactions in the subsequent chapters. It establishes the necessary theoretical background and introduces the software tools that support the project's implementation.Theoretical Foundations

2.1. Theoretical Foundations

2.1.1. Credit card theory

A credit card, in its digital form, functions as a financial tool that allows both personal and business transactions, such as making purchases and obtaining cash advances. It is generally used as a substitute for cash or checks and often offers an unsecured revolving line of credit. The cardholder must repay a portion of the outstanding balance every billing cycle, as specified in the cardholder agreement. As the outstanding debt is reduced, the available credit for accounts in good standing is restored. These financial agreements are complex and subject to frequent changes in terms and fees. The variety of credit card products available to consumers continues to expand. Each credit card product comes with specific terms and conditions, including the Annual Percentage Rate (APR), the formula for the minimum monthly payment, and applicable fees, all of

which are disclosed in the cardholder agreement, as required by regulatory standards. The following sections will explore some common categories of credit card products

2.1.2. Theory applied to the research topic

2.1.2.1. Overview of Machine Learning

Machine learning (ML) is a specialized field within artificial intelligence (AI) that involves the development of algorithms that enable machines to automatically learn from data and improve their performance without explicit programming for every task. Instead of following pre-defined instructions, ML models analyze large datasets to recognize patterns, make predictions, and identify insights that would otherwise be difficult or impossible to program manually. These models can handle a wide variety of tasks, such as predicting stock prices, classifying images, detecting fraud, recommending products, and even autonomously driving vehicles.

The foundation of machine learning is data: the more data a system can access, the more it can learn. This data can be either labeled, meaning that the output is already known (such as a dataset with customer credit scores and whether they defaulted on loans), or unlabeled, where the model must uncover the underlying structure or patterns in the data without guidance. The most common machine learning techniques are categorized based on the type of data they use and the way they learn. Currently, four main types of machine learning are employed:

(i) Supervised learning, the most commonly used type of ML, involves training a model on labeled data, where the input data has corresponding output values. For example, in a supervised learning task such as spam email detection, the algorithm is trained using a dataset of emails labeled as "spam" or "not spam" and learns to classify new emails into those categories based on the patterns it discovers in the labeled examples. Supervised learning is widely used in applications that require predictions or classifications, such as medical diagnoses or fraud detection.

(ii) Unsupervised learning, on the other hand, works with unlabeled data, where the model must find patterns and relationships on its own. This type of learning is often used for clustering, where the system groups similar items together based on shared characteristics, or for dimensionality reduction, where the goal is to reduce the complexity of data while retaining important features. Examples of unsupervised learning include customer segmentation for marketing, anomaly detection for fraud detection, and data compression for efficient storage.

(iii) Semi-supervised learning combines elements of both supervised and unsupervised learning. It uses a small amount of labeled data and a large amount of unlabeled data, which makes it particularly useful when labeling data is expensive or time-consuming. Semi-supervised learning algorithms can achieve high accuracy with less labeled data, making them a valuable tool in many real-world applications, such as image recognition or speech processing, where labeled data may be scarce.

(iv) Reinforcement learning is a more complex form of machine learning where models learn by interacting with an environment and receiving feedback in the form of rewards or penalties. It involves trial-and-error learning, where the model explores different actions, learns from the consequences of those actions, and gradually improves its decision-making process. This type of learning is commonly used in robotics, autonomous vehicles, and game-playing algorithms like AlphaGo.

The success of a machine learning model depends heavily on several factors, including the quality and quantity of data, the choice of algorithm, the computational power available, and the way the model is tuned and validated. While large datasets are typically required for training effective models, they must also be clean and representative of the real-world scenario to avoid biases and inaccuracies in predictions. Additionally, model interpretability remains a significant challenge, as many advanced machine

learning algorithms, particularly deep learning models, are often seen as "black boxes" with limited transparency into how they make decisions.

As the field of machine learning continues to evolve, it is being applied to an increasing range of industries and sectors. In healthcare, ML is being used for disease diagnosis, drug discovery, and personalized medicine. In finance, it helps detect fraudulent activity, predict market trends, and optimize investment strategies. Retailers leverage machine learning to create personalized shopping experiences, optimize supply chains, and predict customer behavior. In autonomous driving, ML algorithms process data from cameras, sensors, and maps to enable vehicles to navigate safely. Machine learning is also transforming industries like manufacturing, entertainment, and agriculture by providing insights into operational efficiency, consumer preferences, and environmental conditions.

Despite its widespread applications, machine learning faces several challenges. Training a model can be computationally expensive and time-consuming, especially when large datasets are involved. Furthermore, ensuring the ethical use of machine learning is an ongoing concern. Issues like data privacy, fairness, and algorithmic bias need to be addressed to prevent unintended consequences. Lastly, domain expertise is often required to understand the context of the data and to interpret the results of machine learning models effectively. These challenges, however, continue to drive innovation in the field, as researchers and practitioners work to improve the efficiency, transparency, and ethical use of machine learning technologies.

2.1.2.2. LightGBM (Light Gradient-Boosting Machine)

LightGBM (Light Gradient Boosting Machine) is a machine learning algorithm based on Gradient Boosting Decision Trees (GBDT), developed by Microsoft. It is designed to optimize speed and performance on large datasets by improving traditional boosting algorithms. Compared to models like XGBoost, LightGBM trains faster, consumes less memory, and handles large-scale data more efficiently.

One of the key differences of LightGBM is its decision tree growth strategy. Instead of growing trees level-wise like XGBoost, LightGBM follows a leaf-wise growth approach. This means it expands the tree at the leaf node with the highest error reduction, rather than growing all nodes at the same level simultaneously. This method allows the model to learn deeper patterns and improve prediction accuracy, but it also increases the risk of overfitting if hyperparameters are not properly tuned.

LightGBM employs a histogram-based technique for splitting data, significantly reducing training time compared to traditional approaches. Instead of checking all possible values to find the optimal split point, the algorithm examines only a small set of representative values, thereby increasing efficiency while maintaining good accuracy. This makes LightGBM particularly well-suited for handling large datasets with numerous numerical features.

Another major advantage of LightGBM is its ability to handle sparse data efficiently. It can work with datasets containing many zero values or missing values without requiring extensive preprocessing. Additionally, LightGBM supports categorical features directly, eliminating the need for one-hot encoding, which reduces data dimensionality and speeds up training.

Despite its advantages, LightGBM has some drawbacks. Because it optimizes error reduction aggressively, the model can become overly complex and prone to overfitting if not properly regulated with constraints like maximum tree depth or regularization techniques. Furthermore, while LightGBM is highly efficient for large datasets, it may not always outperform XGBoost or CatBoost on smaller datasets or those with specific structures.

Thanks to these innovations, LightGBM is widely used in various machine learning applications, particularly in finance, fraud detection, search ranking, and other areas requiring efficient large-scale data processing.

2.1.2.3. Synthetic Minority Oversampling Technique (SMOTE)

The Synthetic Minority Oversampling Technique (SMOTE) is a widely used method for addressing class imbalance in machine learning datasets, particularly when the minority class is underrepresented. This technique works by generating synthetic instances of the minority class rather than merely duplicating existing instances. The process begins by identifying the nearest neighbors of each minority class instance in the feature space. SMOTE then creates new synthetic instances by interpolating between the minority instance and its neighbors, effectively expanding the decision boundary for the minority class. This approach helps the model to better generalize and reduces the risk of overfitting, which is common when merely replicating minority class instances (Chawla et al., 2002).

SMOTE is particularly useful in scenarios where imbalanced datasets can severely affect the performance of machine learning algorithms. For instance, in fraud detection, class imbalances can range from 100:1, and in more extreme cases, up to 100,000:1, making it difficult for traditional models to accurately detect the minority class. In such cases, evaluation metrics like prediction accuracy become unreliable because they often do not reflect the true performance of the model on the minority class. SMOTE alleviates this issue by generating synthetic instances that provide a more balanced training set, enabling the model to learn more effectively from both classes.

One of the key advantages of SMOTE over other oversampling methods is its ability to generate synthetic examples that are not exact copies of existing data points, but rather new, distinct instances based on the underlying structure of the data. This leads to a richer feature space and allows the model to learn better representations of the minority class. Furthermore, SMOTE can be easily combined with other techniques, such as Tomek Links or Edited Nearest Neighbors (ENN), to improve the quality of the generated instances and reduce the possibility of introducing noise.

Despite its benefits, SMOTE is not without challenges. For instance, it can sometimes generate synthetic instances that are far from the true distribution of the

minority class, especially when the data is noisy. Additionally, SMOTE may not be suitable for certain types of data where generating new instances based on interpolation may not reflect the actual characteristics of the minority class. Nonetheless, when used appropriately, SMOTE is an effective tool for improving the performance of machine learning models in imbalanced class scenarios, leading to better overall results in tasks like fraud detection, medical diagnosis, and more.

In conclusion, SMOTE is a powerful oversampling technique that enhances the learning process in imbalanced datasets by generating synthetic instances for the minority class. By expanding the feature space and providing a more balanced training set, SMOTE helps machine learning algorithms to better learn and generalize from the minority class, resulting in improved model performance. Its versatility and ability to reduce overfitting make SMOTE an invaluable tool for tackling class imbalance problems in various real-world applications.

Overall, SMOTE is a powerful technique for addressing imbalanced datasets by intelligently generating new minority class examples. This helps to improve the model's ability to learn the underlying patterns in the minority class, leading to better overall performance.

2.1.2.4. Evaluation metrics for models

Evaluating the performance of machine learning models is a critical step in the modeling process, ensuring that the model performs as expected and meets the required standards for deployment. The evaluation of a model’s effectiveness typically involves several performance metrics that provide insights into different aspects of its performance. Common metrics include accuracy, precision, recall, F1-score, and the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve. These metrics help to understand how well a model generalizes to new data and can effectively distinguish between classes.

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

Figure 2.1. Formula to calculate Accuracy (Vujović, 2021)

Accuracy is one of the simplest and most widely used metrics. It is calculated by dividing the number of correct predictions by the total number of predictions made. While accuracy offers an overall assessment of a model's performance, it may not always be a reliable indicator in cases where the dataset is imbalanced. For example, in a scenario where 95% of the instances belong to one class, predicting only the majority class would result in a high accuracy, yet the model would fail to identify the minority class. In such cases, additional metrics like precision and recall become necessary to get a more complete understanding of the model's performance.

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

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

Figure 2.2. Formula to calculate Precision and Recall (Vujović, 2021)

Precision and recall are metrics that are particularly useful in evaluating models when class imbalance exists. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It indicates how many of the predicted positive instances are actually correct. Recall, on the other hand, measures the ratio of correctly predicted positive observations to all actual positive observations in the dataset. It helps to understand how well the model captures all relevant instances of the positive class. The relationship between precision and recall is often balanced using the F1-score, which is the harmonic mean of precision and recall. The F1-score is particularly

useful when the class distribution is uneven, as it combines both precision and recall into a single metric that gives a more balanced view of the model's performance.

The ROC curve and the AUC are powerful tools used to evaluate the performance of a classification model, particularly in situations where the threshold for classifying positive and negative predictions can be adjusted. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for various threshold values, allowing us to visualize the trade-off between sensitivity (recall) and specificity (1 - FPR). An ideal model will have an ROC curve that hugs the top-left corner of the plot, indicating that it has a high true positive rate and a low false positive rate. The AUC value represents the area under this curve, with higher values indicating better model performance. An AUC of 0.5 suggests no discriminatory power, while an AUC of 1.0 indicates perfect classification.

In conclusion, evaluating machine learning models is essential for assessing their performance and ensuring they meet the desired criteria for deployment. Using a combination of metrics such as accuracy, precision, recall, F1-score, and AUC-ROC allows practitioners to comprehensively evaluate model performance from different perspectives. These evaluation metrics provide valuable insights into the strengths and weaknesses of a model, enabling data scientists to fine-tune and optimize it before putting it into real-world applications. Each metric serves a specific purpose, and selecting the right metrics is critical in understanding the effectiveness of the model in different contexts.

2.2. Applied software and tools

2.2.1. PyCharm

PyCharm is a Python IDE developed for coding in Python. Among the provided features are code analysis, a graphical debugger, unit testing inside, effortless integration with systems of version control, and support of web development using the Django. This is a product of the Czech Republic software company JetBrains.

