

Louisiana Tech University

## Louisiana Tech Digital Commons

---

Doctoral Dissertations

Graduate School

---

Summer 8-2021

### **Credit Card Security System and Fraud Detection Algorithm**

Baker Al-Smadi

Follow this and additional works at: <https://digitalcommons.latech.edu/dissertations>

---

**CREDIT CARD SECURITY SYSTEM AND FRAUD  
DETECTION ALGORITHM**

by

Baker Al-Smadi, B.S.C., M.S.

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy: Engineering

COLLEGE OF ENGINEERING AND SCIENCE  
LOUISIANA TECH UNIVERSITY

August 2021

LOUISIANA TECH UNIVERSITY

GRADUATE SCHOOL

March 23, 2021

Date of dissertation defense

We hereby recommend that the dissertation prepared by

**Baker Al-Smadi, B.S.C., M.S.**

entitled **Credit Card Security System and Fraud Detection Algorithm**

be accepted in partial fulfillment of the requirements for the degree of

**Doctor of Philosophy in Engineering, Cyberspace Conc.**



Manki Min

Supervisor of Dissertation Research



Galen Turner

Head of Engineering

**Doctoral Committee Members:**

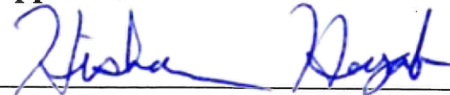
Galen Turner

Jinyuan Chen

Ibrahim Abdoulahi

Weizhong Dai

**Approved:**



Hisham Hegab

Dean of Engineering & Science

**Approved:**



Ramu Ramachandran

Dean of the Graduate School

## **ABSTRACT**

Credit card fraud is one of the most critical threats affecting individuals and companies worldwide, particularly with the growing number of financial transactions involving credit cards every day. The most common threats are likely to come from database breaches and identity theft. All these threats threaten put the security of financial transactions at severe risk and require a fundamental solution.

This dissertation aims to suggest a secure online payment system that significantly improves credit card security. Our system can be particularly resilient to potential cyber-attacks, unauthorized users, man-in-the-middle, and guessing attacks for credit card number generation or illegal financial activities by utilizing a secure communication channel between the cardholder and server. Our system uses a shared secret and a verification token that allow both sides to communicate through encrypted information. Furthermore, our system is designed to generate a one-time credit card number at the user's machine that is verified by the server without sharing the credit card number over the network. Our approach combines the machine learning (ML) algorithms with unique temporary credit card numbers in one integrated system, which is the first approach in the online credit card protection system. The new security system generates a one-time-use credit card number for each transaction with a predetermined amount of money. Simultaneously, the system can detect potential fraud utilizing ML algorithm

with new critical features such as the IMEI or I.P. address, the transaction's location, and other features.

The contribution of this research is two-fold: (1) a method is proposed to generate a unique, authenticatable one-time credit card number to effectively defend against the database breaches, and (2) a credit card fraud prevention system is proposed with multiple security layers that are achieved by the integration of authentication, ML-based fraud detection, and the one-time credit card number generation.

The dissertation improves consumers' trust and confidence in the credit card system's security and enhances satisfaction with credit cards' various financial transactions. Further, the system uses the current online credit card infrastructure; hence it can be implemented without tangible infrastructure cost.

## **APPROVAL FOR SCHOLARLY DISSEMINATION**

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that “proper request” consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author \_\_\_\_\_

Date \_\_\_\_\_

## **DEDICATION**

This dissertation is dedicated to my father, Prof. Sami Al-Smadi, and my mother, Asma Al-Smadi, who supported me in my career with their endless love and encouragement.

## TABLE OF CONTENTS

ABSTRACT.....	iii
APPROVAL FOR SCHOLARLY DISSEMINATION .....	iv
DEDICATION.....	vi
LIST OF TABLES.....	xi
LIST OF FIGURES .....	xii
ACKNOWLEDGMENTS .....	xiii
CHAPTER 1 INTRODUCTION .....	1
1.1 Credit Card Fraud: Statistics.....	2
1.2 Types of Financial Fraud .....	6
1.2.1 Credit Card Fraud .....	6
1.2.2 Telecommunication Fraud .....	7
1.2.3 Computer Intrusion .....	8
1.2.4 Bankruptcy Fraud.....	8
1.2.5 Theft Fraud.....	9
1.2.6 Application Fraud .....	9
CHAPTER 2 LITERATURE BACKGROUND .....	11
2.1 Related Work .....	11
2.2 Related Work of Credit Card Fraud Detection Techniques.....	16
2.2.1 Support Vector Machine (SVM).....	17



2.2.2 Naïve Bayes .....	18
2.2.3 Decision Tree .....	20
2.2.4 Logistic Regression.....	21
2.2.5 Random Forest.....	22
2.3 Other Related Work of ML Algorithms.....	23
2.4 Assessment of the Related Work .....	24
2.5 Conclusion .....	28
<b>CHAPTER 3 THE PROPOSED SYSTEM .....</b>	<b>30</b>
3.1 Traditional Online Credit Card Payment System .....	30
3.2 The Proposed System.....	31
3.2.1 Server’s Side .....	32
3.2.2 Description of Proposed System.....	33
3.2.3 Deployment Overview.....	35
3.3 Generation of the One-time Credit Card Number.....	40
3.4 Machine Learning Fraud Detection Component.....	42
3.5 User Interface.....	45
<b>CHAPTER 4 EXPERIMENTS OF ML ALGORITHMS .....</b>	<b>46</b>
4.1 Datasets.....	46
4.1.1 Real Dataset .....	46
4.1.2 Artificially Generated Datasets.....	46
4.2 Research Methodology .....	56
4.2.1 Machine Learning Algorithms.....	57
4.2.2 Machine Learning Criteria.....	57

4.3 ML Experiment on the Real Dataset (1 <sup>st</sup> Experiment).....	59
4.4 Machine Learning Algorithms Experiment on the Artificial Datasets (2 <sup>nd</sup> Experiment).....	62
4.5 First Experiment Vs. the Second Experiment.....	64
4.5.1 Performance Comparison (1 <sup>st</sup> Experiment vs. 2 <sup>nd</sup> Experiment).....	64
4.5.2 Time Efficiency Comparison (1 <sup>st</sup> Experiment vs. 2 <sup>nd</sup> Experiment) .....	66
4.6 Excluding Features (3 <sup>rd</sup> Experiment).....	67
CHAPTER 5 DISCUSSION.....	70
5.1 Security Analysis .....	70
5.1.1 Defend Against Potential Breaches .....	71
5.1.2 Authentication.....	73
5.2 Overview on ML Algorithms' Results .....	73
5.3 One-Time Credit Card Number Analysis .....	75
5.3.1 One-Time Credit Card Number Generation .....	78
5.4 Conclusions.....	78
CHAPTER 6 CONCLUSIONS AND FUTURE WORK.....	81
APPENDIX A REAL DATASET ANALYSIS AND RESULTS .....	83
A.1 Real Dataset .....	84
A.2 Artificial Datasets .....	87
A.2.1 Statistical Analysis of Datasets.....	87
A.3 Machine Learning Algorithms Experiment on the Real Dataset (1 <sup>st</sup> Experiment) .....	95
A.3.1 Decision Tree (1 <sup>st</sup> Experiment).....	95
A.3.2 Support Vector Machine (SVM) (1 <sup>st</sup> experiment) .....	97

A.3.3 Random Forest (1 <sup>st</sup> experiment).....	101
A.3.4 Logistic Regression (1 <sup>st</sup> experiment) .....	102
A.3.5 Naïve Bayes (1 <sup>st</sup> experiment).....	104
A.4 Machine learning algorithms experiment on the artificial datasets (2 <sup>nd</sup> Experiment).....	105
A.4.1 Decision Tree (2 <sup>nd</sup> experiment).....	106
A.4.2 Random Forest (2 <sup>nd</sup> experiment) .....	106
A.4.3 Logistic Regression (2 <sup>nd</sup> experiment).....	108
A.4.4 Time efficiency of the 2 <sup>nd</sup> experiment.....	109
A.5 Artificial Datasets Generation Code .....	110
A.6 One-Time Credit Card Generation Full Code.....	112
REFERENCES .....	116

## LIST OF TABLES

<b>Table 1-1:</b> <i>Worst Data Breaches in History Due to Credit Card Frauds [8]</i> .....	4
<b>Table 2-1:</b> <i>Advantages and Disadvantages of Fraud Detection Techniques</i> .....	25
<b>Table 4-1:</b> <i>Confusion Matrix (Sample)</i> .....	58
<b>Table 4-2:</b> <i>Performance Comparison</i> .....	61
<b>Table 4-3:</b> <i>Time Efficiency</i> .....	62
<b>Table 4-4:</b> <i>Average Performance of the 2<sup>nd</sup> Experiment</i> .....	63
<b>Table 4-5:</b> <i>Average Time Efficiency of the 2<sup>nd</sup> Experiment</i> .....	63
<b>Table 4-6:</b> <i>Performance Comparison (1<sup>st</sup> Experience Vs. 2<sup>nd</sup> Experience)</i> .....	65
<b>Table 4-7:</b> <i>Time Efficiency (1<sup>st</sup> Experience Vs. 2<sup>nd</sup> Experience)</i> .....	66
<b>Table 4-8:</b> <i>Excluding Location (3<sup>rd</sup> Experiment)</i> .....	68
<b>Table 4-9:</b> <i>One Feature ML Integration</i> .....	69

## LIST OF FIGURES

<b>Figure 1-1:</b> Credit card fraud reports in the United States [7][35] .....	3
<b>Figure 1-2:</b> Money lost due to fraud by method of contact [7].....	3
<b>Figure 1-3:</b> Identity theft fraud reports [7][31] .....	5
<b>Figure 1-4:</b> Identity theft cases in the United States [7][35].....	6
<b>Figure 2-1:</b> Decision tree .....	20
<b>Figure 3-1:</b> The stages of regular (current) online transaction. ....	30
<b>Figure 3-2:</b> The proposed System (online transaction).....	34
<b>Figure 3-3:</b> System overview.....	36
<b>Figure 3-4:</b> One-time number generation .....	37
<b>Figure 3-5:</b> ML integration. ....	38
<b>Figure 3-6:</b> User's interface. ....	45
<b>Figure 4-1:</b> Time gap .....	52
<b>Figure 4-2:</b> Transaction amount histogram.....	53
<b>Figure 4-3:</b> Transaction store histogram.....	54
<b>Figure 4-4:</b> Transaction store boxplot.....	55
<b>Figure 4-5:</b> Time/h boxplot.....	56
<b>Figure 4-6:</b> ROC comparison.....	60

## **ACKNOWLEDGMENTS**

I would like to express my sincere gratitude to my advisor, Dr. Manki Min, for my Ph.D. study and research's continuous support for his patience, motivation, enthusiasm, and extensive knowledge. His guidance helped me in my research and writing this dissertation. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my advising committee members: Dr. Galen Turner, Dr. Jinyuan Chen, Dr. Weizhong Dai, and Dr. Ibrahim Abdoulahi for their encouragement, insightful comments, and guidance.

# **CHAPTER 1**

## **INTRODUCTION**

Today, the credit card system is widely used to settle payments in modern economies and facilitate business transactions worldwide. Given the popularity of the credit card system, it became a target for cyberattacks and fraud worldwide. This calls for a more secure approach to avoid potential breaches and unauthorized users. In particular, the most recognized credit card threats often come from database breaches and identity theft issues. Generally, the credit card system looks vulnerable to various risks, hence the pressing need for a more secure financial transaction worldwide.

Historically, the Diners' Club Inc. introduced the first universal credit card in 1950, followed by another powerful system of this type, known as a Travel and Entertainment card by the American Express Company in 1958. In the same vein, the Bank of America in California introduced the first national plan, called BankAmerica card in 1958, and licensed in 1966, later renamed VISA in 1976 [1].

Currently, the credit card system suffers from many cyber-attacks every second worldwide. The biggest cybercrime in the history of the credit card system happened in July 2019 through the database of the Capital One Bank [2]. 106 million accounts' information was stolen, 100 million were from the United States, and the rest were from Canada. According to Reuter's sources, the approximate cost was estimated up to \$150 million [3]. The second biggest cybercrime took place in January 2009 on The Heartland

company payment system [4]. The data and personal identification of 160 million credit cards were stolen, with an approximate cost of \$140 million.