PyCharm is an integrated development environment that runs on several operating systems, including Windows, Mac OS, and Linux. It provides a number of useful features to software developers. In code editing, PyCharm supports programmers in writing high-quality code by providing syntax highlighting for keywords, classes, and functions. It helps detect errors and also provides automatic completion of code, hence making coding easier and faster. It also does great in code navigation, helping developers go through their codebase easily. In PyCharm, intuitive navigation to symbols, elements, and variables within the source code will be available to the programmers for effective code exploration and modification. More importantly, PyCharm offers very robust debugging. The tool provides comprehensive code debugging and has a lens mode to inspect and debug the whole source code. This feature makes the finding and fixing of problems faster, hence boosting development. Another essential feature in PyCharm involves refactoring. It enables developers to make rapid changes in the variables for improvement of the codebase internally, and where necessary, improvements in code maintainability are made possible. Besides PyCharm supporting flexible restructuring of code, it enables the separation of classes and functions for better extensibility.

In general, PyCharm enables software developers to write high-quality code, navigate and modify their projects effectively, and refactor their codebase efficiently. It is a powerful tool for enhancing productivity and maintaining clean, well-structured code.

2.2.2. MySQL Workbench

MySQL Workbench is the unified visual tool for database architects, developers and DBAs. MySQL Workbench provides data modeling, SQL development and comprehensive administration tools for server configuration, user administration and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.

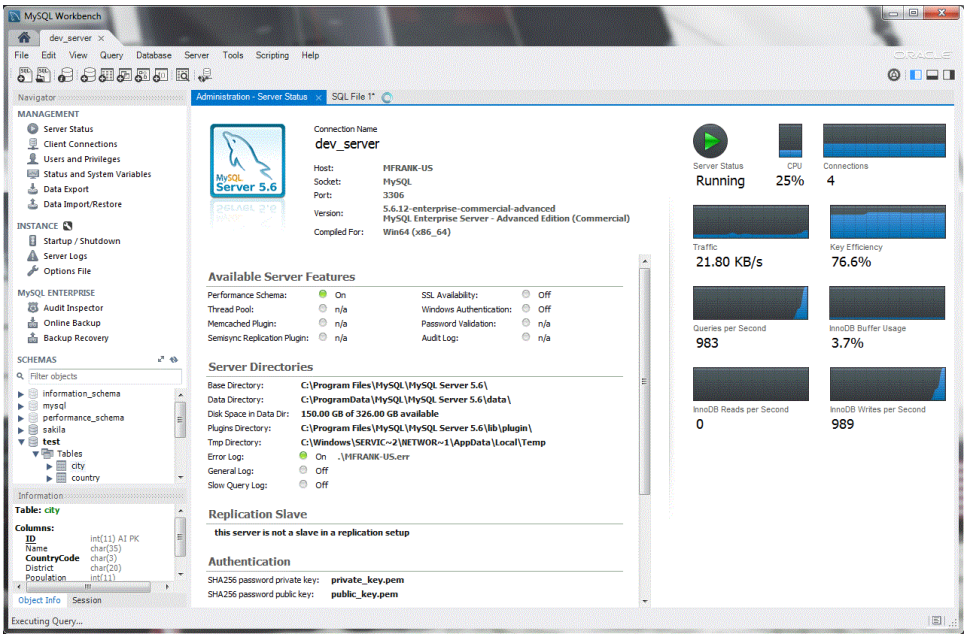


Figure 2.3. MySQL Workbench Home (Source: MySQL)

MySQL Workbench offers several essential features:

(i) Modeling and Designing: There is a GUI for designing Database Model: It also provides support to multipage models on one surface for editing Graphical model in a TABLE EDITOR

(ii) Development: It supports the SQL programming language, and a person can execute SQL queries for creating reading update and deleting an operation on RDBMS: About a focal location and also a workbench for all kinds of Databases

(iii) Administration: It provides for setting up servers, user administration, and backup processes. It has different sets of features to perform the administrative role for databases.

(iv) Database Migration: The MySQL Workbench allows easy connections and migrations between different databases, allowing the transfer of data and structures.

Above all, MySQL Workbench is a compelling tool, at ease with simplifying the process of database modeling, development, administration, and migration. This tool provides comfort to the user with its user-friendly interface, coupled with inclusive features for handling databases efficiently.

2.2.3. Qt Designer

Qt Designer is a tool that can be used with Qt in designing and developing Graphical User Interfaces. Through its easy drag-and-drop interface, we can drop buttons, text fields, combo boxes, etc., without prior coding.

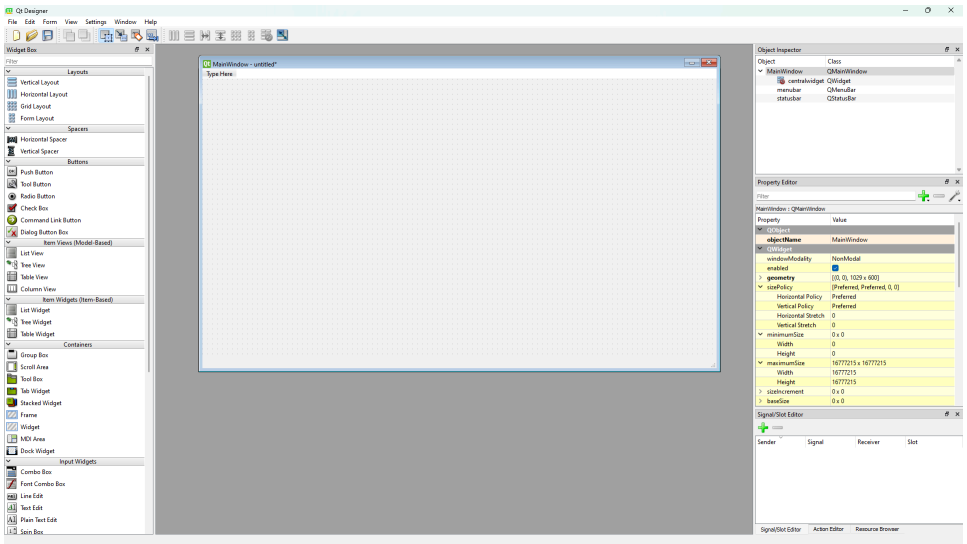


Figure 2.4. Qt Designer software interface (Source: Authors)

Qt Designer is easiest to use with Python when the former is integrated with PyQt. For general purposes, PyQt is the Python interface to Qt, which merges the Python

programming language with the Qt library. It comprises interface control widgets and also allows the facilitation of designing user interfaces for functional software applications.

Qt Designer and PyQt make it easier for developers to create attractive and interactive GUIs in their Python applications without requiring them to write much code.

CHAPTER 3. PROPOSAL MODEL DEVELOPMENT

After studying the theoretical foundations, the team proceeded with analysis, model construction, and software design for integration. A detailed presentation of this content will be provided in Chapter 3. Specifically, the team established objectives, followed by data collection and preprocessing. This formed the fundamental basis for proposing and constructing models to achieve optimal performance. The proposed model is presented by the team in four stages, as shown in **Figure 3.1**.

In the data collection phase, the team gathered a comprehensive credit card transaction dataset from Kaggle, which includes features such as transaction amount, location, time, merchant details, and user behavior patterns.

The preprocessing phase involved exploratory data analysis (EDA), cleaning the data, handling missing values, and transforming the dataset to ensure consistency and accuracy. Additionally, feature engineering techniques were applied to enhance fraud detection by identifying key behavioral indicators.

For the model construction phase, the team selected the LightGBM model due to its robustness in detecting anomalies and classifying fraudulent transactions. The LightGBM model was meticulously tuned through hyperparameter optimization and cross-validation to enhance its detection accuracy while minimizing false positives.

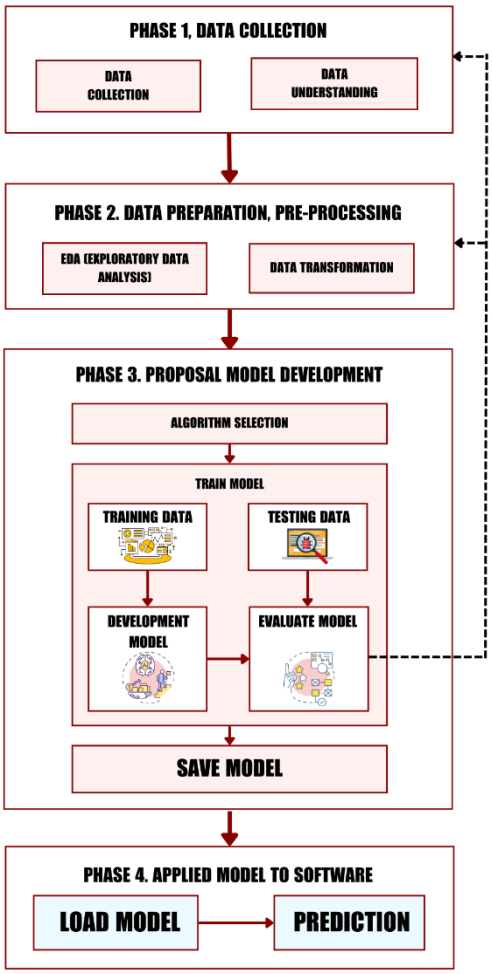


Figure 3.1. Proposal Model (Source: Authors – See more)

3.1. Data collection

The input data source has been taken from a publicly available dataset and does not pose any information security concerns. The dataset used in this study contains credit card transaction records, including both legitimate and fraudulent transactions.

This dataset consists of various transaction attributes such as transaction date and time, credit card number, merchant information, transaction amount, customer details (e.g., name, gender, address, job), and geographical coordinates. The dataset also includes a fraud label ("is_fraud"), where "0" represents legitimate transactions, and "1" indicates fraudulent transactions.

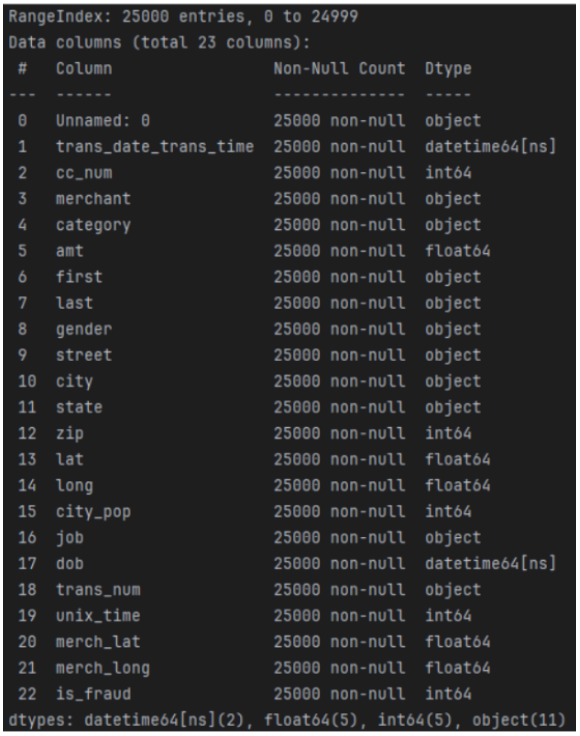
The dataset is stored in a CSV file and provides a detailed view of transactions made by credit cardholders. This dataset is utilized for fraud detection analysis, helping to develop models that can identify potentially fraudulent transactions based on various transaction attributes. The details of the dataset are presented in **Table 3.1**.

Table 3-1. Detail Information about application data set (Source: Authors)

	Attribute Name	**Description**	**Data Type**	**Possible Values**
1	**Unnamed: 0**	Row index	Integer	
2	**trans_date_trans_time**	Transaction timestamp	datetime	
3	**cc_num**	Credit card number	Integer	
4	**merchant**	Merchant name	string	fraud_Rippin, Kub and Mann, fraud_Heller, Gutmann and Zieme, fraud_Rowe-Van dervort, etc.
5	**category**	Transaction category	string	misc_net, grocery_net, entertainment, etc.
6	**amt**	Transaction amount	float	
7	**first**	Customer first name	String	Jenifer, Edward, Tyler, etc.
8	**last**	Customer last name	String	Chase, Hill, Gill, etc.
9	**gender**	Customer gender (M: Male, F: Female)	character	{M;F}
10	**street**	Customer street address	String	561 Perry Cove, 4655 David Island, 870 Rocha Drive, etc

11	**city**	Customer city	string	Manor, Orient, Dublin, etc.
12	**state**	Customer state	string	PA, VA, KS, etc
13	**zip**	ZIP code	Integer	
14	**lat**	Customer latitude	float	
15	**long**	Customer longitude	float64	
16	**city_pop**	City population	int64	
17	**job**	Customer occupation	string	Systems developer, Naval architect, IT trainer, etc.
18	**dob**	Customer date of birth	datetime	
19	**trans_num**	Transaction ID	string	
20	**unix_time**	Transaction timestamp (Unix)	Integer	
21	**merch_lat**	Merchant latitude	float	
22	**merch_long**	Merchant longitude	float	
23	**is_fraud**	Fraudulent transaction (0: Normal, 1: Fraudulent)	integer	{0;1}

The data shape of the dataset, there are 25000 rows, 23 columns as shown in **Figure 3.2**.



```
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             25000 non-null  object
 1   trans_date_trans_time  25000 non-null  datetime64[ns]
 2   cc_num                 25000 non-null  int64
 3   merchant               25000 non-null  object
 4   category               25000 non-null  object
 5   amt                    25000 non-null  float64
 6   first                  25000 non-null  object
 7   last                   25000 non-null  object
 8   gender                 25000 non-null  object
 9   street                 25000 non-null  object
 10  city                   25000 non-null  object
 11  state                  25000 non-null  object
 12  zip                    25000 non-null  int64
 13  lat                    25000 non-null  float64
 14  long                   25000 non-null  float64
 15  city_pop               25000 non-null  int64
 16  job                    25000 non-null  object
 17  dob                    25000 non-null  datetime64[ns]
 18  trans_num              25000 non-null  object
 19  unix_time              25000 non-null  int64
 20  merch_lat              25000 non-null  float64
 21  merch_long             25000 non-null  float64
 22  is_fraud               25000 non-null  int64
dtypes: datetime64[ns](2), float64(5), int64(5), object(11)
```

Figure 3.2. Detail Information about application data set (Source: Authors)

By analyzing credit card transaction data, we can observe that while the overall transaction volume has steadily increased over time, there is a notable pattern where fraudulent transactions tend to occur more frequently for higher transaction amounts. Based on this insight, financial institutions can enhance their fraud detection systems by implementing stricter monitoring for large transactions and refining their fraud detection models to mitigate risks more effectively.

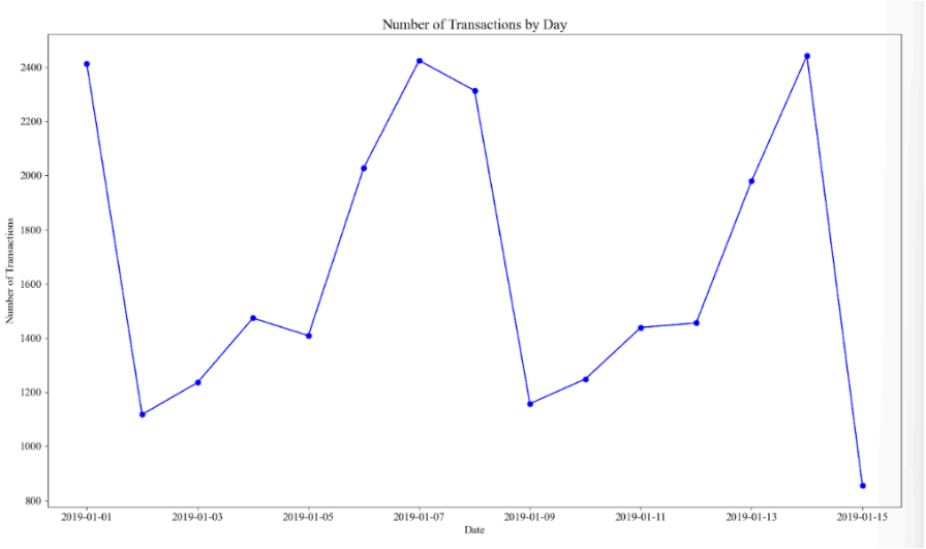


Figure 3.3. Number of transactions by day (Source: Authors)

The chart shows the number of transactions per day with strong fluctuations, featuring distinct peaks and troughs. The number of transactions varies significantly from day to day, reaching over 2,400 at certain times but dropping below 1,000 on others. This trend suggests a cyclical pattern, where after a period of strong growth, the number of transactions tends to decline sharply. Additionally, these changes occur rapidly rather than following a gradual upward or downward trend over time.

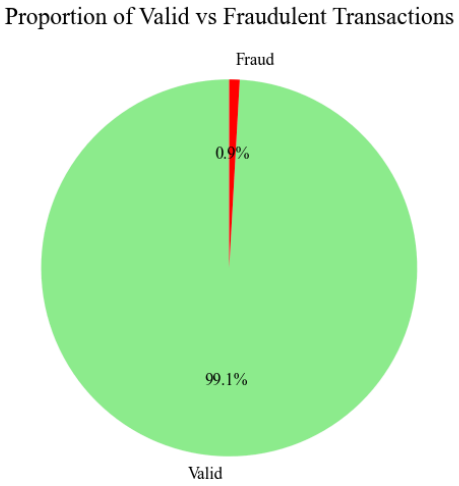


Figure 3.4. Proportion of valid and fraudulent transactions (Source: Authors)

According the **Figure 3.4**, proportion of valid and fraudulent transactions of the customers are 0.9% and 99.1% are valid.

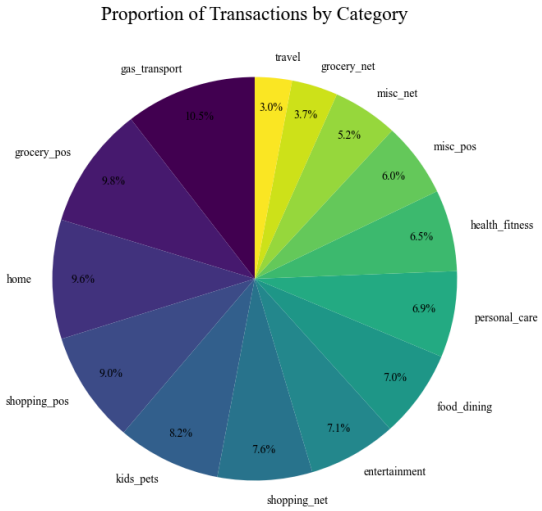


Figure 3.5. Proportion of transactions by gender (Source: Authors)

According to **Figure 3.5**, Proportion of transactions by gender 10.5% are gas_transport, 3.0% are travel, 3.7% are grocery_net, 5.2% are mise_net, 6.0% are mise_pos, 6.5% health_fitness, 6.9% personal_care , 7.0% are food_dinning, 7.1% are entertainment, 7.6% are shopping_net, 8.2% are kids_pets, 9.0% are shopping_pos, 9,6% are home and 9,8% are grocery_pos.

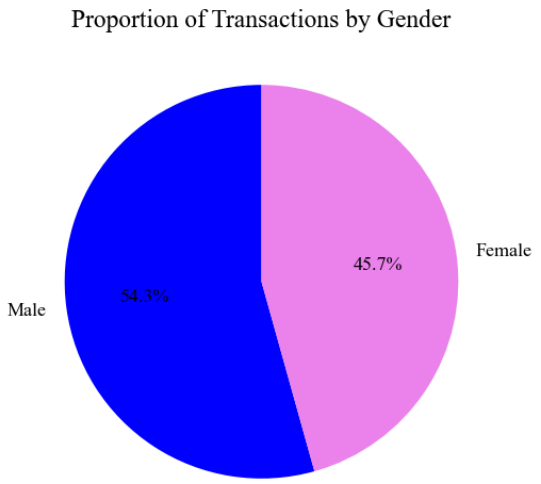


Figure 3.6. Proportion of transactions by gender (Source: Authors)

According to **Figure 3.6,** proportion of transactions by gender of the customers are 45.7 % female and 54.3% are male.

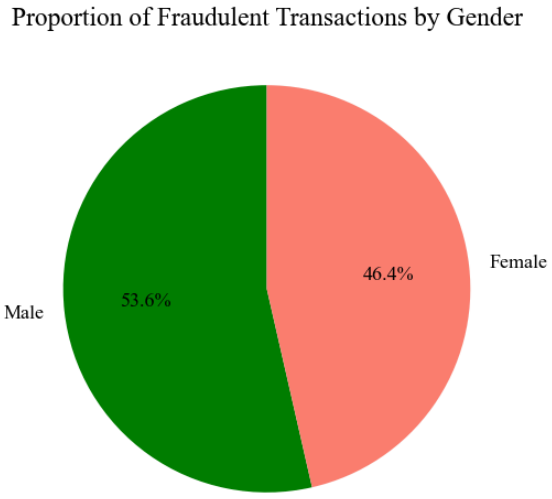


Figure 3.7. Proportion of fraudulent transactions by gender (Source: Authors)

According to **Figure 3.7**, proportion of fraudulent transactions by gender of the customers are 46.4 % female and 53.6% are male.

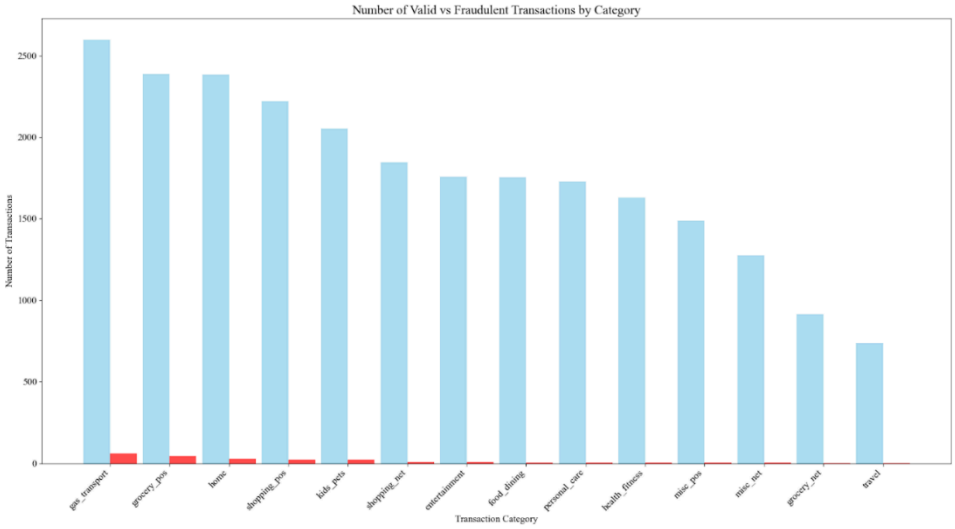


Figure 3.8. Number of valid and fraudulent transactions by category (Source: Authors)

As shown in **Figure 3.8**, The chart shows the number of valid and fraudulent transactions by category. The majority of transactions are valid, while the number of fraudulent transactions is very small. Some categories with high total transactions such as gas_transport, grocery_pos, home, shopping_pos also have higher fraud levels than other categories.

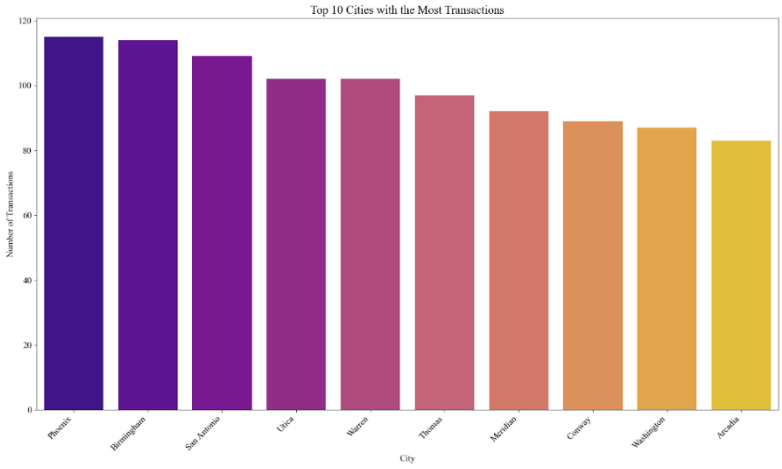


Figure 3.9. Top 10 cities with the most transactions (Source: Authors)

As shown in **Figure 3.9**, The chart shows the top 10 cities with the highest number of transactions. The cities with the highest number of transactions include Phoenix, Birmingham, and St. Louis, while cities such as Charlotte and Austin have lower numbers of transactions. The difference between cities is not too large, the level of transactions is distributed relatively evenly among the top cities.