According to The Nilson Report [6], card fraud loss worldwide was \$23.97 billion in 2017, increased to \$27.85 billion in 2018, and is projected to reach \$40.63 billion in 10 years [3]. The United States accounts for 22.19% of the total volume worldwide of financial card fraud in 2019, making 33.57% of gross losses worldwide. Financial card fraud reached \$19.03 billion for all other countries in 2019, which equaled 5.79¢ per \$100 in total volume [31].

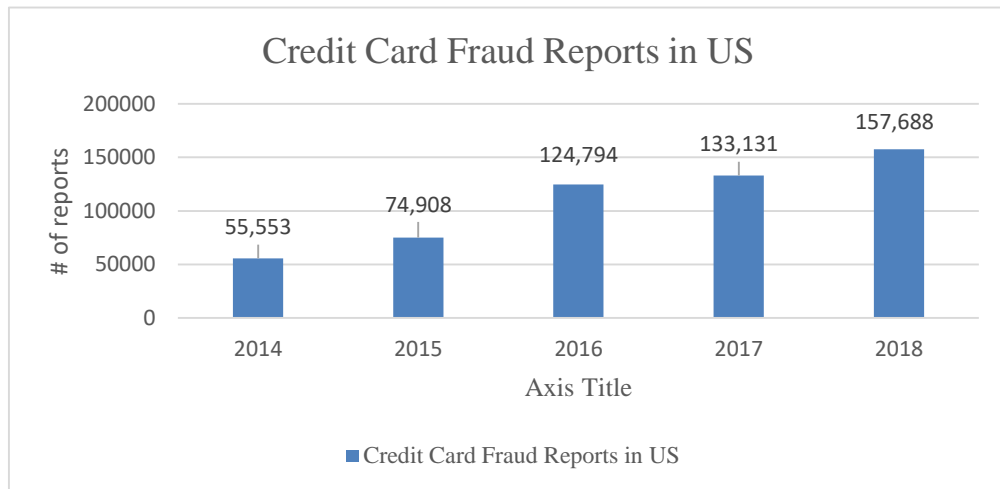
The massive financial losses through credit/debit cards are climbing every year and may contribute to a potential financial crisis in the world. Hence, this situation's urgency calls for a more secure financial transaction system. In this research, we will rely on the Secure Socket Layer (SSL) for the connections through webpages and will focus mainly on the processing of information to defend against online fraud. This research provides a secure system to help individuals and companies conduct their business transactions with trust and confidence.

### **1.1 Credit Card Fraud: Statistics**

Although many proposed systems attempted to improve the credit card system's security over the past five decades, the amount of lost money and the number of reported cybercrimes are increasing dramatically over time. As cyberattacks badly victimized many individuals and companies, tremendous efforts were made to improve the credit card system's security to make it more reliable and secure. A survey was conducted on consumer feedback by the Identity Theft Resource Center' in 2018 and updated in 2019 [7]. This survey shows that 85.71% of the victims felt worried, angry, and frustrated,

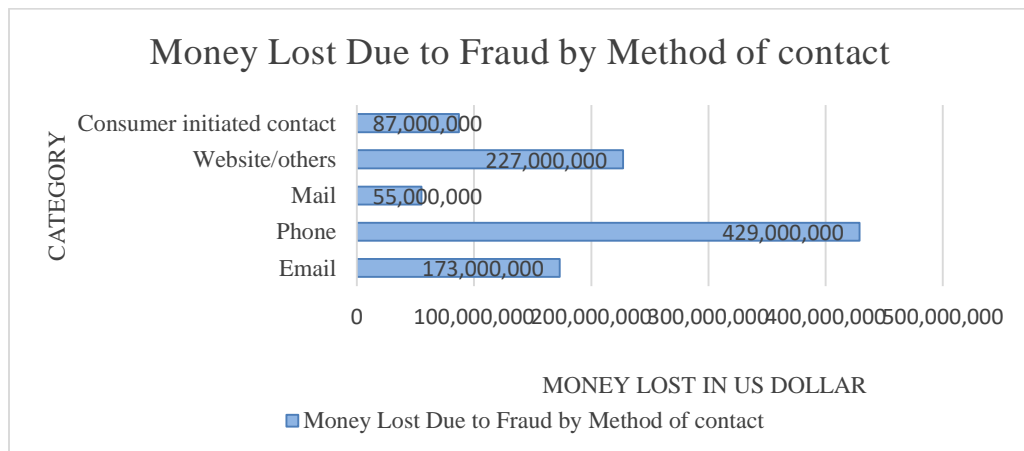


while 83.7% felt violated, and 69.4% could not trust others and felt unsafe. The study revealed that credit card fraud was significantly growing over time, as illustrated in Figure 1-1. The credit card fraud issue is serious in the United States' financial system, and potentially in other places in the world worsened.



**Figure 1-1:** Credit card fraud reports in the United States [7][35]

Further, the study shows that money lost due to fraud by the method of contact and data breaches by the business sector in 2018, as illustrated in Figure 1-2.



**Figure 1-2:** Money lost due to fraud by method of contact [7]

Clearly, most identity theft and credit card fraud were done by phone or through websites (as shown in Figure 1-2). Thus, the growing need to find a suitable solution that eliminates these cybercrimes strongly motivates this dissertation's work.

John S Kiernan's report summarized the most severe credit card data breaches in the period 2005-2014 in Table 1-1 [8]. Figures in the table show that millions of accounts were affected by database breaches.

**Table 1-1: Worst Data Breaches in History Due to Credit Card Frauds [8]**

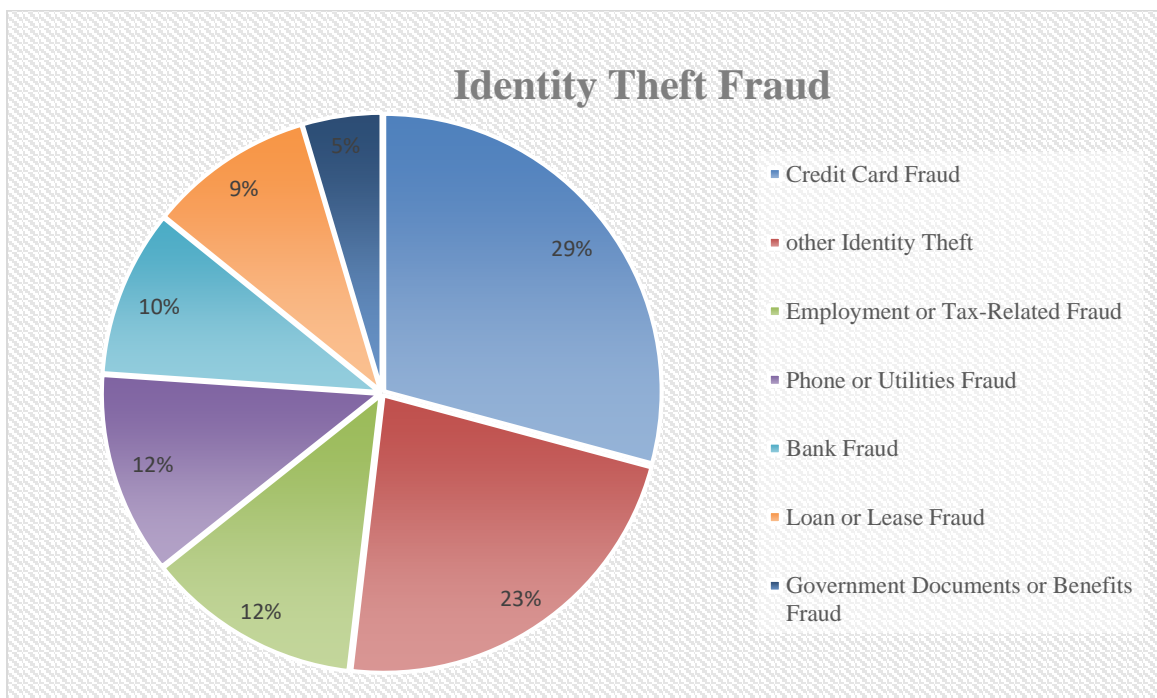
<b>COMPANY</b>	<b>YEAR</b>	<b>NUMBER OF ACCOUNTS AFFECTED</b>
<b>CARD SYSTEM SOLUTION.</b>	2005	40 Million
<b>TJX COMPANIES, INC.</b>	2006	94 Million
<b>US VETERANS' AFFAIRS</b>	2006	17.5 Million
<b>CERTEGY</b>	2007	8.5 Million
<b>FIDELITY NATIONAL INFORMATION SERVICES</b>	2007	3.5 Million
<b>HEARTLAND PAYMENT SYSTEM</b>	2008	134 Million
<b>BANK OF NEW YORK MELLON</b>	2008	12.5 Million
<b>HANNAFORD BROS. SUPERMARKET CHAIN</b>	2008	4.2 Million
<b>SONY</b>	2011	12 Million
<b>GLOBAL PAYMENTS</b>	2012	1 Million
<b>TARGET</b>	2013	40 Million
<b>HOME DEPOT</b>	2014	56 Million

The vast numbers in Table 1-1 show a severe need to improve the credit card security systems to ensure high security and a safe environment for the customers.

According to this report [8], identity theft cases in the U.S. have increased from 444,358

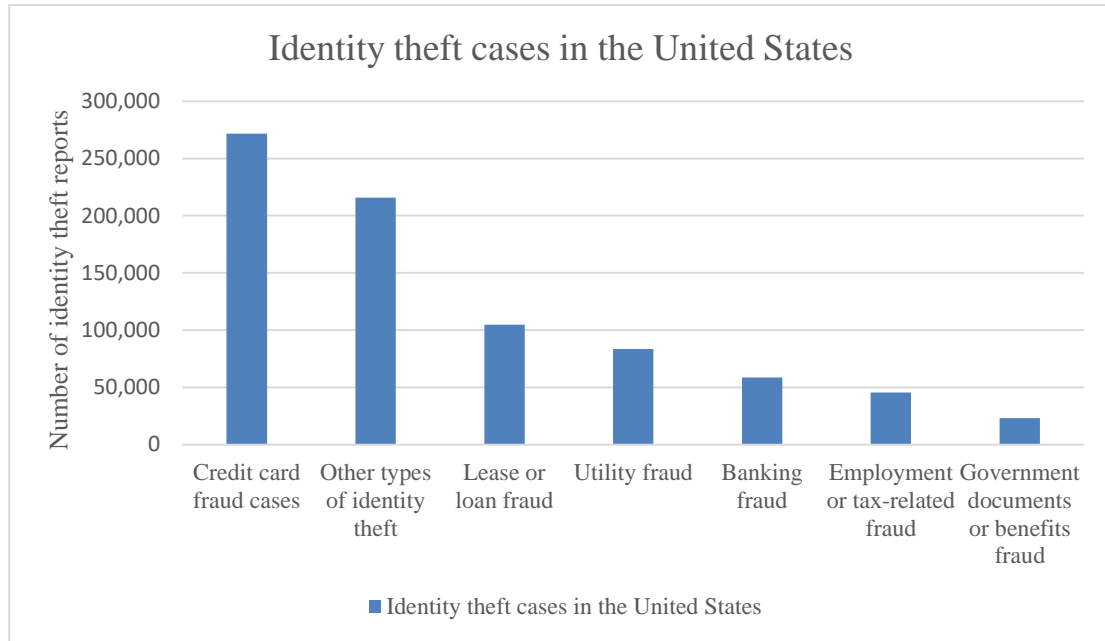
in 2018 to 650,572 in 2019. Credit card fraud is considered one of the most common types of identity theft as of January 2020.

Another type of threat is identity theft crimes. According to a study revealed by the Shift Credit Card Processing website in 2018 and updated in 2020 [7], credit card fraud was the most common identity theft. It accounted for 29% of all identity theft reporting in 2018, as shown in Figure 1-3.