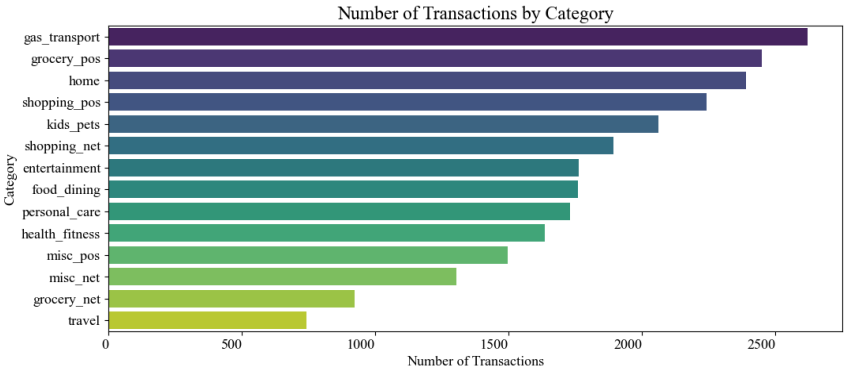


Figure 3.10. Number of transactions by category (Source: Authors)

As shown in **Figure 3.10**, the transaction with the highest number of transactions is gas_transport with more than 2500 transactions and the lowest is travel with around 750 transactions.

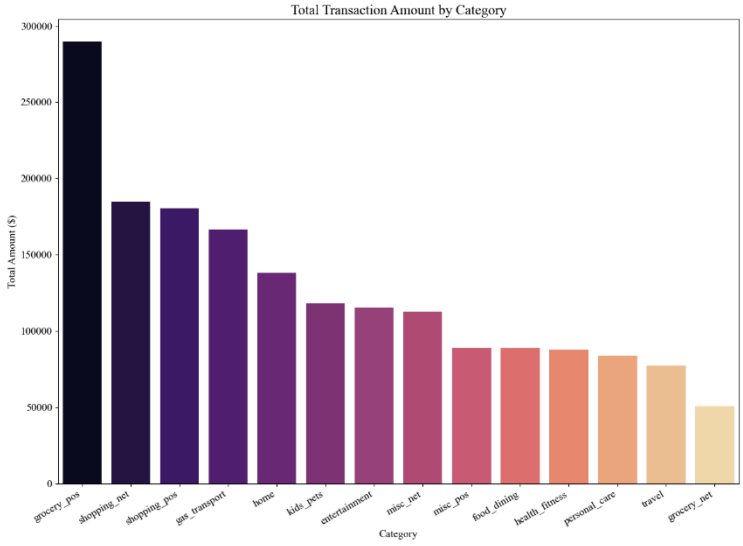


Figure 3.11. Total transaction amount by category (Source: Authors)

As shown in **Figure 3.11**, The chart shows the total transaction amount by category. The grocery_pos category has the highest total transaction value, significantly outperforming the other categories. The shopping_net, shopping_pos, gas_transport categories also have significant transaction values. Meanwhile, travel and grocery_net have the lowest total transaction value.

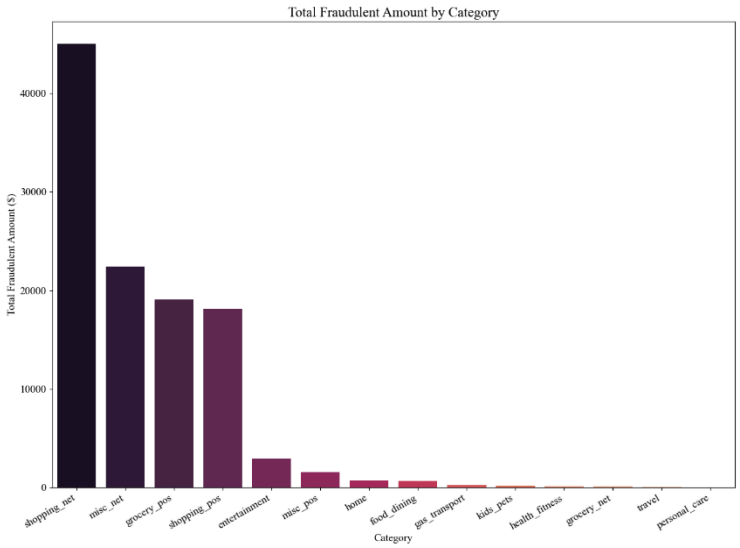


Figure 3.12. Total fraudulent amount by category (Source: Authors)

As shown in **Figure 3.12** The chart shows the total fraud amount by category. The shopping_net category has the highest fraud amount. The categories gas_transport, grocery_pos, shopping_pos also have significant fraud, online shopping and essential consumer transactions have a higher risk of fraud. Meanwhile, categories such as personal_care, travel, grocery_net have very low fraud amount.

3.2. Data preparation

In this step, we start by checking and handling missing values in the data. By using the df.isnull().sum() function of the pandas library, we can easily determine the number of missing values in the DataFrame. The results from the checking process are shown below:

```
Unnamed: 0              0
trans_date_trans_time   0
cc_num                  0
merchant                0
category                0
amt                     0
first                   0
last                    0
gender                  0
street                  0
city                    0
state                   0
zip                     0
lat                     0
long                    0
city_pop                0
job                     0
dob                     0
trans_num               0
unix_time               0
merch_lat               0
merch_long              0
is_fraud                0
dtype: int64
```

Figure 3.13. Missing values check results (Source: Authors)

From the results of checking the missing values, we observe that there are no missing values in the dataset across all columns.

```
Most positive correlations with is_fraud:
      Feature  Correlation
0    is_fraud     1.000000
1         amt     0.277629
2   unix_time     0.045225
3   merch_lat     0.028603
4         lat     0.027844
5    city_pop     0.013051
6        long     0.001857
7  merch_long     0.001833
```

Figure 3.14. Most positive correlations with “is_fraud” (Source: Authors)

```
Most negative correlations with is_fraud:
  Feature  Correlation
0  cc_num    -0.013691
1     zip    -0.030502
```

Figure 3.15. Most Negative Correlations (Source: Authors)

The correlation heatmap created in this stage offers essential insights into the connections between different variables in the normalized dataset. The color intensity in

the heatmap reflects the strength of the correlations, with darker shades indicating stronger positive or negative correlations and lighter shades representing weaker correlations.

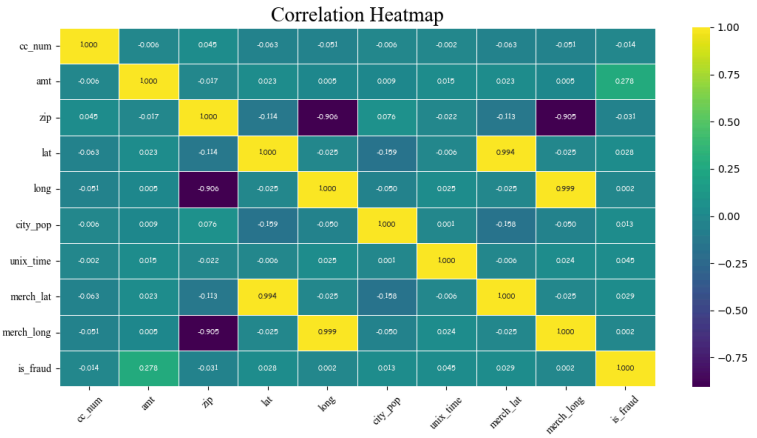


Figure 3.16. Correlation Heatmap (Source: Authors – See more)

From the correlation heatmap in **Figure 3.16**, we can observe that the strongest correlation is between zip code and latitude, with a correlation coefficient of -0.30, and between zip code and longitude, with a correlation coefficient of -0.39. Additionally, the correlation between merch_lat and merch_long is notably high, indicating that merchant location data is strongly related. However, the correlation between is_fraud and other variables is relatively weak, suggesting that fraudulent transactions may not be strongly influenced by any single variable but rather by a combination of factors.

The final critical step in preprocessing was normalizing the data using StandardScaler. This stage is vital in data preprocessing, especially for machine learning and data mining tasks. The goal is to adjust the data features to share a consistent range or measurement unit. This ensures that no single feature disproportionately impacts the model training process. One of the most widely used normalization techniques is the StandardScaler from Python's sklearn library. This method modifies each feature's values so that they approximate a mean of 0 and a standard deviation close to 1, resolving scale transformation issues. This step is crucial for ensuring that machine learning algorithms

perform effectively on the normalized data and that prediction outcomes are reliable when applied to new data. Data normalization is indispensable for optimizing model performance.

3.3. Proposal model development

After preprocessing the data, the team proceeded with building and developing the proposed model model to optimize credit card fraud prediction. The dataset exhibited a significant imbalance between the observations of fraudulent and non-fraudulent transactions, where 24,776 transactions were non-fraudulent, while only 224 transactions were fraudulent. This imbalance could lead to biased learning, reducing the model's ability to correctly identify fraudulent transactions. Addressing this imbalance is crucial in ensuring that the model does not develop a bias towards the majority class, which would negatively impact its ability to correctly classify fraudulent transactions.

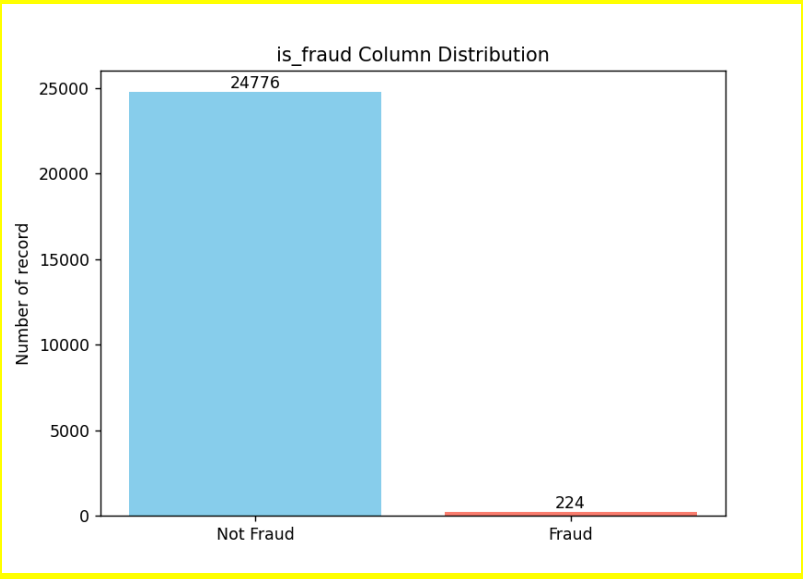


Figure 3.17. Class Imbalance in Fraud Detection: Distribution of 'is_fraud' Variable (Source: Authors)

To address this issue, the team applied Synthetic Minority Over-sampling Technique (SMOTE), which generates synthetic samples from the minority class rather

than simply duplicating existing data. After applying SMOTE, the training dataset became more balanced, with 19,818 observations for both classes.

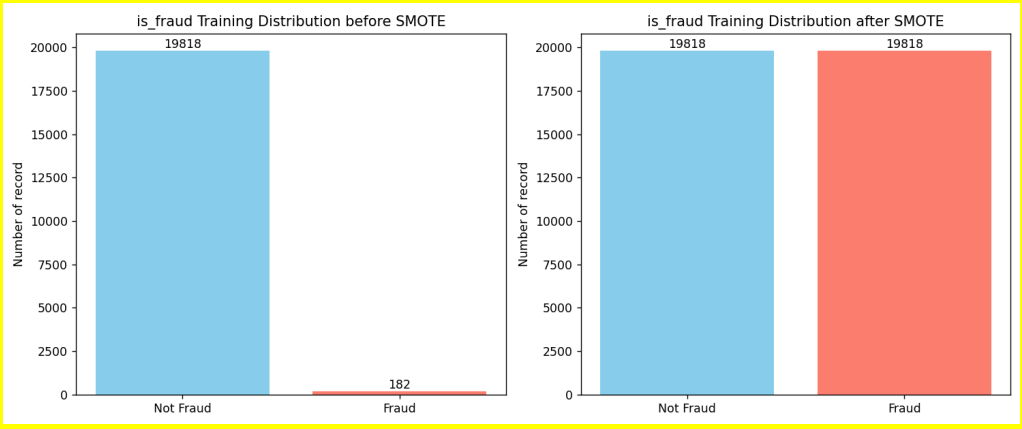


Figure 3.18. Distribution of 'is_fraud' Variable in training dataset before and after SMOTE (Source: Authors)

The original dataset showed a 1:100 imbalance ratio, which was reduced to 1:1 after the application of SMOTE. This allowed the model to learn fraudulent transaction patterns more effectively, improving classification accuracy. By artificially generating synthetic data points based on the feature space of the existing minority samples, SMOTE ensures that the model is exposed to a diverse set of fraudulent transaction patterns, enhancing its predictive capabilities.

In **Figure 3.18**, we can see the distribution of transactions before and after SMOTE. The synthetic data creation has significantly increased the number of fraudulent transactions in the training set, making it easier for the model to recognize fraudulent behavior. This directly contributes to the improvement of classification performance in later phases.

The processed dataset was split into 80% for training and 20% for testing, ensuring that the model was evaluated on unseen data. The application of SMOTE significantly enhanced the model's ability to detect fraudulent transactions, overcoming the challenges of training machine learning models on imbalanced datasets. The training process was

conducted using cross-validation to prevent overfitting and ensure that the model generalizes well to new data.

The model was trained with the following hyperparameters:

Table 3-2. Hyperparameter Configuration for LightGBM (Source: Authors)

No.	Parameter	Value
1	n_estimators	200, 300
2	num_leaves	31, 62
3	max_depth	7, 9
4	learning_rate	0.05, 0.1
5	is_unbalance	TRUE

- **n_estimators** = 200, 300: This defines the number of boosting iterations. A higher value increases model complexity and potential accuracy but also raises the risk of overfitting.
- **num_leaves** = 31, 62: Determines the complexity of tree structures. More leaves allow for capturing more patterns but may lead to overfitting if not properly tuned.
- **max_depth** = 7, 9: Specifies the maximum depth of trees. Deeper trees can improve learning but also increase computation time and overfitting risks.

- **learning_rate** = 0.05, 0.1: Controls how much the model adjusts with each iteration. A smaller learning rate leads to more gradual learning but requires more boosting rounds.
- **is_unbalance** = TRUE: Indicates that the dataset is imbalanced, and the model should automatically adjust weights to improve classification performance.

Apply the initialized LightGBM to the balanced Training Data for training. Then, use the Testing Data for testing to evaluate the achieved results and assess the quality of the model. If the model performs well, proceed to save it and integrate it into the software system. If the model does not meet the desired quality, there could be various solutions. In the proposed model, we suggest revisiting Phase 1 and Phase 2. Both of these stages could be the reasons for the model's unsatisfactory performance, as understanding and preprocessing are crucial in achieving desired model quality.

CHAPTER 4. IMPLEMENTATION AND RESULTS

Next, the team will integrate a fraud detection system into the software, enabling real-time monitoring and detection of fraudulent activities. The software design process emphasizes both model accuracy and user experience, ensuring that bank analysts can easily navigate and utilize the system effectively.

The system features an intuitive interface that facilitates transaction tracking, generates alerts for suspicious activities, and provides advanced analytical tools to uncover emerging fraud trends. Additionally, integrated data visualization tools allow users to quickly recognize common fraud patterns, enabling timely and informed decision-making to mitigate risks.

Moreover, the system generates comprehensive reports on transaction activities and fraud risks, equipping banks with valuable insights to formulate robust fraud prevention strategies. These capabilities not only help minimize financial losses but also strengthen risk management and optimize fraud control mechanisms within banking operations.

4.1. Integrated software, applying models for prediction

4.1.1. Login

The login process plays a crucial role in safeguarding your personal information by implementing security measures that ensure only authorized individuals can access your account. This process requires users to provide valid credentials, such as a user ID and password, which act as the first layer of protection. By verifying the identity of users, the login system helps prevent unauthorized access to your account, minimizing the risk of data breaches and protecting sensitive information. Additionally, this security measure helps maintain the confidentiality of your important data, preventing unwanted intrusion or misuse by malicious parties. The login process is essential for establishing a secure environment, giving you peace of mind that your personal and financial information remains protected.

Figure 4.1. Login Interface (Source: Authors)

4.1.2. Statistic Tab

The Statistics page is an important part of this software, helping users gain a clearer understanding of the applicant's information. After logging in, users can view charts and data tables related to the graphs. The charts we refer to on this page are: In the Statistics section, the application provides various analytical functions, such as detecting fraud by gender, category, state, and other factors.

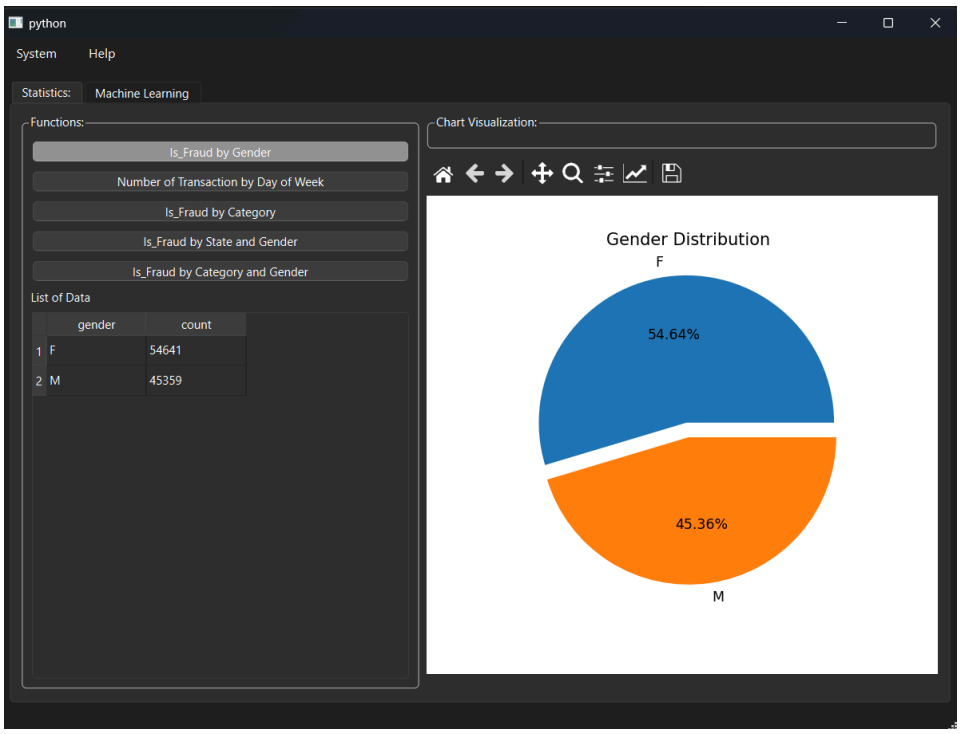


Figure 4.2. Distribution of Is _Fraud by Gender (Source: Authors)

Figure 4.2 displays the Statistics interface for the Distribution of Is _Fraud by Gender . On the left side, there is a list of available functions, with the Is_Fraud by Gender button currently selected. Below the function list, a data table displays the total number of fraudulent transactions by gender, showing 54,641 transactions for females (F) and 45,359 for males (M).

On the right side, the data visualization panel presents a pie chart illustrating fraud distribution by gender. The chart indicates that fraudulent transactions involving females account for 54.64% (blue), while those involving males make up 45.36% (orange). Above the chart, a toolbar contains control buttons for zooming, panning, and saving the image.

The interface is intuitively designed, allowing users to easily analyze fraud data based on different criteria, supporting informed decision-making.

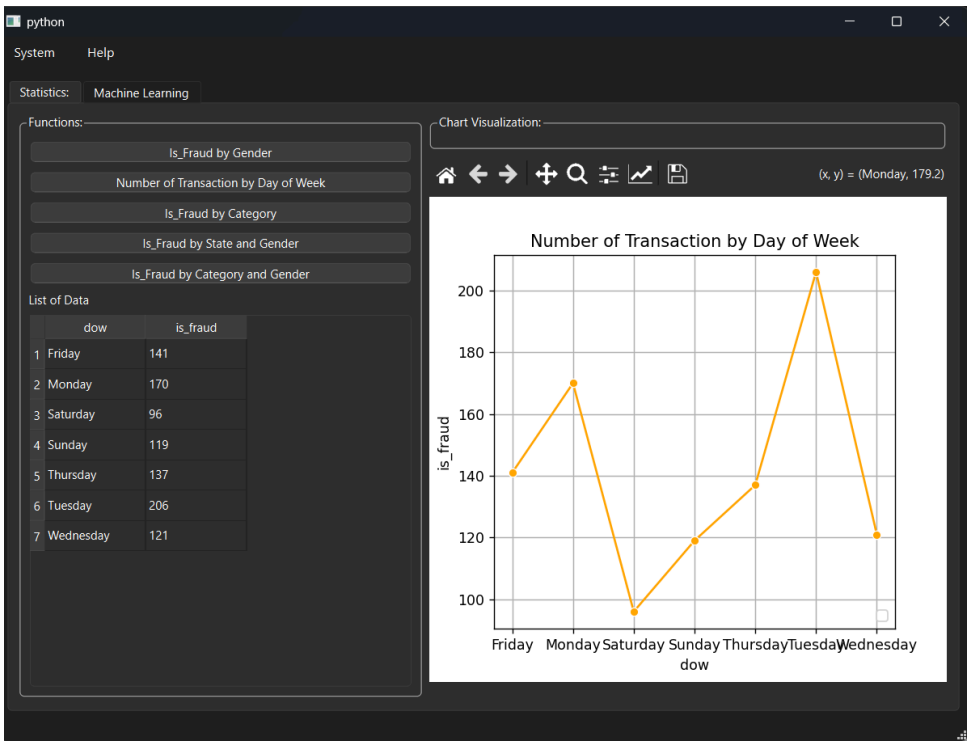


Figure 4.3. Number of Transactions by Day of Week Interface (Source: Authors)

According to **Figure 4.3**, On the left side, a data table displays the number of fraudulent transactions for each day of the week. The highest number of fraud cases occurs on Tuesday (206), followed by Monday (170), while Saturday has the lowest count (96).

On the right side, the data is visualized using a line chart labeled Number of Transaction by Day of Week. The x-axis represents the days of the week, while the y-axis indicates the number of fraudulent transactions. The plotted points are connected by an orange line, illustrating fluctuations in fraudulent activities. A noticeable peak appears on Tuesday, while Saturday experiences the lowest fraud rate.

Above the chart, a toolbar provides controls for zooming, panning, and saving the visualization. The cursor in the chart is currently hovering over Monday, displaying the coordinates (Monday, 179.2), suggesting an interactive feature for precise data inspection.

The interface is well-structured, making it easy for users to analyze fraud patterns over time, identify high-risk days, and derive actionable insights for fraud prevention strategies.