**Figure 1-3:** Identity theft fraud reports [7][31]

Further, the study revealed the identity theft fraud reports in the United States in 2018, as illustrated in Figure 1-4.



**Figure 1-4:** Identity theft cases in the United States [7][35]

Credit card fraud cases have doubled in the last year and are even projected to increase. The stolen funds affect both consumers and businesses as they can deaden the business operations and cause severe damage to these financial systems. Unauthorized financial operations can make it difficult for a business to obtain regular payments and even lead companies into bankruptcy.

## 1.2 Types of Financial Fraud

Various types of fraud can be recognized [9], including credit card fraud, telecommunication fraud, computer intrusion, bankruptcy fraud, theft/counterfeit fraud, and application fraud.

### 1.2.1 Credit Card Fraud

Credit card fraud consists of two parts: Offline and Online. Offline fraud happens when a thief uses a stolen physical credit card at a point-of-sale or any other place. In

contrast, Online fraud occurs when a thief uses the stolen information from someone's credit card to commit fraud through the internet.

### **1.2.2 Telecommunication Fraud**

The telecommunication industry has grown dramatically worldwide in the last decade, particularly with several technologies using phones [10]. With this extensive use of phone technology, global mobile phone fraud is increasing accordingly. This global problem causes significant annual losses for many companies, businesses, and communication service providers.

Telecommunication fraud is considered the simplest, with the lowest risk for fraudsters to make money illegally. Telecommunication fraud has two types: subscription fraud and superimposed fraud. Subscription fraud happens when fraudsters obtain an account without the intention to pay the bill, which performs at the level of a phone number. Hence, all transactions made by the received phone number will be fraudulent. These accounts are most likely used for call selling by committing fraud through phone calls or any other criminal practices. In the meantime, superimposed fraud involves stealing a legitimate account. In this case, the unusual usage is superimposed on top of the routine use of genuine customers. An example of superimposed fraud is cellular cloning. Furthermore, telecommunication fraud might occur from an insider when an employee sells information to a fraudulent attacker for illegal purposes.

Our proposed system should protect the credit card users in both telecommunication fraud types since our system focuses on detecting unusual payment activity through a machine learning algorithm and stopping any possible fraudulent transaction. Moreover, the proposed approach uses the one-time credit card number for

every online transaction. Thus, even the insiders would not be able to sell the users' information to anybody because these cards' information would be useless.

### **1.2.3 Computer Intrusion**

Computer intrusion is defined as a cyber-crime that includes hacking into private computers and cell phones or other electronic devices, manipulating, or stealing information for illegal purposes [11]. In other words, it is an unauthorized attempt to access information. This type of crime typically targets individuals. Computer intrusion is often committed by an insider who knows the infrastructure and the design of the system. The intruder can also be an outsider (hacker). The proposed system also comes to address this kind of fraud by determining the source (location) of each payment transaction so the system can ban suspicious transactions. On the other hand, our approach has another protection level by utilizing the one-time credit card number.

### **1.2.4 Bankruptcy Fraud**

Bankruptcy is intended to give an individual or a company a chance to rearrange and settle their financial issues [12]. Bankruptcy fraud here means using a credit card by someone who has a prior intention to declare a state of bankruptcy. Bankruptcy fraud is counted as one of the most complex scams to predict because the credit card issuer does not know precisely their customer's financial status, so they will have to cover the losses themselves in case of any customer's bankruptcy. One solution to stop this kind of fraud is by checking with the credit bureau to know the customers' financial history. The credit bureau [13] helps banks and credit card issuers investigate the financial history of applicants who want to have a credit card.

### **1.2.5 Theft Fraud**

Theft fraud here means using a credit/debit card that is not yours. The fraudster steals a victim's credit card and attempts to use it as many times as possible before the actual customer has reported suspicious transactions on his/her account and required the card issuer to block the card. The sooner the reporting by the victimized customer, the faster the response of the bank.

These kinds of fraud can also be addressed through our system. Our approach offers a virtual credit card using mobile and computer applications. Therefore, users don't need to have a physical card for online transactions to prevent theft fraud.

### **1.2.6 Application Fraud**

Application fraud occurs when someone applies for a credit card with incorrect information [14]. A study carried by Phua and others examined more than 300 million fraudulent account applications and concluded that 88% of those fraudulent accounts were opened using identity fraud techniques [15].

Two different situations of application fraud can be recognized: Duplicate application and identity fraudsters. Duplicate application happens when applications come from identical users with identical information. In this case, cross-matching techniques are used to detect such duplication by generating a suspicious numeric score on credit card applications based on implicit links to each other in real-time. Identity fraud occurs when applications come from different individuals with similar features. However, most banks require applicants to fill out a form with specific information to confirm credit card eligibility. This information contains identification, address, contact number, personal details, and some other relevant information. Most of the required

information is used for searching for duplicates and identification purposes. In general, ML algorithms can avoid this kind of fraud since ML can be used to detect such malicious applications in the future.



## **CHAPTER 2**

### **LITERATURE BACKGROUND**

#### **2.1 Related Work**

Many research studies have attempted to minimize credit card fraud worldwide, including fraud detection algorithms, by monitoring users' behavior to eliminate suspicious credit card transactions. Saxena and Ponnappalli [16] pioneered a research study to reduce credit card fraud by producing a "one-time credit card number." The study proposed a system that generates a one-time credit card number at the user's side offline without contacting the server or being online. The system uses a shared key that generates a credit card number. Transaction details can be signed in with a private key for each specific customer, providing non-repudiation of the online transaction. The authors use the current credit card numbering structure to continue processing online transactions with conventional infrastructure.

Meredith et al. [17] conducted a project in which they designed a method to detect any credit card fraud via mobile device location tracking. This project included a processor that calculated a fraud percentage through tracking the user's device associated with the credit card account with the first area location where the user sets up his/her credit card account. Rajasekaran and Varadarajan [18] developed a new model to reduce the potential credit card fraud by a one-time credit card number generator and single round-trip authentication. This model generates a one-time credit card number at the

user's device and sends the generated number to the card issuer and the merchant. The card issuer applies some authentication criteria in order to verify the user's identity, such as a one-way password or a string of letters that the user should use to verify the transaction. There is a shared key between the user and server to move forward, and the server accepts the generated one-time credit card number on the user's device and matches the generated number with the user's number sent to the merchant. The user is granted an authentication if the numbers are matched.

In 2013, Lynam et al. [19] invented a system called "System and Method for Authenticating Payment Transactions." This system is designed to find potential fraud based on the I.P. address or the IMEI number of the device where the transaction comes from. The system stores the I.P. address or the IMEI number first associated with the genuine cardholder when the credit/ debit card is activated. Then the system matches the stored IP/IMEI number in the history of the previous payment transactions with the upcoming transactions. If the system finds a match, the transaction will be authenticated and authorized; otherwise, the system sends a message to the cardholder indicating a suspicious transaction.

Essebag et al. [23] developed a model using a comprehensive dynamic security code system. The system can change the security code CVV (Card Verification Value) of a prepaid, debit, or credit card. The system provides dynamic security code values that have limited use to online transactions only and can be calculated by the dynamic security code generator and used within existing payment infrastructures. The system can also be used in different environments not related to payments, such as balance inquiries.

Ashfield [24] developed a method and apparatus for using at least a portion of a one-time password as a dynamic CVV. A credit/debit card can generate a dynamic CVV. The user can be authorized according to a dynamic CVV by getting a transaction authorization request for a specific credit/debit card, wherein the transaction authorization request includes a dynamic CVV. The dynamic CVV is compared to at least a portion of a one-time password generated for the specific credit/debit card. A one-time password is used for logging in to the system for every online transaction. A transaction authorization can be sent to the merchant when the dynamic CVV matches all or a portion of the one-time password.

Patel [25] developed a dynamic CVV temporary task system that increases credit/debit card security or other similar financial apparatus security. The dynamic CVV is read, modified, and rewritten to the card with each online transaction. The proposed system provides a static CVV to facilitate online shopping. Alternatively, the static CVV can be used to evoke a user when the user cannot remember an unmarked static, such as reading the digits in an order requested by a user, like a PIN number.

McDonald [26] addressed the credit card security issue in a different way by a system for cardless secure online purchasing using a credit/debit card. The study was based on the presented online purchaser performing the online purchase and at least one online credit/debit card service provider holding an online purchaser interface. An e-authentication and credential service provider should have an online purchaser interface. The system authenticates users using a Personal Digital Identity Token or (PDIT). The PDIT is biometric of the cardholder with a means that provides a link to a set of proven civil identity credentials. The system maintains at least one online credit/debit card

service provider and provides a means for secure online transactions. It provides anonymity to the online purchase by covering credit/debit card data during the online purchase, making the purchase invisible to identity thieves and hackers.

Barbour [27] invented a system to execute one or more financial transactions over a communication route for a cardholder holding an account linked with a permanent account number. The card count number is deactivated for financial transactions over a communication path. A single-use number associated with the permanent account number is issued, and funds are approved for transfer using the single-use number extracted from the account holder's account.

The single-use number is activated after the cardholder inquiries about the activation of the single-use number. Funds are then transferred from the account in response to the account holder's authorization, using a single-use account number. The single-use number is then deactivated after accomplishing the transfer of funds.

Gupta and Johari [28] published a paper that reviews and analyzes the current progress in online authentication procedures, including biometrics, one-time-password systems, mobile devices, and Public Switched Telephone Network for cardholder authentication. The authors propose an entirely new framework for both onsite and online (online shopping) credit card transactions.

Yingjiu and Zhang [29] used a hash function in the generation of one-time credit card numbers. The next one-time number is computed by hashing the current one-time number with a secret known only by the cardholder and issuer. This system uses a small chip that is embedded into each credit card for hash computations and for storage of a

past credit card token (CCT). Their proposed scheme places less overhead on credit card issuers and can be easily used in both online and offline payment scenarios.

Trivedi, Kumar, et al. [30] introduced a credit card fraud detection mechanism, including a feedback system, dependent on machine learning algorithms. The feedback approach contributes to enhancing the classifier's detection rate. The authors examined the performance of different ML methodologies--Random Forest, Decision Tree, Artificial Neural Networks (ANN), SVM, Naïve Bayes, machine, Logistic Regression, and Gradient Boosting Classifier Strategies—on a slightly skewed credit card fraud dataset. They showed that Random Forest has better results with 95% precision compared to other machine learning classifiers. However, R.F. is considered a time-consuming model.

Gupta, Shalini, and Johari [66] discussed and analyzed the current online authentication procedures, including biometrics, one-time-password (OTP) systems, mobile devices, and Public Switched Telephone Network for cardholder authentication.

The one-time-password (OTP) system involves single-use one-time passwords for user authentication. When the cardholder's credit card information is passed to the payment gateway for authentication, the gateway sends a one-time password to the cardholder, either on his mobile device or to his e-mail address. The merchant then urges the cardholder to enter that one-time password specific to the transaction on his device or website in case of online shopping. If the one-time password is entered successfully, the cardholder's authenticity is proved, and the transaction is completed. This approach causes a high overhead on the cardholder since he/she is required to enter the one-time password every online transaction for authentication.

Some popular credit card issuers offer virtual credit card numbers. For example, Capital One's Bank provides this service for all its customers. This service must go through a third-party Eno, an "intelligent assistant" that provides Capital One customers with various tasks. To generate a virtual credit card number through Eno, you need to be on a computer and have the Eno extension for Google Chrome or Mozilla Firefox. Capital One's customers are not able to use this service using any other internet browser. Eno does not have customers' payment history, so they are only responsible for providing the virtual credit card number service to Capital One's customers. Thus, they will not be able to detect fraudulent transactions based on the customer's payment history. The Eno system assigns a credit card number to each merchant for future transactions [44].

Another example of a credit card issuer that offers virtual credit card numbers is City Bank. But this service is limited to a specific customer's category who use "Only Select Citi cards." Still, the process is straightforward for those cards that do qualify. The user needs to register his/her Citi credit card in the program, and then they can generate a virtual credit card number through the online interface. Any generated virtual card number remains valid for up to 12 months. This service is limited to a specific website, and customers can use the virtual number as much as they want until they go to their accounts and request another number. Simply, the fraudster can use the virtual number until the customer notices any abnormal activity on his/her account. Then the customer can log in to his/her account and change the credit card number.