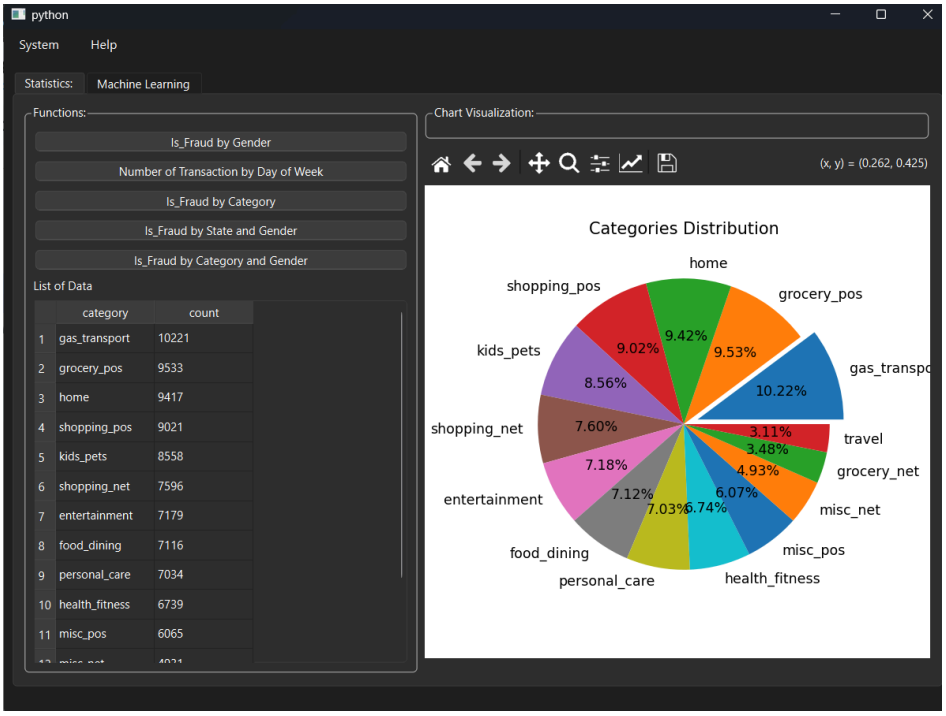


Figure 4.4. Is Fraud by Category Interface (Source: Authors)

On the left side, a data table displays the number of fraudulent transactions for each category. The gas_transport category has the highest fraud count (10,221), followed by grocery_pos (9,533) and home (9,417). The lowest fraud counts are found in misc_net and misc_pos.

On the right side, a pie chart titled Categories Distribution visualizes the fraud distribution across different spending categories. Each segment represents a category, with labels and percentage values indicating the proportion of fraud cases in that category. The largest segment belongs to gas_transport (10.22%), while smaller segments represent categories like travel (3.11%) and grocery_net (4.93%).

Above the pie chart, a toolbar allows users to interact with the visualization by zooming, panning, and saving the chart. A cursor is hovering over a section of the pie

chart, displaying its coordinates, indicating that the chart supports interactive data exploration.

The interface is well-structured, making it easy for users to analyze fraud trends across different spending categories. This visualization helps identify high-risk transaction types, aiding in fraud detection and prevention strategies.

4.1.3. Machine Learning Tab

After logging into the system, the user can interact with the buttons on the Machine Learning Tab as follows:

- View Data: Click this button to display the dataset containing transaction records. This allows users to inspect the data before training the model.
- Train Model: After selecting the test data ratio (Test Size) and setting a Random State, click this button to train the machine learning model using the available dataset.
- Evaluate Model: Once the model has been trained, click this button to assess its performance. The evaluation section will display key metrics, including True Positive, False Positive, True Negative, False Negative, Accuracy, Recall, Precision, F1-Score, and ROC AUC.
- Predict: After entering relevant details such as Category, ATM, Gender, State, Zip, Job Title, and Age, click this button to predict whether a transaction is fraudulent based on the trained model.

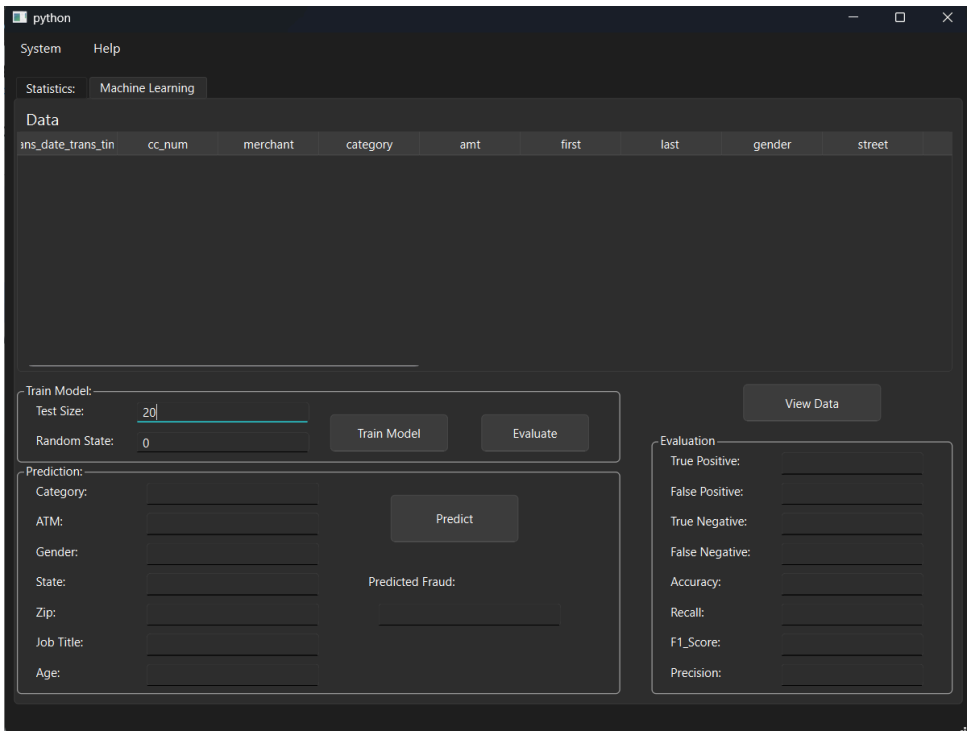


Figure 4.5. "Machine Learning Tab" Interface (Source: Authors)

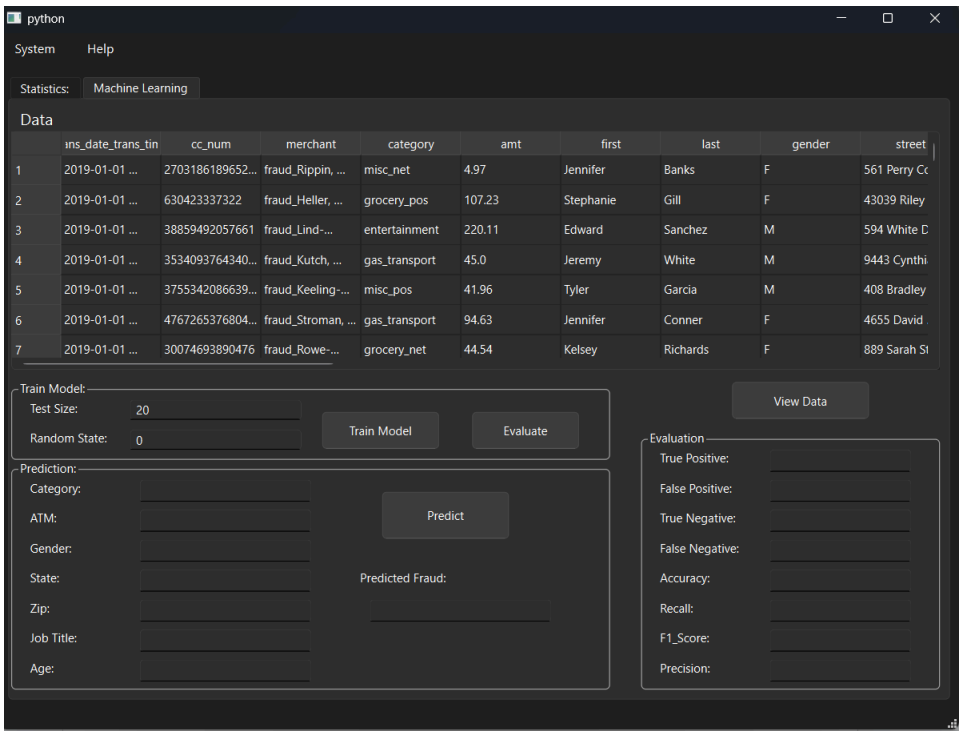


Figure 4.6. Interface when clicking to view "Applicant Data" (Source: Authors)

When Training the Model, users can select and customize the Test Size and Random State parameters according to their preferences in order to optimize the model. After making these selections, the user can click the "Train" button. After the training process is successfully completed, a pop-up notification will be displayed as shown in **Figure 4.24**.

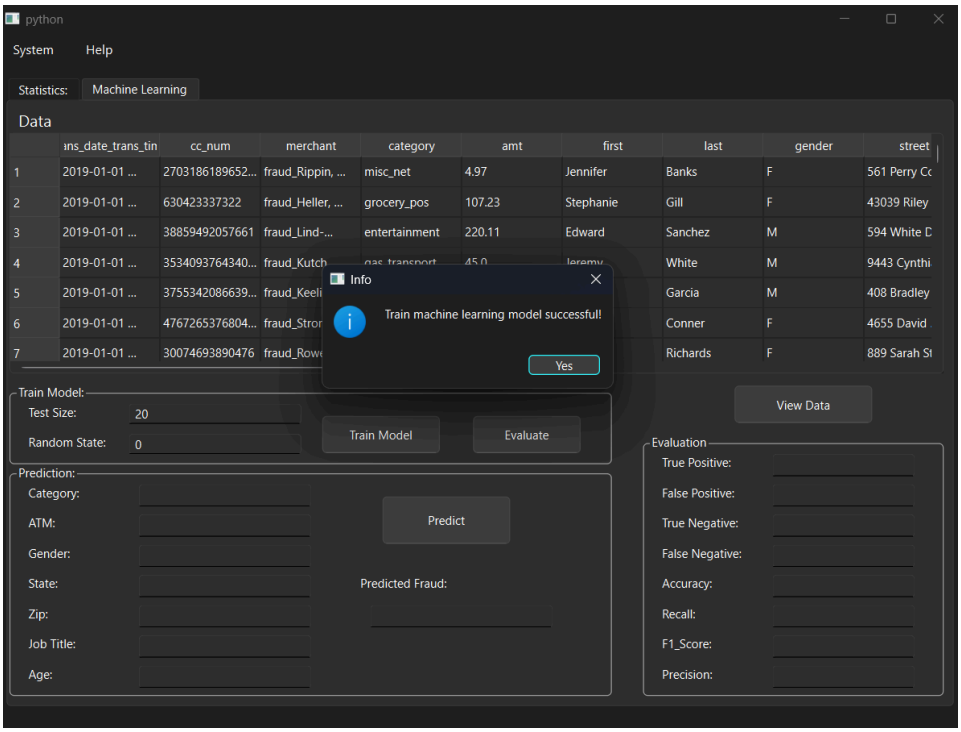


Figure 4.7. Notification "Model training has been completed (Source: Authors)

After the model has been trained, the user can click the "Evaluate Model" button to assess the model's performance. The evaluation metrics will then be displayed in the "Evaluation" section, as shown in **Figure 4.7**. These metrics provide a comprehensive view of the model's performance, helping the user evaluate the accuracy and reliability of the model in detecting fraudulent credit card transactions.

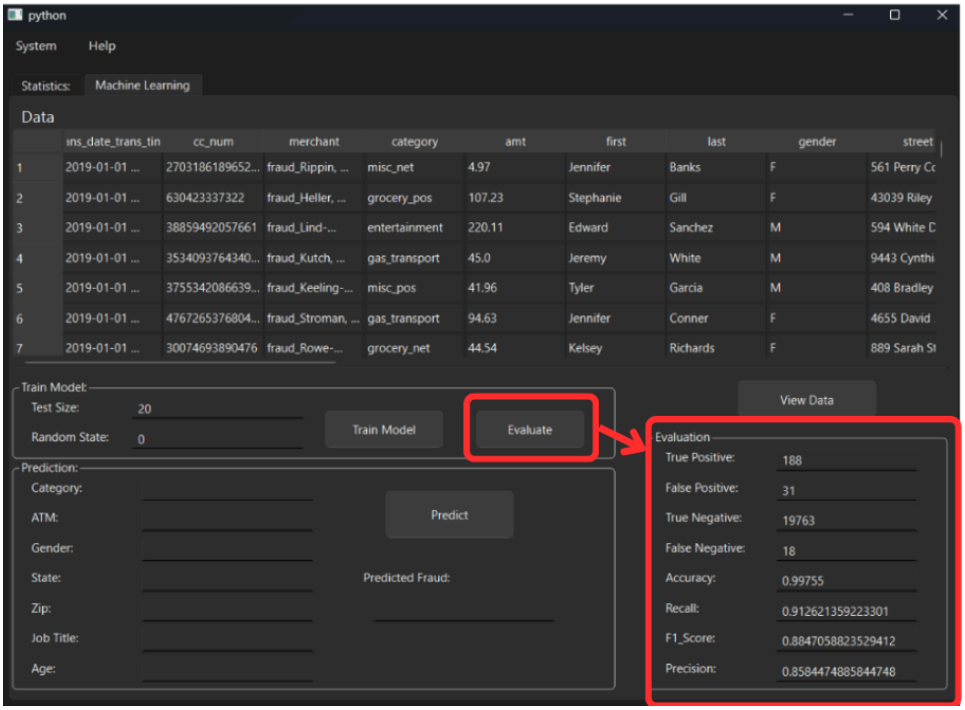


Figure 4.8. Interface for evaluating machine learning model performance (Source: Authors)

For testing purposes, we predict the outcome based on the transaction details provided for a specific customer. The result is recorded in the "Predicted Fraud" field. If the result is "1", it indicates a fraudulent transaction, whereas "0" means the transaction is legitimate. In the case shown in **Figure 4.8**, after clicking the "Predict" button, the recorded result is "0", meaning the transaction is not considered fraudulent.