## **2.2 Related Work of Credit Card Fraud Detection Techniques**

In this section, the six main techniques that are used for credit card fraud detection are identified.

### 2.2.1 Support Vector Machine (SVM)

SVM is a good example of supervised learning that can be implemented for classification and regression issues. Support Vector Machine decides the best fitting technique for classifying our dataset's information [43].

A separate hyper-plane formally defines a Support Vector Machine as a discriminative classifier. In other words, given the specified training, the model finds an optimal hyper-plane produced by the algorithm that classifies new cases. This hyper-plane is a line dividing a plane into two segments (two-dimensional spaces) where it is specified on either side in each class.

Data points on the right side of the hyper-planes are classified as legitimate transactions, while the other points are classified as fraudulent transactions.

The optimal hyperplane correctly classifies the data points by fraud. Still, the most effective hyper-plane is the one that achieves a similar level of accuracy when unknown data points need to be classified. SVM selects the optimal hyper-plane based on the line distance. The SVM separates the class to the nearest point. This range is called the margin, and the margin point is known as support vectors.

Sahin and Duman [38] investigated credit card fraud detection using Decision Trees and Support Vector Machines. The authors show that the proposed classifiers of the Decision Tree approach better than SVM does in credit card fraud detection. As the training data scales, SVM model detection accuracy equals the Decision Tree models but falls short in the number of frauds detected.

Bhattacharyya et al. [39] evaluate Logistic Regression performance with two advanced data mining approaches, Support Vector Machine, and Random Forest, for

credit card fraud detection. This study shows that Logistic Regression maintains comparable performance with different under-sampling levels. In contrast, SVM performance points to an increase with a lower proportion of fraud in the training data. On the other hand, Logistic Regression shows appreciable performance, usually exceeding that of the SVM models with different parts.

### 2.2.2 Naïve Bayes

John and Langley first introduced the Naïve Bayes algorithm in 1995 [45]. This model is a probabilistic classifier model. This model indicates that it can obtain predictions for multiple classes at once.

This model is based on the Bayes Theorem. Naïve Bayes has probabilistic classifiers that make this model able to predict multiple classes. The decision is made based on conditional probability. This model utilizes a set of algorithms rather than a single algorithm, but all of these have a common principle. This model implies that each variable makes an equal and unique contribution to the result. Furthermore, this model has a particular advantage over other models as it requires only a small amount of training data [46].

Naïve Bayes classifier is based on the Bayes theorem [47][63] that picks the highest probability-based decision. Bayesian probability is estimated from known values and known probabilities.

Naïve Bayes is a supervised machine learning algorithm that is represented by the following formula.

$$p(A|B) = \frac{p(A|B).p(A)}{p(B)} \quad \text{Eq. 2-1}$$



Bayes theorem gives a method of determining the posterior likelihood  $P(A|B)$ , the likelihood of outcome (A) provided special conditions (B). The Bayes Theorem calculates the later probability by utilizing a probability ratio  $P(B|A) = P(B)$  to relate it to the result's previous probability without any knowledge of clear conditions. The theorem of the Naïve Bayes has based on the theory that each factor influences the outcome independently and is therefore naïve.

Phua et al. [34] have applied Back-Propagation (B.P.), together with Naïve Bayesian (N.B.) and C4.5 algorithms, to skewed data partitions derived from minority oversampling with replacement. The paper shows that the innovative use of Naïve Bayesian, C4.5 and Back-Propagation classifiers to process the same partitioned numerical data has the potential of cutting costs (stacking-bagging of data cost).

Sherly [41] presented a comparative assessment of supervised data mining techniques for fraud prevention. The author evaluated several methods, Decision Tree, Neural Networks, and Naïve Bayes classifiers. The study reported that neural network classifiers are suitable for more extensive databases and take a long time to train the model. Bayesian classifiers are much more accurate and faster to train and ideal for different data sizes but are slower when applied to new instances.

Pun and Lawryshyn [42] applied a meta-classification strategy to improve credit card fraud detection. The approach comprises three base classifiers constructed using the Decision Tree, Naïve Bayesian, and K-Nearest Neighbor algorithms. The result shows a 28% improvement in performance using the naïve Bayesian algorithm as the meta-level algorithm to combine the base classifier predictions.

### 2.2.3 Decision Tree

The Decision Tree (D.T.) approach has been developed by Quinlan [48], which can deal with consecutive data. The Decision Tree is a table of tree appearances made of leave nodes, root nodes, and internal nodes.

As in Figure 2-1, the decision tree makes a decision based on the trained system that comes up with a set of conditions at each level. The decision tree is based on data mining techniques that recursively partition a dataset of records utilizing the depth-first greedy or the breadth-first approach [49] [50]. All nodes and leaves are connected with lines. Each node might be a branch node followed by more nodes or only one leaf node allocated by the Decision Tree classification method.

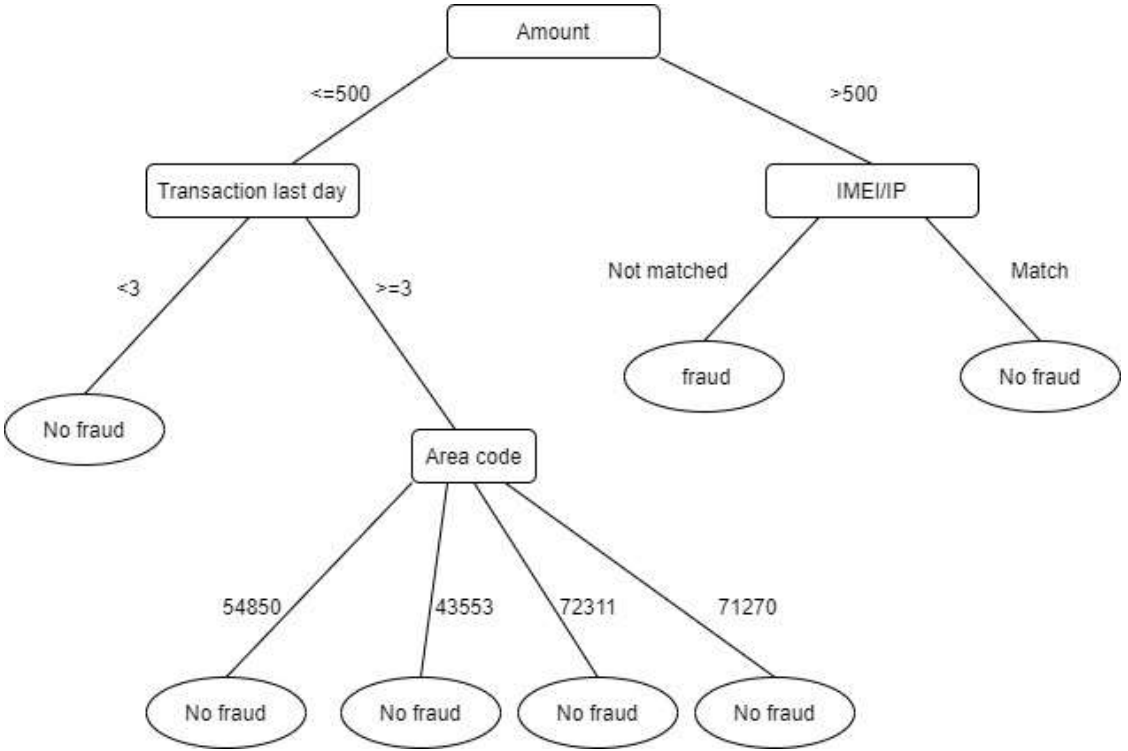


Figure 2-1: Decision tree

Figure 2-1 shows an example of the Decision Tree construction and how this model makes a decision based on the used variables. The Decision Tree solves complex problems by separating them into simple ones and resolves them by constructing a Decision Tree based on the earned knowledge through the data mining technique. The Decision Tree model's basis is building a tree with high precision and a tiny scale.

A research paper by Save et al. [22] used the Decision Tree with a combination of Luhn's algorithm and Hunt's algorithm to detect fraudulent transactions. The paper matched the billing address with the shipping address of the genuine user. It is assumed that these addresses should be matched in order to proceed as a legitimate transaction. Otherwise, the transaction is classified as a suspicious one since a fraudulent one is more likely to differ from the genuine user's address. The paper called this process "Outlier detection" and concluded that validation of the card is genuine and has low false alarms.

#### **2.2.4 Logistic Regression**

Logistic Regression (L.R.) [51] [52] uses a functional strategy that predicts a binary response probability based on one or more variables. The Logistic Regression model includes data mining tasks with more statistical models involving discriminant analysis, regression analysis, multiple-logistic regression, and some other analysis.

The Logistic Regression model has many benefits in credit card fraud situations; it can predict some results of the presence or absence of characteristic values based on a set of variables (predictor variables). Logistic Regression coefficients can be used to evaluate odds ratios for each of the model's independent variables. It applies to a broader range of research situations than characteristic analysis.

A comparison of Logistic Regression and Naïve Bayes is presented by Jordan et al. [32]. The authors provided the mathematical analysis of each algorithm, and they found that the discriminative Logistic Regression algorithm has a lower asymptotic error. Hence, the generative Naïve Bayes classifier may also converge more promptly to its asymptotic error. Some cases have been reported in which Logistic Regression's performance underperformed Naïve Bayes, but this is recognized primarily in small datasets.

Shen et al. [36] have tested three different classification methods (Decision Tree, Neural Networks, and Logistic Regression) for their applicability in fraud detection. The authors show that the proposed classifier of Neural Networks and Logistic Regression procedures work better than the Decision Tree to solve the problem under investigation.

In another study by Sahin and Duman [40], classification models based on Artificial Neural Networks (ANN) and Logistic Regression (L.R.) are applied to credit card fraud detection problems using highly skewed data. This study shows that the suggested ANN classifiers outperform L.R. classifiers in solving the problem under investigation. The L.R. classifiers manage to fit the training data as they increase due to a lack of adequate work sampling.

### **2.2.5 Random Forest**

The Random Forest model is an aggregate classifier. It uses multiple trees by combining many decision tree classifiers. The main idea behind utilizing numerous trees is to train the trees enough, such that participation from each of them comes in the structure of a model. After constructing the tree, the result would be combined through the majority. This model uses multiple decision trees to depend on a particular dataset

possessing similar distribution throughout the tree [53]. This model can be used to solve both classifications and regression problems.

Lakshmi et al. [67] investigate different ML algorithms, Logistic Regression, Decision Tree, and Random Forest performance for credit card fraud detection. They used a popular credit card transaction dataset from Kaggle that comprises 284,808 credit card transactions of a European bank data set. The three techniques are applied for the dataset using the R programming language. The performance of the methods is evaluated for different variables based on sensitivity, specificity, accuracy, and error rate. They have investigated a different number of variables separately, 5, 10, 21 variables. The average result shows that the accuracy for logistic regression, Decision tree, and random forest classifier are 90.0, 94.3, 95.5, respectively.

### **2.3 Other Related Work of ML Algorithms**

Bentley et al. [20] suggested an algorithm called “Fuzzy Darwinian Detection of Credit Card Fraud.” They proposed a system that detects credit card fraud through Fuzzy Clustering, Neural Networks, and Genetic programming by classifying financial transactions into two categories: suspicious and non-suspicious transactions. The findings illustrated that Fuzzy Logic could be an accurate and intelligible classification of complex data. The Fuzzy Logic approach used in this paper is considered one of the oldest approaches to the current ML algorithms, such as Logistic Regression and Decision Forest.

Behera and Panigrahi [21] suggested a system that detects credit card fraud using Fuzzy Clustering & Neural Network in three phases: initial user authentication and verification of the card details; Fuzzy c-means clustering algorithm; and Neural Network

algorithm. The credit card transaction should pass these three phases to determine whether the transaction is fraudulent, suspicious, or legitimate. Once the transaction is found as a suspicious one, the neural network learning mechanism is applied. This system can minimize the generation of false alarms and leads to a more accurate credit card fraud detection system, but the computation time also increases.