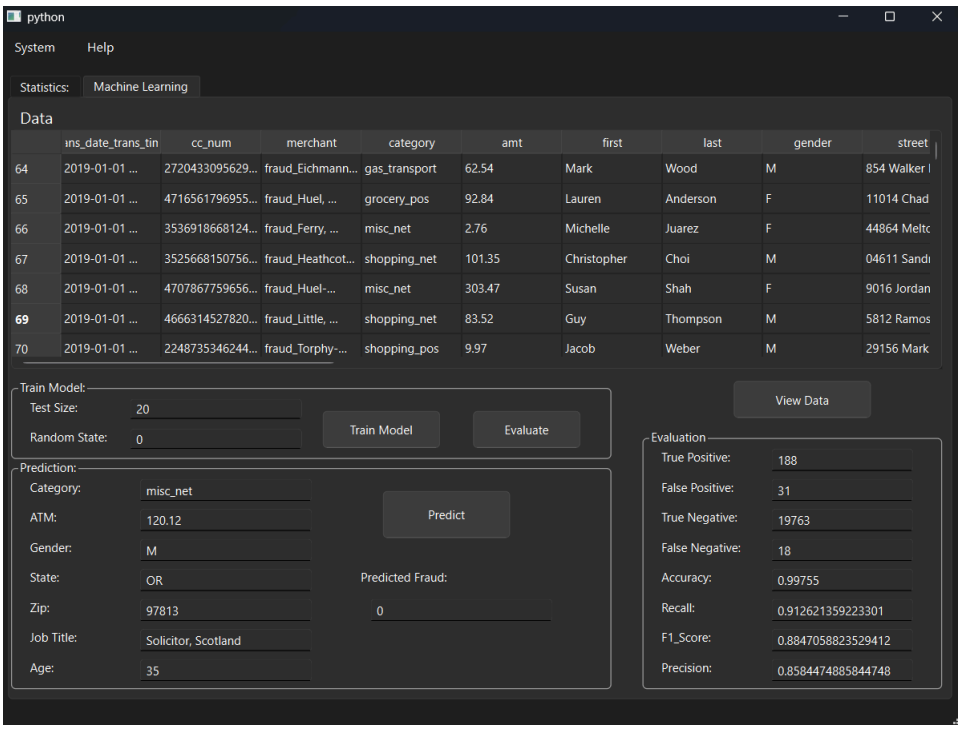


Figure 4.9. "Prediction" Interface (Source: Authors)

CHAPTER 5. DISCUSSION

Chapter 5 discusses the model developed for credit card fraud prediction, detailing the experimental process, result evaluation, and performance analysis on a real-world dataset. The results are highly promising, with an Accuracy of 0.9976, Recall of 0.8775, F1_Score of 0.8775, and Precision at 0.8775.

This chapter plays a crucial role in the development and assessment of the model, providing valuable insights into its effectiveness and applicability in fraud detection. Additionally, the team examines the impact of using SMOTE for data balancing compared to not using it. The findings indicate that while SMOTE significantly improves data balance, some performance metrics, such as Accuracy, Recall, and Precision, experience slight reductions.

5.1. Results of the model when using SMOTE

Through the research, investigation and analysis, the team has found a number of proposed algorithms and models that can be applied to predict credit card fraud. During the research, the team has tested many algorithms with different machine learning models. However, the team found that the model proposed in Chapter 3 with the LightGBM gives the most promising performance, with a Precision of 0.8775, a Recall of 0.8775 and a Precision of 0.8775.

Figure 5.1 provides a detailed analysis of the model performance. The confusion matrix shows:

- True Positives (TP): 40;
- True Negatives (TN): 4954;
- False Positives (FP): 1;
- False Negatives (FN): 5.

The LightGBM model with SMOTE performs very well, achieving high accuracy and a good balance between Precision and Recall.

The number of errors is very small (only 1 False Positive and 5 False Negative), showing that the model reduces false alarms and fraud misses.

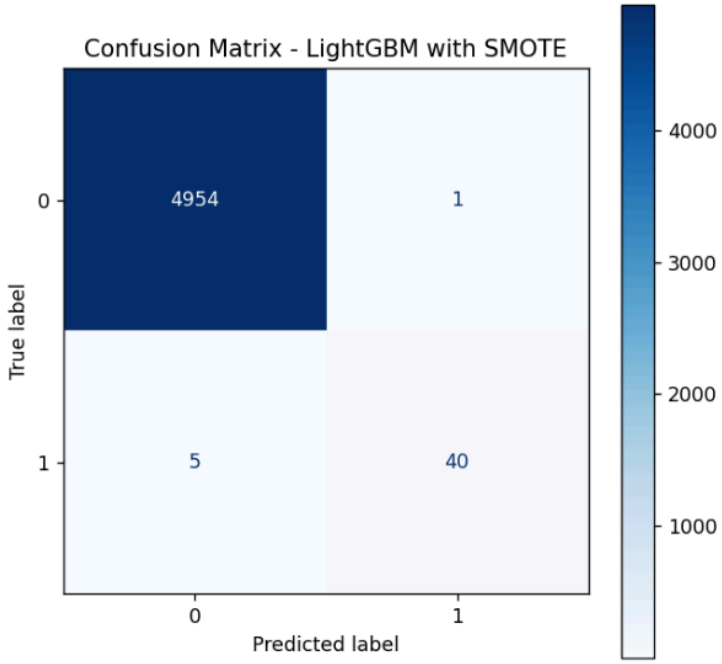


Figure 5.1. LightGBM Confusion Matrix (Source: Authors).

The LightGBM model developed for credit card fraud prediction has shown outstanding performance, with very high Accuracy, Recall, and Precision. This result demonstrates that the model is capable of effectively classifying legitimate and fraudulent transactions, which plays an important role in preventing financial risks.

Near-perfect accuracy implies that most transactions are classified correctly, significantly reducing the number of false predictions. This is especially important in the field of fraud detection, where a false positive can cause inconvenience to legitimate customers, while a false negative can lead to serious financial losses.

In addition, high recall ensures that the model successfully identifies the majority of fraudulent transactions, minimizing the risk of missing suspicious transactions. This is of great significance to financial institutions, as timely fraud detection and prevention can protect the interests of both banks and customers.

The successful application of LightGBM with the SMOTE technique to handle data imbalance demonstrates the great potential of the model in detecting fraud in practice. The low number of misclassified transactions proves that the model has good generalization ability to new data, ensuring robustness and reliability.

In summary, the LightGBM model has demonstrated impressive performance in predicting credit card fraud, with high accuracy and low error rate. These results confirm that the model can become a useful tool for financial institutions in automating and improving the efficiency of fraud detection, helping to reduce risks and protect customers from financial fraud.

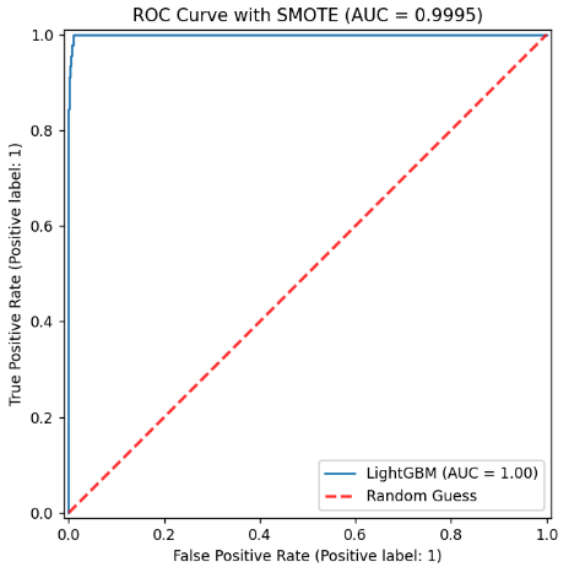


Figure 5.2. Receiver Operating Characteristic curve (Source: Authors)

Figure 5.2 shows the ROC curve of the LightGBM model with SMOTE, showing impressive performance in detecting credit card fraud. The curve is very close to the upper left corner of the graph, indicating a high True Positive Rate and a low False Positive Rate.

The area under the curve (AUC) value is 0.9995, which is almost perfect, demonstrating that the model has an excellent ability to distinguish between fraudulent and legitimate transactions. An AUC = 1 represents a perfect classifier, while an AUC =

0.5 represents a random classifier. Therefore, with a near-maximum AUC value, LightGBM significantly outperforms a random classifier.

In addition, the red dashed line in the graph represents the random guess classifier, while the blue line of LightGBM shows that the model performs very well, far from the random classifier line. This confirms that LightGBM is capable of accurately detecting fraudulent transactions, helping to reduce financial losses and improve the efficiency of the fraud detection system.

5.2. Model results without using SMOTE

The results above are the empirical results when using the SMOTE technique to balance the data. When not using SMOTE, with similar algorithms and hyperparameters, the results obtained are as follows:

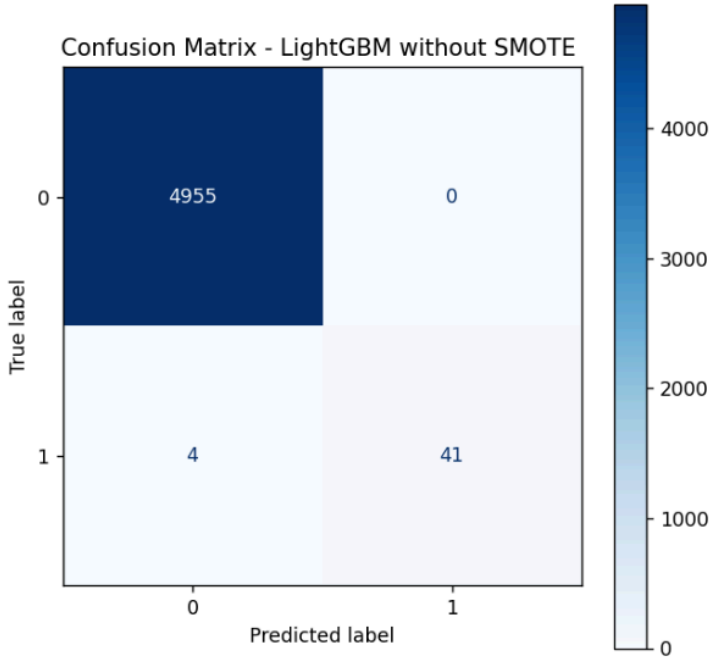


Figure 5.3. LightGBM Confusion Matrix - Don't use SMOTE (Source: Authors)

The figure above shows the confusion matrix of the LightGBM model without SMOTE, showing that the model has a very high accuracy of 99.78%. This proves that

the model has good prediction ability, with an overwhelming number of correct predictions.

Precision is 99.8%, meaning that the majority of transactions predicted by the model as fraudulent are actually correct.

Recall is 99.98%, showing that the model is able to identify most of the actual fraudulent transactions, only missing a very small number.

However, without using SMOTE to balance the data, the model may have class imbalance problems.