Maes et al. [33] studied another comparative study on credit card fraud detection using Bayesian and Neural Networks. The result shows that the Bayesian Networks yield better results concerning fraud detection with shorter training periods, but the fraud detection process is faster with Artificial Neural Networks.

Paniarahi et al. [37] proposed a fusion approach using Dempster-Shafer theory and Bayesian Learning for detecting credit card fraud. They concluded that Bayesian Learning brings down the false-positive rates to values close to 5%. Based on the stochastic synthetic transactions used in this study, the analysis of the system's performance shows that it yielded up to 98% true positive ratio and less than 10% false positive ratio.

## **2.4 Assessment of the Related Work**

In this section, assessment among credit card fraud detection techniques will be applied in terms of cost, time, and performance, as discussed earlier (see Table 2-1).

**Table 2-1: Advantages and Disadvantages of Fraud Detection Techniques**

<b>Fraud Detection Technique</b>	<b>Advantages</b>	<b>Disadvantages</b>
<b>1. Support Vector Machine</b>	<ul style="list-style-type: none"> <li>A. Effective with high dimensional data</li> <li>B. Works for both classification and regression problems</li> <li>C. Works with image data</li> </ul>	<ul style="list-style-type: none"> <li>A. Poor performance with overlapping classes</li> <li>B. Time-consuming with large datasets</li> <li>C. Difficult to understand the produced vectors compared with D.T.</li> </ul>
<b>2. Naïve Bayes</b>	<ul style="list-style-type: none"> <li>A. Requires small training</li> <li>B. Easy to implement</li> <li>C. Fast</li> <li>D. Highly scalable</li> </ul>	<ul style="list-style-type: none"> <li>A. All the attributes need to be mutually independent</li> <li>B. Data scarcity</li> <li>C. Zero frequency</li> </ul>
<b>3. Decision Tree</b>	<ul style="list-style-type: none"> <li>A. Very flexible</li> <li>B. Easy to implement</li> <li>C. Easy to understand</li> <li>D. High accuracy</li> </ul>	<ul style="list-style-type: none"> <li>A. Checks transaction conditions in sequence one by one</li> <li>B. Slow</li> <li>C. Expensive</li> </ul>
<b>4. Logistic Regression</b>	<ul style="list-style-type: none"> <li>A. More accurate with a larger sample size and predictor variables</li> <li>B. Low cost</li> </ul>	<ul style="list-style-type: none"> <li>A. Dependency on one multi predictor variables for more accuracy</li> </ul>
<b>5. Random Forest</b>	<ul style="list-style-type: none"> <li>A. Reduces overfitting and improve accuracy</li> <li>B. Works for both classification and regression problems</li> <li>C. No data normalization required</li> <li>D. Fast and high performance</li> </ul>	<ul style="list-style-type: none"> <li>A. High computational capability is required</li> <li>B. It takes a long time for training</li> </ul>

We can observe the advantages and disadvantages of each credit card fraud technique explained in Table 2-1. A variety of characteristics are found for these techniques in terms of cost, time, efficiency, and the sample size that fits some of these techniques to achieve more accurate and reliable results. Some of these techniques require high computing capability. For example, the Random Forest algorithm is considered very powerful and precise with many transactions and very compatible with vital database infrastructure, but it needs high computing capability. Thus, this model is considered a slow and time-consuming model because it constructs several decision trees based on the dataset size.

On the other hand, we discovered some other inexpensive techniques to implement to obtain good results, such as the Naïve Bayes fraud detection algorithm. However, this model suffers from a few problems, such as data scarcity and zero frequency. Zero frequency happens when a category of any categorical variable is not seen in the training dataset. In that case, the model assigns a zero probability to the category with no fraudulent transactions, and then a prediction cannot be made. In contrast, data scarcity is when a category of any categorical variable has few observations that do not represent the category properly.

We found that the Support Vector Machine is considered one of the slowest fraud detection techniques, although it is an effective model. It can obtain a good accuracy compared with other approaches; besides, it is more appropriate with small datasets. The decision tree is also considered very expensive and relatively slow because it checks transactions one by one. Still, on the other hand, it has higher accuracy and flexibility than other fraud detection systems.

One of the low-cost techniques adopted for fraud detection is the Logistic Regression fraud detection system, but it is more accurate with more predictive variables. Moreover, L.R. is considered one of the fastest models and is easy to implement.

This dissertation addresses these weaknesses. In particular, this study combines more than one factor to detect and prevent any potential fraud to ensure the highest possible security for both customers and card issuers.

In the previous section, we have discussed most of the literature in the credit card security field. According to the literature, we can observe some weaknesses and strengths of different kinds of fraud detection. Research has been conducted to improve



the financial payment systems, including credit cards. This research focuses on online credit card payment systems and tries to enhance the payment system security through some fraud detection algorithms and other methods to prevent any potential fraud or any database breaches in the future.

Based on the literature review, many proposed systems focused on one factor to enhance the credit card security system and ignored other important factors that might be a very powerful way to detect credit card fraud.

Previous research by Rajasekaran and Varadaragan [18] and Yingjiu and Zhang [29] used the one-time credit card number similar to the temporary credit card number for online purchases. The authors proposed good systems to prevent credit card fraud through the one-time credit card system. Still, they ignored many important factors such as the user's geographical location, the device that issues the transaction from, the delivery address, and many more attributes that might ensure higher security for cardholders and card issuers. They could improve their systems by adding one or more predictable factors such as average consumption for each cardholder to ensure more security and reduce error percentage [18, 29].

Many one-time credit card approaches have been adopted for a long time. One of the systems developed by Saxena and Ponnappalli uses a one-time credit card number that generates the credit card number at the user's side offline without contacting the server or being online. Hence, this system will not be able to look up each user's transaction history to observe the user's spending behavior before providing the user with the one-time credit card number.

Rajasekaran's model uses a one-time credit card number generator. This kind of model generates a one-time credit card number at the user's device and sends the generated number to the card issuer and the merchant. Herein, we can conclude that this model does not use any databases to store users' activities to build a solid background of each user's spending behavior or the user's location that is used to make a payment. These factors might be a good indicator to use for future transactions, so the card issuer cannot determine whether it's a fraudulent transaction or not based on the user's transaction history.

These systems might be useful and powerful in detecting credit card fraud, but many companies are not flexible enough to update their conventional systems to fit these new systems. Many customers are also comfortable with the currently used system and are unwilling to make any changes to accommodate these systems. However, these systems may not be appropriate with some other companies due to compatibility issues or costs. Also, several techniques use the one-time credit card payment system in an offline mode. These systems are robust and have less overhead for the customer. Still, they don't consider users' spending behavior in fraud detection. In some cases, these systems can't confirm if the cardholder has exceeded his/her credit limit or not.

## **2.5 Conclusion**

Previously, many credit card fraud detection techniques have been discussed. These fraud detection techniques aim to detect and prevent credit card fraudulent transactions, leading to high security to the cardholders and protecting their personal information. Credit card issuer companies must use more than one method simultaneously to ensure higher protection. Applying these fraud detection techniques to

any credit card company helps the company to minimize the annual losses due to credit card fraud that happen every day.

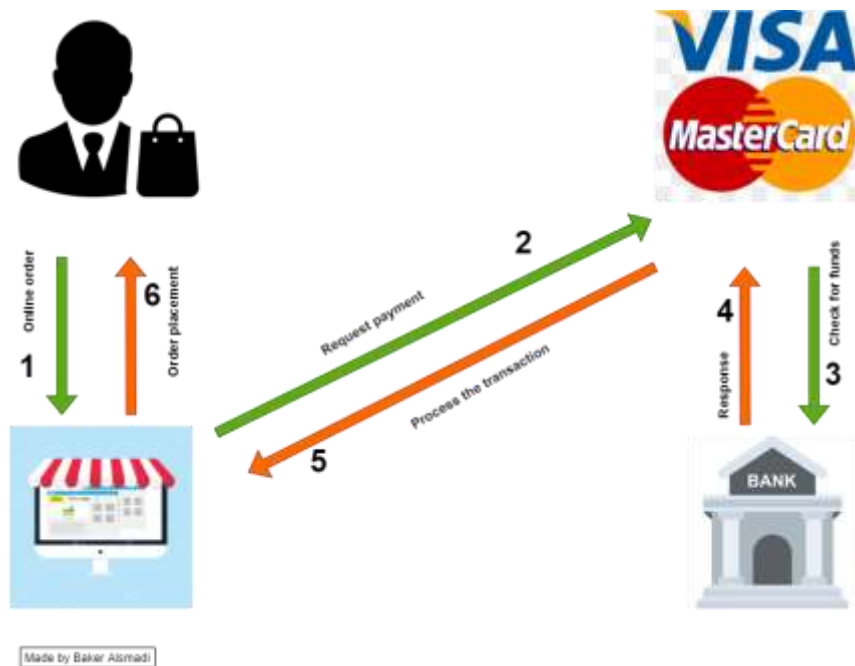
## CHAPTER 3

### THE PROPOSED SYSTEM

This chapter will discuss the current credit card payment system and the proposed system in detail.

#### 3.1 Traditional Online Credit Card Payment System

Security is the main issue in the credit card system for individual and corporate finance transactions. To observe the difference between the proposed system and the conventional system, we should understand how the current credit card system works for online transactions.



**Figure 3-1:** The stages of the regular (current) online transaction.

Figure 3-1 shows that the customers' cycle starts when they enter their credit card number and the 3-digit CVV (Card Verification Value) directly to the merchant's website. All connections from the cardholder through to the issuer bank are secured by conventional security systems such as Secure Socket Layer (SSL). The customer uses the merchant's website to send his/her credit card information to pay for the purchased goods or services. In turn, the merchant passes on the received information to Visa/Mastercard company or another card issuer company to verify the cardholder's identity using the provided credentials. Then the card issuer checks if the cardholder has sufficient funds/credit line to proceed with the transaction. The whole procedure throughout the transaction cycle is processed in a matter of seconds for an online transaction. As for a point-of-sale transaction, the cardholder swipes the credit card at the point-of-sale instead of entering the 16-digit credit card number and the CVV code to the merchant's website. The rest of the procedure is similar to the online transaction, which similarly takes a few seconds. This is how the current online payment system works using a credit card.

### **3.2 The Proposed System**

In this section, we are going to describe the proposed system in detail. The proposed approach is constructed to enhance credit card security in online transactions.

The proposed system's basic requirements include an internet connection and a smartphone or computer application to be able to implement the proposed system. Every user should sign up to the system and fill out a registration form with the required information (the user's name, date of birth, mailing address, phone number, e-mail address, and three security questions if the user forgets his/her username or password). Moreover, every user must set up a password which must include an uppercase letter,

special character, and number. Also, our system uses the user's password as a part of the shared secret between the user and the card issuer. The cardholder should control any payment transaction by using his/her credentials to log in to the system to handle the online transactions without the need to have a physical credit card.

### **3.2.1 Server's Side**

The server is the most important part of our system since it integrates an ML algorithm for fraud detection. Also, it is responsible for doing several tasks. The server receives requests from users that include the transaction information such as transaction amount, merchant's name, and hidden information such as the user's longitude and latitude, transaction time, and the IP/IMEI number. The server handles the following tasks:

1. Authenticates users to log in to their accounts:

The server will verify the user's username and password to ensure that the one-time credit card request comes from the genuine user. Then the user should be able to obtain the credit card number through his/her account.

2. Stores all online transactions information in a database:

All requests will be stored in the database, making transactions history.

3. Runs the machine learning algorithm for fraud detection:

The server will run the ML algorithm for fraud detection based on the stored transaction history in the database.

4. Verifies high-risk transactions:

The server will send a verification message to the user's phone number if it has been evaluated as a high-risk transaction by the ML algorithm. Once the user

verified the transaction's information, the server will activate the user's one-time credit card number and store the transaction information in the database.

5. Activates the one-time credit card number along with the CVV number to the user:

The server is responsible for activating the one-time credit card number and the CVV number directly if it has been evaluated as a legitimate transaction by the ML fraud detection algorithm.

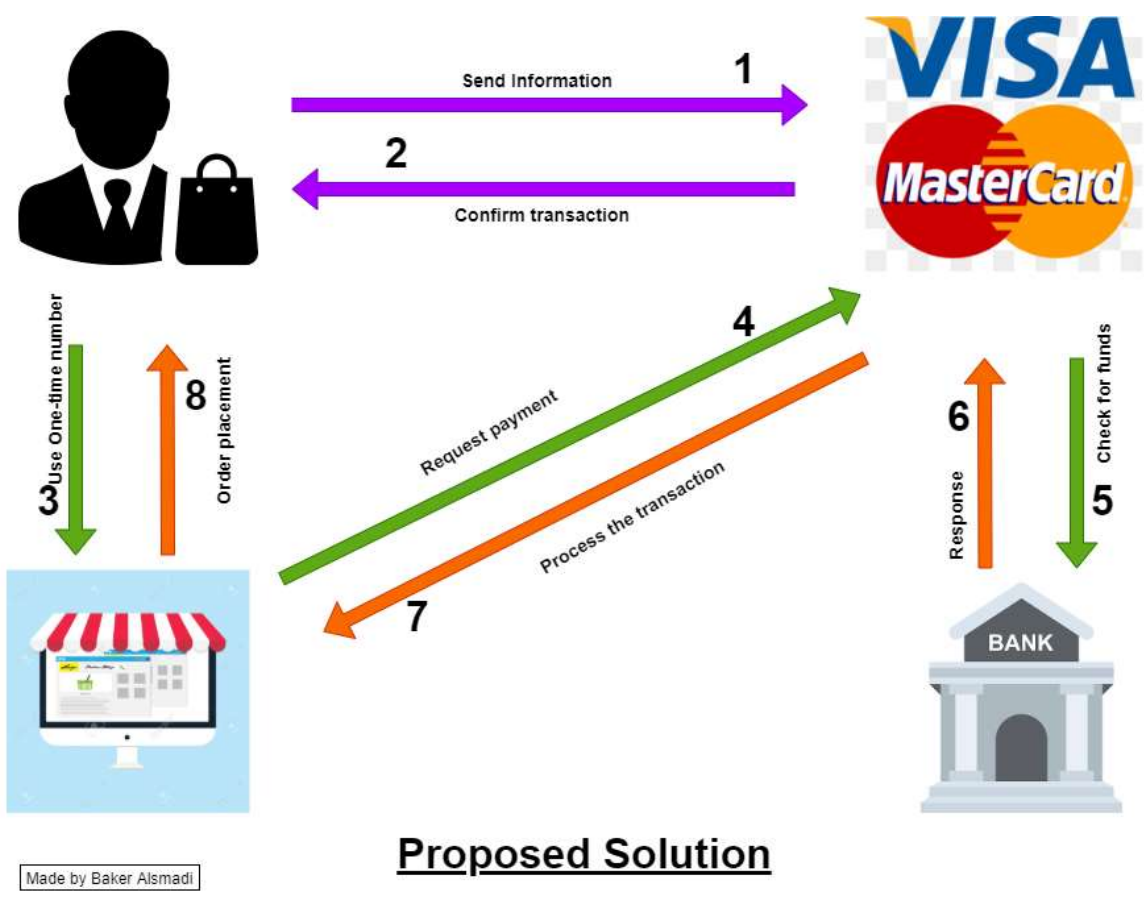
Once the server activated the one-time credit card number and the CVV number, he/she will be able to make an online purchase using the generated one-time number, and each time the user needs to generate a new number for a new online transaction.

### **3.2.2 Description of the Proposed System**

The proposed system comes up with an additional phase (the purple arrows in Figure 3-2) before proceeding to the regular transaction using the same conventional equipment. The new phase is expected to provide higher security to the cardholder as well as the card issuer (see Figure 3-2).

After the user's login to the system through a smartphone or a personal computer using the user's login credentials, the system will move forward to the first step in the new phase. The user generates a one-time credit card number and a CVV code associated with a specific online transaction. In step #1, the cardholder requests the server to confirm the generated one-time credit card number. The user's request comprises the following pieces of payment information: The transaction amount, merchant's website, time of the transaction, and the public key. The user's request also comprises other hidden information without the user's acknowledgment, such as the user's location

(longitude, latitude) and the IMEI (mobile login)/I.P. address (P.C. login). In turn, the server should be able to generate the same one-time credit card number based on the provided information by the user and confirm the transaction. The server will process the given information and store every transaction in a database as a transaction history for machine learning training purposes. The transaction should be done within a limited time session. Otherwise, the server will require the user to generate a new one-time credit card number with new information. The server sets up an amount limit so the merchant cannot charge the cardholder any extra amount of money beyond the requested amount of money.



**Figure 3-2:** The proposed system (online transaction)



The server uses the IMEI/IP address and the other embedded information in the user's request to apply a fraud detection algorithm using machine learning algorithms to prevent any possible fraudulent transactions. The fraud detection algorithms include the user's location, IMEI/IP address, transaction time, and the average consumption of money based on the user's credit card transaction history.

The server, in turn, should be able to verify each transaction using the shared secret (user's password) at the user's machine. The server confirms the user's information and activates the one-time credit card number with a predetermined amount of money if the number is uniquely generated. The ML algorithm considers the repeated information provided by the user as a normal transaction. The normal transaction is the transaction that comes from the same IP/IMEI address, location (longitude, latitude) that the user used to make the online transactions from, and within the average user's consumption. The moderate transaction is the transaction that somehow follows the user's spending behavior. The risky transaction is the one that doesn't follow the genuine user's spending behavior. The prior user-server communication comes to avoid any potential fraudulent transaction.

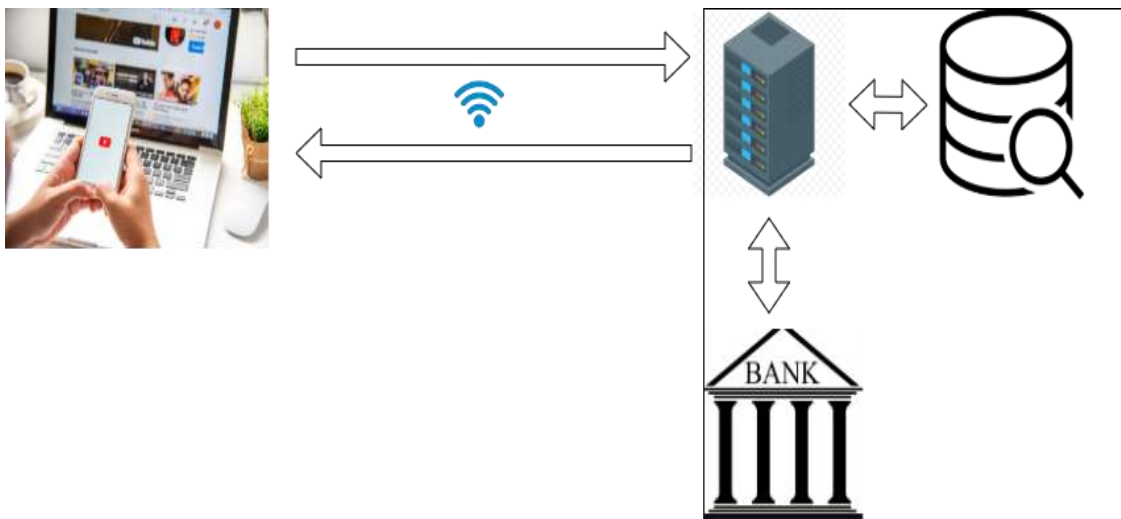
### **3.2.3 Deployment Overview**

Due to the added phase, as shown in Figure 3-3, before continuing to the regular credit card transaction, the user needs to have customized client software installed or log in to his/her account through our website as a user. In order to apply our approach, the following elements are needed:

1. Smartphone/personal computer. The user needs to either use a smartphone or a personal computer to get the one-time credit card number and provide the server

with the transaction's details, such as the transaction amount, the merchant's website. In addition, the user needs to verify his/her identity by entering the username and password to perform any transaction.

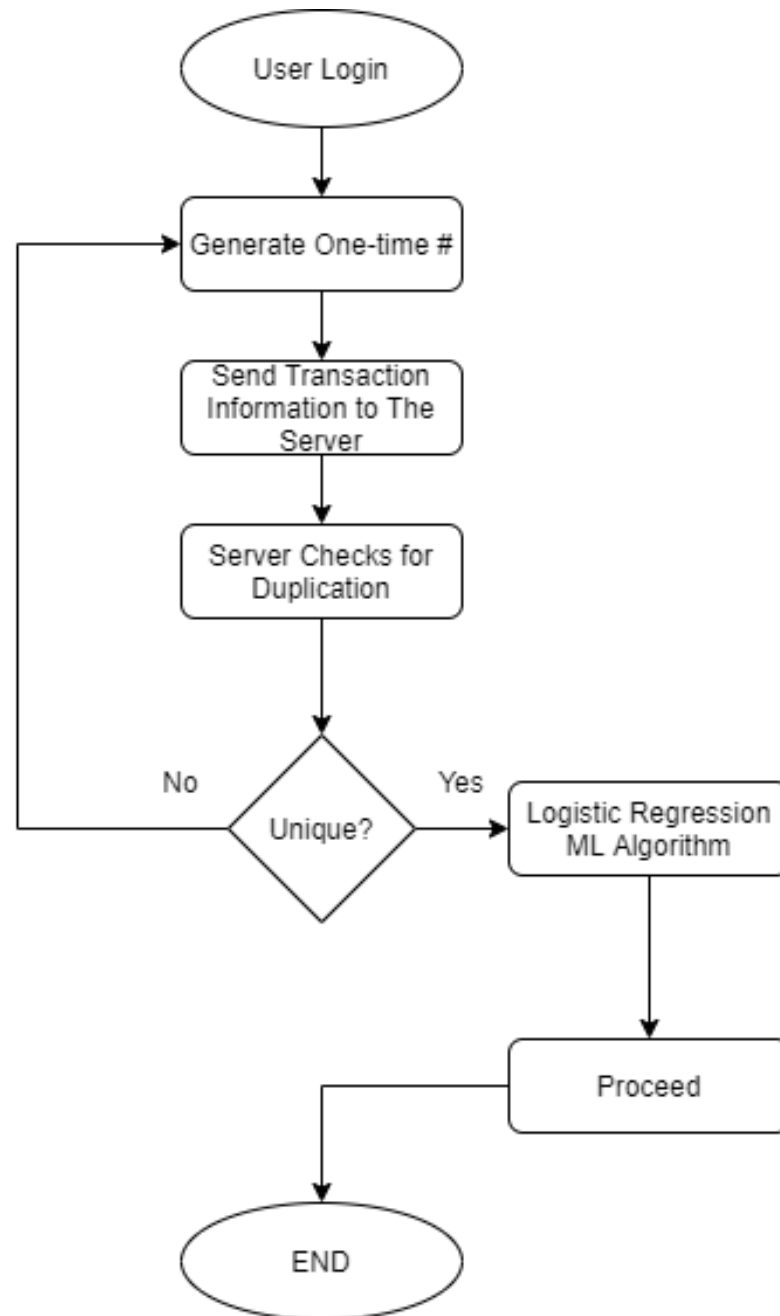
2. Webserver. The system performs most operations at the server-side, including verifying the user's identity, storing the transaction details in the database, applying fraud detection algorithms to determine the transaction's risk, and responding accordingly.
3. Database. The system should have a database to store all user's information and all credit card transactions.
4. Internet connection. Both client and the server should be connected to the internet to perform the desired transaction.