In summary, the model performs very well in terms of precision and recall, but the discrimination between the two classes is still not perfect. This may lead to some confusion when classifying, especially with fraudulent transactions that have similar characteristics to legitimate transactions.

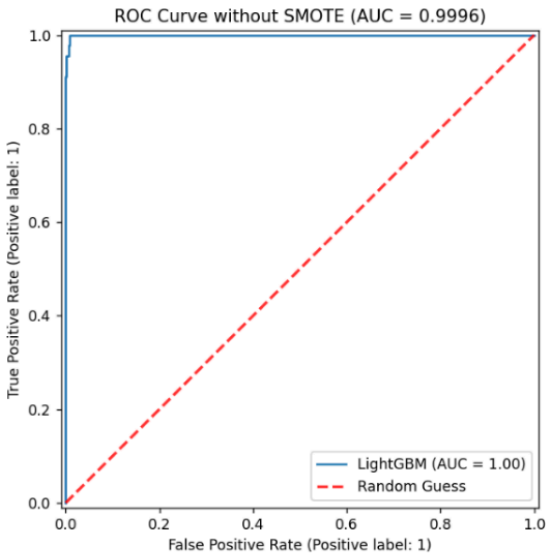


Figure 5.4. Receiver Operating Characteristic curve - Don't use SMOTE (Source: Authors)

The two ROC charts comparing the performance of the LightGBM model with and without SMOTE show that both models achieve very high performance, with an AUC close to perfect (~0.9995 - 0.9996). The ROC curves for both models are nearly touching the top-left corner, indicating a high true positive rate and a low false positive rate.

However, there are some minor differences. The model without SMOTE performs well on the original data but may be biased toward the non-fraudulent class, limiting its ability to detect fraud. Meanwhile, the model with SMOTE helps balance the data, improving fraud detection capability, but may slightly increase the false positive rate.

Overall, if the priority is high accuracy with fewer false alarms, the model without SMOTE might be the better choice. On the other hand, if improving fraud detection is the main goal, the SMOTE-applied model would be more suitable, even at the cost of some false alerts. For a more comprehensive evaluation, it would be beneficial to also consider the Precision-Recall metric to assess the trade-off between fraud detection and false alarms.

5.3. Comparison of Models with and without SMOTE

Table 5-1. Comparison of Models with and without SMOTE (Source: Authors)

Criteria	With SMOTE	Without SMOTE
Data Balance	Good	Unbalanced
Accuracy	~99.76%	~99.78%
Fraud Detection	Better	Biased
False Alarms	Higher	Lower

The table above provides a clear comparison of model performance with and without SMOTE. Using SMOTE helps address data imbalance, which is a critical issue in

fraud detection. By generating synthetic minority samples, the model learns to recognize fraudulent transactions more effectively, leading to improved fraud detection. However, this approach slightly reduces overall accuracy and increases the number of false alarms, as the model may misclassify some legitimate transactions as fraudulent.

On the other hand, the model without SMOTE achieves slightly higher accuracy because it is trained on the original dataset without synthetic samples. While this may seem like a positive outcome, it also means that the model is biased towards non-fraudulent transactions, potentially overlooking fraudulent cases. Since fraudulent transactions are rare, a model without SMOTE might classify most transactions as legitimate, leading to fewer false alarms but also missing some fraud cases.

Ultimately, the decision to use SMOTE depends on the specific goals of fraud detection. If minimizing false alarms is a priority, using the original dataset without SMOTE may be preferable. However, if the primary objective is to detect as many fraudulent transactions as possible, incorporating SMOTE can be beneficial. A balanced approach, possibly combining SMOTE with other techniques such as cost-sensitive learning or ensemble methods, may help achieve better overall performance in real-world applications.

CHAPTER 6. CONCLUSION AND FUTURE WORKS

Fraud detection in credit card transactions is a crucial challenge in the financial industry, and this study has demonstrated how machine learning can be effectively applied to address this issue. Through the implementation of various models, we have been able to analyze and classify fraudulent activities with high accuracy. This chapter summarizes our key findings and discusses areas for further research and improvement.

6.1. Conclusion

6.1.1. Advantages

After completing the project and evaluating our achievements, we have identified the following successes:

First, the use of machine learning technology in credit card fraud detection, based on LightGBM model, has enhanced the accuracy and reliability of identifying fraudulent transactions, thereby strengthening fraud prevention and risk management.

Second, the application of the Synthetic Minority Oversampling Technique (SMOTE) in conjunction with LightGBM helps balance uneven data, creating a strong and reliable predictive model.

Third, by using LightGBM combined with SMOTE, financial institutions can not only predict customer eligibility but also protect themselves against potential credit risks. This leads to smarter decisions when issuing credit cards and helps attract customers who are likely to make timely and full payments.

Finally, our software provides users with comprehensive information to have the clearest and most thorough understanding of customer data and information through the Statistic Tab. Additionally, the software features an easy-to-navigate interface and high security, as only employees with an account can log in and use it.

In conclusion, we are committed to continually improving our software to better meet the needs of our users.

6.1.2. Disadvantages

One of the most significant advantages of this study is its ability to improve fraud detection accuracy. By using multiple machine learning models, we were able to reduce the number of false positives and false negatives, ultimately making the system more reliable for financial institutions.

The integration of SMOTE also helped address data imbalance, ensuring that the models could effectively learn from minority class examples. This allowed for more comprehensive training and enhanced fraud detection capabilities.

Additionally, the scalability of machine learning-based fraud detection systems is another major advantage. These models can process large amounts of transactional data efficiently, making them well-suited for real-time fraud detection.

The ability to analyze vast datasets quickly allows financial institutions to detect fraud patterns in real-time and take immediate action.

Moreover, automating the fraud detection process reduces the reliance on manual review. Traditional fraud detection methods often require human intervention, which is time-consuming and prone to errors.

By implementing machine learning models, financial institutions can save resources and enhance security by minimizing the risk of undetected fraudulent transactions.

6.2. Future works

Future work on fraud detection systems can focus on several key areas to further enhance accuracy, efficiency, and applicability. One of the most promising directions is the integration of deep learning models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). These models can learn more complex fraud patterns and provide a higher level of accuracy compared to traditional machine learning techniques.

Another essential improvement is the optimization of real-time fraud detection by leveraging cloud-based technologies and distributed computing. Current fraud detection models often require substantial computational resources, and improving system efficiency will help reduce latency, making fraud detection more practical for high-speed financial transactions.

Moreover, the development of adaptive learning models can help fraud detection systems dynamically adjust to new fraudulent patterns without requiring frequent manual retraining. By implementing reinforcement learning techniques, models can continuously learn from incoming transaction data and improve their fraud detection capabilities over time.

Expanding datasets to include cross-border transactions can also be a valuable area for future research. Fraudulent activities often vary across different regions, and training models on more diverse data will improve their ability to detect international fraud cases more effectively.

Another crucial direction is improving model interpretability. Many advanced fraud detection models, particularly deep learning-based ones, operate as black boxes, making it challenging for financial analysts to trust and validate their predictions. Implementing Explainable AI (XAI) techniques will allow fraud detection models to provide more transparent decision-making, increasing their usability in financial institutions.

Additionally, we propose further enhancements in software implementation to ensure that fraud detection models are seamlessly integrated into practical financial applications. Some specific ideas include:

Multithreading for Real-Time Processing: Implementing multithreading in Python will allow the system to handle multiple tasks simultaneously, improving efficiency and reducing response time when processing large volumes of transactions.

Enhanced Data Visualization: Developing interactive dashboards will help financial analysts better interpret fraud detection results, enabling them to make more informed decisions.

Automated Model Updating: Creating a pipeline that automatically updates models with new fraud patterns will help improve adaptability and ensure continuous model improvement.

Integration with Financial Security Systems: Connecting fraud detection models with existing banking security infrastructures through APIs will enable seamless fraud detection deployment and automated response mechanisms.

Finally, incorporating blockchain technology for fraud prevention can be another avenue for future research. Blockchain provides a decentralized and tamper-resistant transaction ledger, which can help enhance security and detect fraudulent behavior more effectively.

REFERENCES

Al-Azzam, N., & Shatnawi, I. (2021). Comparing supervised and semi-supervised machine learning models on diagnosing breast cancer. *Annals of Medicine and Surgery, 62*, 53-64.

Aptech,, B. C. (2022). Báo cáo tổng kết thị trường thẻ Việt Nam năm 2021 và 6 tháng đầu năm 2022.

Baesens, B., Van Vlasselaer, V., & Verbeke, W. (2015). *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*: John Wiley & Sons.

Balaji, T., Annavarapu, C. S. R., & Bablani, A. (2021). Machine learning algorithms for social media analysis: A survey. *Computer Science Review, 40*, 100395.

Blagus, R., & Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. *BMC bioinformatics, 14*, 1-16.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition, 30*(7), 1145-1159.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research, 16*, 321-357.

Cioffi, R., Travaglioni, M., Piscitelli, G., Petrillo, A., & De Felice, F. (2020). Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability, 12*(2), 492.

Cruz, F., Magg, S., Weber, C., & Wermter, S. (2016). Training agents with interactive reinforcement learning and contextual affordances. *IEEE Transactions on Cognitive and Developmental Systems, 8*(4), 271-284.

Huang, C., Li, S.-X., Caraballo, C., Masoudi, F. A., Rumsfeld, J. S., Spertus, J. A., . . . Krumholz, H. M. (2021). Performance metrics for the comparative analysis of clinical risk prediction models employing machine learning. *Circulation: Cardiovascular Quality and Outcomes, 14*(10), e007526.

Jearakul, P., Wong, K. W., & Fung, C. C. (2010). *Classification of imbalanced data by combining the complementary neural network and SMOTE algorithm.* Paper presented at the Neural Information Processing. Models and Applications: 17th International Conference, ICONIP 2010, Sydney, Australia, November 22-25, 2010, Proceedings, Part II 17.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering, 160*(1), 3-24.

Lewis, F. L., & Vamvoudakis, K. G. (2010). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 41*(1), 14-25.

Liu, B., Gao, X., & Zhang, H. (2019). BioSeq-Analysis 2. 0: an updated platform for analyzing DNA, RNA and protein sequences at sequence level and residue level based on machine learning approaches. *Nucleic acids research, 47*(20), e127-e127.

Lộc, Đ. (2019). Nguồn cơn ra đời 'lạ lùng' của chiếc thẻ cả thế giới đang dùng. Accessed May 5, 2024.

Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics, 1*(3), 283-289.

Naeem, S., Ali, A., Anam, S., & Ahmed, M. M. (2023). An unsupervised machine learning algorithm: Comprehensive review. *International Journal of Computing and Digital Systems*.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology, 24*(12), 1565-1567.

Oduor, Z. A. (2007). *A Survey of Credit Card Risk Assessment Practices in Kenya Banks*. University of Nairobi,

Saeeda, U., Janb, S. U., Ahmadb, J., Shaha, S. A., Alshehric, M. S., Ghadid, Y. Y., . . . Buchanan, W. J. Generative Adversarial Networks-enabled Anomaly Detection Systems: A Survey.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning, 109*(2), 373-440.

Vujović, Ž. (2021). Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications, 12*(6), 599-606.

Zeineddine, H., Braendle, U., & Farah, A. (2021). Enhancing prediction of student success: Automated machine learning approach. *Computers & Electrical Engineering, 89*, 106903.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

Nguyễn, T. H., & Lê, V. K. (2018). Phương pháp học máy trong phát hiện gian lận thẻ tín dụng. Tạp chí Kinh tế và Phát triển, 256(II), 120-128.

Nguyễn, V. H., & Trần, Q. A. (2021). Đánh giá một số thuật toán học máy không giám sát sử dụng trong phát hiện gian lận thẻ tín dụng. Tạp chí Ngân hàng, 9, 16-25.

Đặng, N. H., Phạm, T. H. D., & Cao, T. N. (2022). Nghiên cứu gian lận báo cáo tài chính tiếp cận theo thuật toán rừng ngẫu nhiên. Tạp chí Khoa học và Công nghệ, 60(3), 45-55.

Nguyễn, M. C., & Trần, D. T. (2023). Phát hiện gian lận thẻ tín dụng sử dụng mô hình học sâu. Tạp chí Kinh tế - Luật và Ngân hàng, 5(2), 100-115.