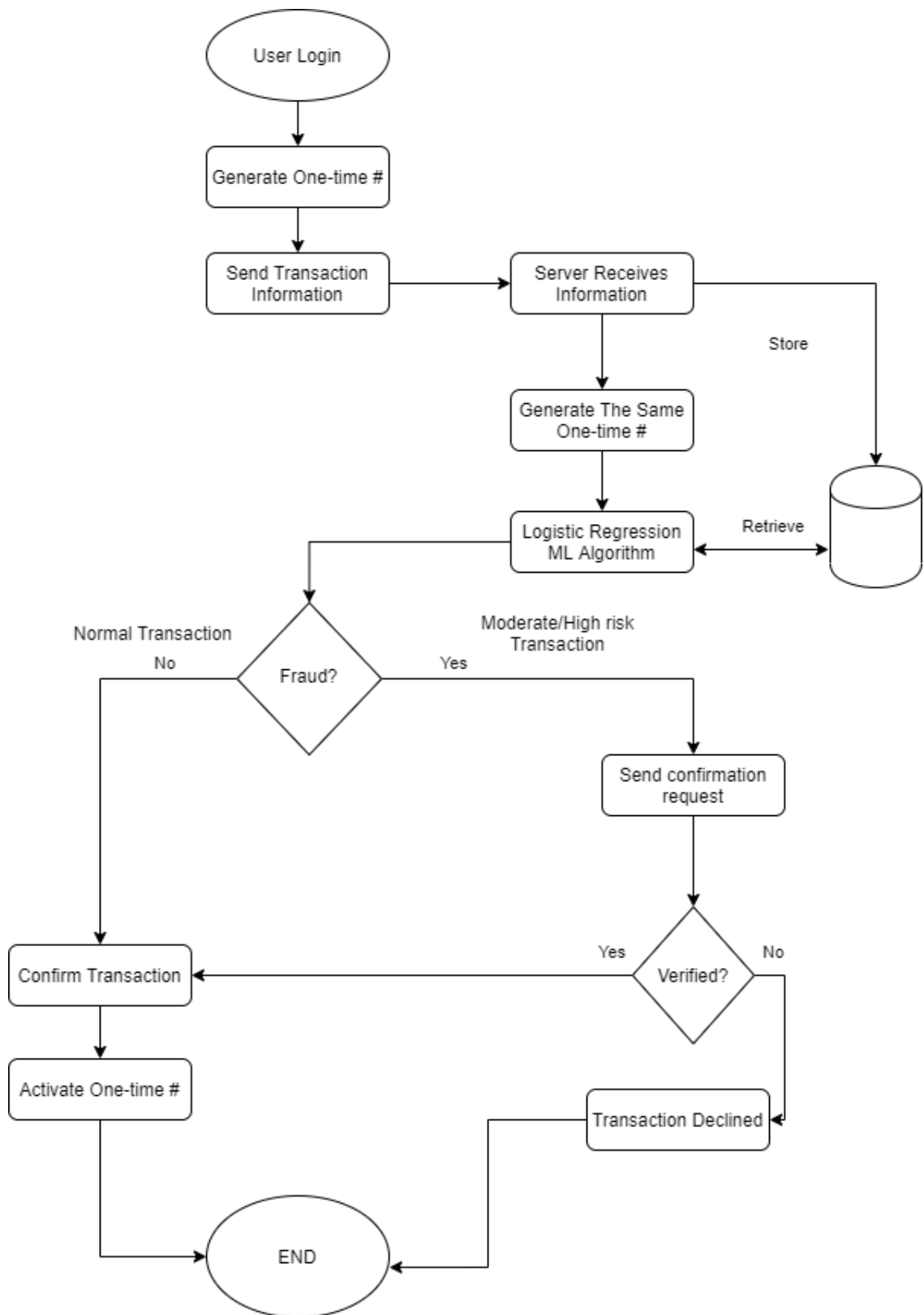
**Figure 3-3:** System overview

As previously discussed, the proposed system uses the current infrastructure and the existing network security systems such as SSL (secure socket layer). The user must create an account, and every created account information should be stored in the system's

database for future login activities and verification purposes. The procedure is depicted as flow charts in Figures 3-4, 3-5.



**Figure 3-4:** One-time number generation



**Figure 3-5: ML integration**

Figure 3-3 shows the flow chart of the one-time credit card number uniqueness verification. The user generates a one-time credit card number and a CVV code to the corresponding transaction at the user's machine. The user sends a public key with the transaction information for every online transaction for verification. The server should be able to obtain the same credit card number using the shared public key, shared secret, and verify that the generated number is unique by checking the server's database for duplication. If the one-time number is unique, the server passes the user's request to the next stage (LR ML algorithm) for fraud detection based on the user's transaction history. It sends a confirmation message to the user informing the user that the generated credit card number is activated and ready to use if the transaction is considered legitimate. Otherwise, the server asks the user to regenerate a new one-time credit card number. If the one-time number is not existing in the server's database, the server passes the transaction to the ML algorithm.

According to Figure 3-4, the ML integration flow chart, the server confirms the transaction if it is considered legitimate based on the ML decision. Otherwise, the server takes further action; it sends a transaction confirmation message that includes the requested transaction's full information to the user's phone number. The user must be able to confirm the transaction's information by clicking "Yes" or decline the transaction by clicking "No." This procedure helps the server verify the cardholder's identity, update the database with the new verified information, or stop the payment transaction considering it as a high-risk transaction if the confirmation message is not confirmed.

According to the payment card industry data security standard (PCI DSS) [54], the 16-digit credit card number is made of three groups, and each group represents a

different code. The first six digits were used to identify the card issuer institution, such as Mastercard, Visa, or American Express, etc. The card issuer must assign a unique identifier to the proposed system to avoid any possible collisions with the real credit card numbers. The following 7-15-digits are used as a cardholder identifier. Hence, the system will generate a unique 9-digit number as a cardholder identifier.

Furthermore, the system will not reissue the same number for any online transaction until the number is used and discarded to avoid credit card number collision. The last digit is a check number generated using Luhn [55] algorithm to ensure that the credit card number has been entered correctly.

Once the user completes the transaction, the generated credit card number will be stored in the database for a month if any product returns to the same credit card number. In this period of time, the stored credit card number will no longer be valid for future transactions. The proposed system uses three security dimensions (one-time credit card number, secure communication medium, integrated with ML fraud detection algorithm) to ensure high security for every online credit card transaction.

### **3.3 Generation of the One-Time Credit Card Number**

The unique one-time credit card number will be generated based on three variables: transaction time, transaction amount, and a random number. This combination is used to generate a unique one-time credit card number and to avoid a possible collision. Mainly, the transaction time itself is a unique variable, especially when we use the transaction time in microseconds. However, the server checks the one-time number for duplication in the database before activating the one-time number to ensure that the generated number is not stored in the server's database.

The one-time credit card number will be generated at both user and server's sides. Both numbers should be matched. Hence, there is no need to share the one-time number between the user and the server. As discussed earlier, the system generates only the middle 9-digit of the credit card number, which is the cardholder identifier number, to maintain the usability of this number at any merchant's online store.

To ensure high security of the credit card number, the server and user use a secret key. The secret key is made of the user's login password that is stored on both sides concatenated with the result of the time (T) multiplied by the transaction amount (A) to generate the same one-time number. Herein, we will have different "Sha256" hash chains for every transaction. The user generates a random number (N) 1-99,999 range to use as a hash exponential value of the hashed secret key. The user applies N hash operations to the secret key to get a public key (verification token) to be sent over to the server. The following equation shows the hash operation implemented at the user's side:

$$public\ key = SHA256^{(N)}(secret\ key || (T * A)) \quad \text{Eq. 3-1}$$

The first 5-digits of the 9-digits will be produced from another hash function's result. The system extracts the first five integer numbers of the following formula's result:

$$digits = sha256(N * A + T) \quad \text{Eq. 3-2}$$

The following 4-digits are generated based on the transaction's time (T) in microseconds and the transaction's amount (A) according to the following formula:

$$X = flooring\left(\frac{T * A}{N}\right) \% 10000 \quad \text{Eq. 3-3}$$

The last digit will be computed by the Luhn algorithm based on the previous 15-digits to meet the current credit card infrastructure. The public key will be sent to the

server with the transaction's amount and time to generate the same one-time number. The server uses the public key and the stored secret key to obtain the same generated one-time number (the code is included in Appendix A.6).

The server also uses the transaction's amount and time to find the following 4-digits using the same formula used at the user's side. By applying the Luhn algorithm to obtain the last digit, the server will have the same one-time credit card number. The server checks the credit card number for duplication and activates the number if the generated number does not exist in the server's database.

The CVV (card verification value) will be generated at both user and server's side based on the shared public key with the transaction time according to the following formula:

$$CVV = (N * T) \% 1000 \quad \text{Eq. 3-4}$$

The CVV number will be associated with the one-time credit card number for each online transaction.

### **3.4 Machine Learning Fraud Detection Component**

The proposed system integrates the machine learning fraud detection algorithm to enhance the security of our system with several features, such as:

1. Location (longitude, latitude)
2. I.P. address (If the transaction was made through a P.C.)
3. IMEI number (If the transaction was made through a smartphone)
4. Time (The time in hours of the day)
5. Time difference between every two transactions (The difference in days of every consecutive transaction)



## 6. Transaction amount

The proposed system will apply the L.R. algorithm based on the features as a second security layer. The server stores every transaction in a database. Hence, the L.R. algorithm can build a consumption pattern based on the transaction's history for every cardholder. Our system is designed to decline suspicious transactions based on the L.R. algorithm decision.

Use case scenario: suppose that a user wants to use his/her credit card to make an online transaction from Amazon. The desired product is priced at \$100. The user must log in to his/her account using his/her username and password. Then, the user should be able to generate a one-time credit card number to use for a specific transaction. Also, the user should provide the server with the transaction's information, such as the merchant's website "Amazon," the amount of the transaction "\$100," and a public key that is associated with this certain transaction. The information will be sent to the server with other hidden information through the client-server communication, such as the user's location (longitude, latitude), IP/IMEI number, and transaction time. The server will generate the same one-time number at its end and store it in a database. Furthermore, the server checks if the generated one-time number exists in the database, so the user needs to regenerate another one-time number. Once no duplication is found, the server will pass the transaction to the ML algorithm for fraud detection.

The server will use the provided information to evaluate the transaction by integrating the ML fraud detection algorithm based on the user's transaction history. If the transaction is evaluated as a legitimate transaction, the server will activate the one-time number. The generated number has a limited amount of \$100, so the user cannot

spend more than \$100 using the same one-time credit card number. Suppose the transaction has been evaluated as a fraudulent transaction. In that case, the server will send a verification code to the user's phone number in order to verify that the request comes from the genuine user. If verified, the server would activate the one-time credit card number. Otherwise, the transaction would be declined.

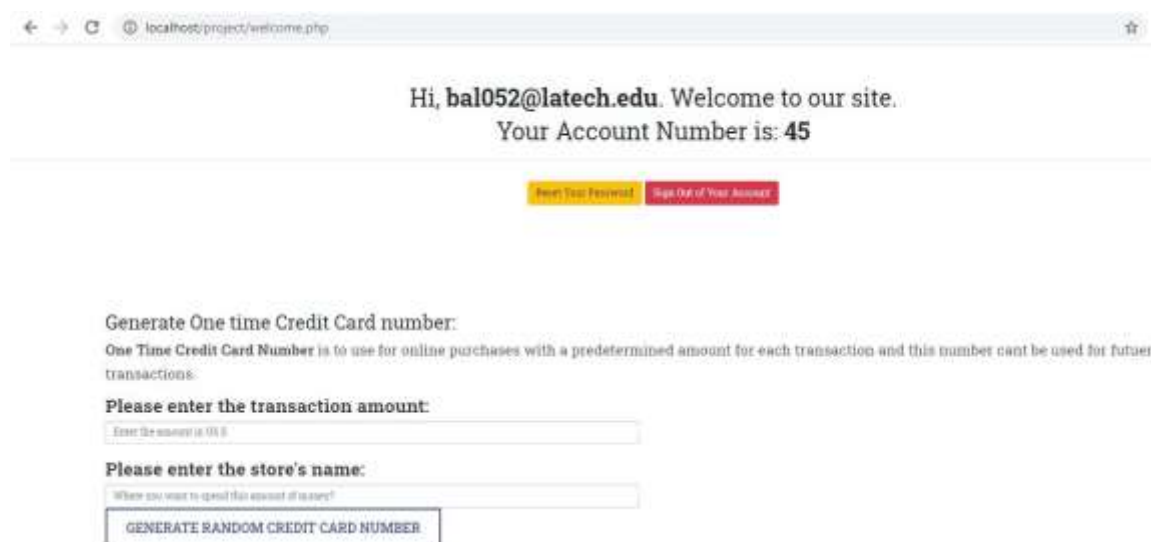
We will use a credit card transactions dataset that is used widely in many research papers. This dataset is made by European cardholders in September 2013 [56]. The provided data is a PCA (Principal Component Analysis) transformed in this dataset due to confidentiality and privacy issues. Also, we will use artificial datasets that incorporate our new features.

We will apply several machine learning algorithms to the dataset, such as the Decision Tree algorithm, Logistic Regression algorithm (L.R.), Random Forest, Support Vector Machine (SVM), and the Naïve Bayes. These machine learning algorithms will help us determine each feature's influence of using more than one attribute to detect potential fraud. Herein, we can observe the difference between using all variables rather than using some of them. Our system is expected to overcome the current systems' weaknesses since all reviewed literature focuses on a few variables and ignores some other variables.

The proposed system will be based on the transaction history to predict future potential fraudulent transactions. Therefore, the more data we have stored in the database, the more accurate the system can perform, whereas a small dataset might result in higher false-positive cases.

### 3.5 User Interface

The user interface allows the user to obtain a one-time credit card number after logging into the user's account using a username and password. The user should determine the amount of money that he/she wants to spend and where to spend it. This provided information will improve the customer's security since the server can constrain the generated one-time credit card number. Figure 3-6 shows how the user's interface looks.



localhost/project/welcome.php

Hi, bal052@latech.edu. Welcome to our site.  
Your Account Number is: 45

Press This Payment! Sign Out of Your Account!

**Generate One time Credit Card number:**  
One Time Credit Card Number is to use for online purchases with a predetermined amount for each transaction and this number cant be used for future transactions.

**Please enter the transaction amount:**  
Enter the amount in USD

**Please enter the store's name:**  
Where you want to spend this amount of money?

GENERATE RANDOM CREDIT CARD NUMBER

**Figure 3-6:** User's interface

After filling out the required fields, the user hits the “GENERATE RANDOM CREDIT CARD NUMBER” button to obtain a one-time credit card number determined for a specific transaction to a particular store.

This information helps the system to set up a limitation to the generated credit card number. Also, the system will match the provided information by the user with the merchant's information during the online transaction.

## **CHAPTER 4**

### **EXPERIMENTS OF ML ALGORITHMS**

This chapter discusses experiments with multiple ML algorithms to determine the best performing algorithm in terms of accuracy and time efficiency that will be used in our system.

#### **4.1 Datasets**

This section describes the datasets that have been used in this research. We have used multiple datasets in our project:

##### **4.1.1 Real Dataset**

The first dataset is the one that many researchers around the world widely use. This dataset consists of 284,807 online credit card transactions recorded for two days by European cardholders in September 2013 [56]. The provided data is a PCA (principal component analysis) transformed in this dataset due to confidentiality and privacy issues. (See Appendix A.1 for more details).

##### **4.1.2 Artificially Generated Datasets**

We have produced several artificial datasets that meet our new features representing different users' behavior based on the users' spending behavior and how they deal with the online payments with various stores or websites.

These datasets have been generated to imitate different hypothetical situations, so we generated six datasets classified into six categories. The six categories and each category specification are listed as follows:

1. Regular spending behavior, i.e., “regular person” (3 different users with 1000 transactions each). Each user
  - A- Has few IMEI/IP addresses and locations (latitudes, longitudes)
  - B- Shops from few online stores
  - C- Consumes on average \$150
  - D- Transacts purchases between 8 am and 11:59 pm
2. Multiple locations, i.e., “person who makes online transactions on several websites” (three different users with 1000 transactions each). Each user
  - A- Has several IMEI/IP addresses with various locations
  - B- Shops from few online stores
  - C- Consumes on average \$500
  - D- Shops at various times of day with no time pattern
3. High spending behavior, i.e., “person who spends a lot of money on online stores” (three different users with 1000 transactions each). Each user
  - A- Has few numbers of IMEI/IP addresses with few locations
  - B- Uses several online stores
  - C- Consumes on average \$3000
  - D- Shops at various times of day with time pattern

4. Vast locations usage. i.e., “a person who makes online transactions at more than 20 different locations” (three different users with 1000 Transaction each). Each user
  - A- Has many various locations and IMEI/IP addresses
  - B- Has a moderate number of different stores
  - C- Consumes on average \$500
  - D- Transacts purchases between 8 am and 11:59 pm
5. Different stores buyer “a person who buys on many online websites” (3 different users with 1000 Transaction each). Each user
  - A- Has moderate number IMEI/IP addresses and locations
  - B- Shops from many online stores
  - C- Consumes on average \$1000
  - D- Transacts purchases between 8 am and 11:59 pm
6. A mix of all the five categories (three different users with 1000 Transaction each). Each user
  - A- Shops from various locations with many IMEI/IP addresses
  - B- Shops from many online stores
  - C- Consumes on average \$3000
  - D- Transacts purchases at various times of day with no time pattern

The six categories were based on various variables with different online credit card payment behaviors. These datasets have different variables that imitate the real human spending behavior, so we still maintain the most probable situations for each user included in the synthetic datasets.

These six datasets are created to represent various difficulties in guessing the user's pattern, where Cases 1 and 2 represent the simplest user's behavior, while Cases 3 and 4 are set to express more complex situations by extending the variables' data range. These kinds of cases would be more challenging for ML algorithms to find a clear pattern. Hence, we expect a decreasing accuracy and precision in more complex cases. The last two Cases, 5 and 6, represent the most complex situations, so the ML algorithms would face a significant challenge to find an apparent spending behavior for each user in these cases. Therefore, we expect the weakest performance in the last two cases.

Tugba Sabanoglu published a study on online credit card transactions in 2020 [60]. The study shows that the majority of customers make at least one online transaction a month with 31% of the respondents, and 24% of customers make online transactions twice every two weeks, and 20% of people use their credit cards to make online purchases once every week. We observe that the percentages are relatively close to each other. Hence, we considered two transactions per week to represent the real data in the real world.

The initial artificial dataset has nine variables (columns), UserAccountNumber; UserName (email address); IP address; TransactionTime; TransactionAmount; TransactionStore; Latitude; Longitude; and Status (fraudulent determination variable: zero-value indicates a non-fraudulent transaction, and one value indicates a fraudulent transaction). This data is not balanced, but it does not have any inappropriate values or missing values. The generated datasets need to be modified and normalized to get a balanced dataset to fit the ML algorithms. These synthetic datasets went through the following operations before integrating the ML algorithms.

1. Transform the transaction's daytime into a number representing the hour of the day of the transaction
2. Transform the date of the transaction into a day's difference between every consecutive transaction
3. Splitting the Transaction IP address into four groups in order to be numerical and readable by the ML algorithms
4. Deleting the Username column because it is nominal, and the ML cannot deal with nominal variables
5. Normalizing all columns to fit our ML models

After implementing the steps, the synthetic datasets now have 11 columns, and all of them are ready for analysis.

Also, these datasets have no fraudulent transactions with the initialization. Still, we added some fraudulent transactions into these datasets. The fraudulent transactions represent 1% of the total transactions. Furthermore, we look for a spending pattern difference between legitimate and fraudulent transactions. We also consider the online shoppers' behavior in terms of the websites they used to spend their money on. IP addresses and locations might be spoofed. Hence, all fraudulent transactions were made with 10% of a fraudulent transaction with the genuine user's exact location and IP address.

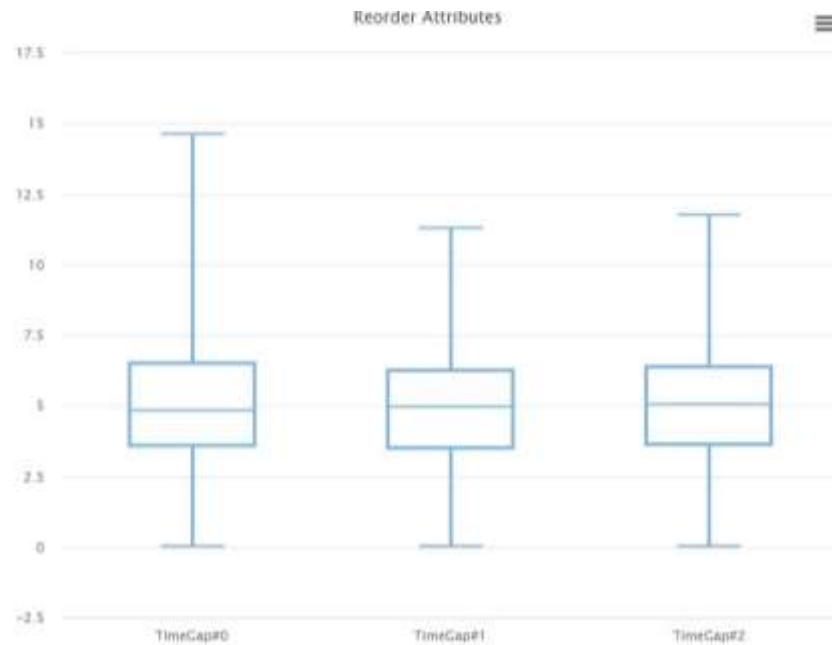
Furthermore, all fraudulent transactions are generated similar to legitimate transactions since fraudsters use genuine accounts to commit fraud but with different consumption patterns considering different situations regarding the amount, time, and



location. We will apply the most effective machine learning algorithms to our datasets to check if they can catch fraudulent transactions.

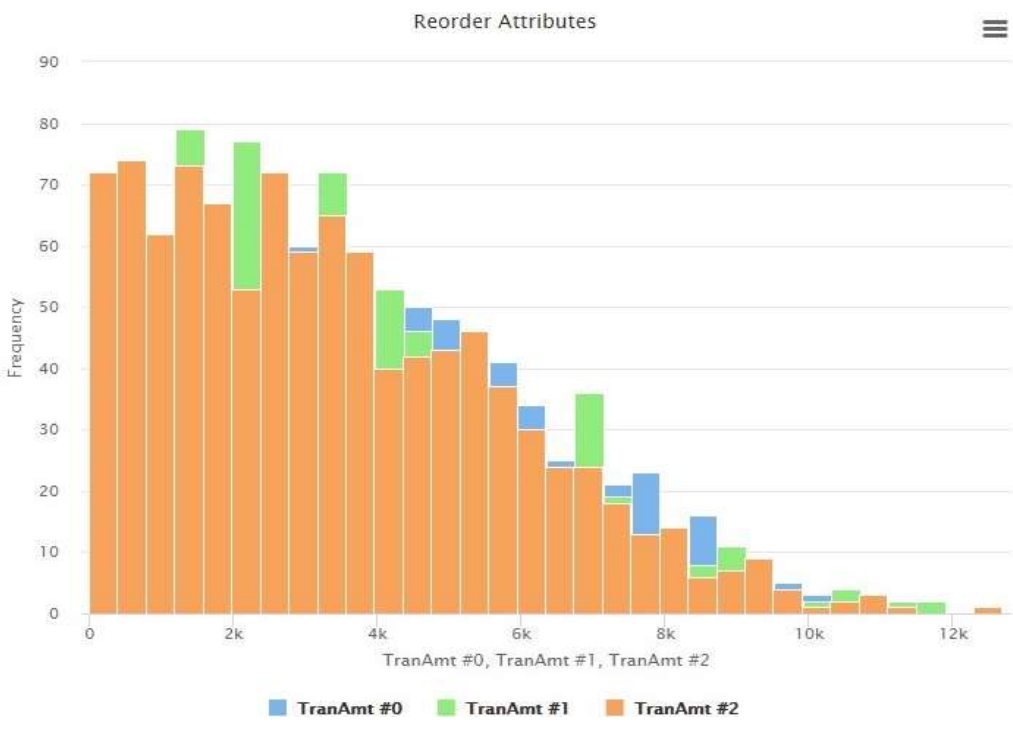
We have generated the artificial data to imitate the real dataset data distribution to avoid any biased datasets. The synthetic datasets represent different user's consumption behaviors; hence these datasets have been generated accordingly. The synthetic datasets comprise nine variables:

1. User Account Number: this number is a unique number assigned to every user in the dataset that identifies each user in number. Thus, the distribution of this variable's values doesn't represent any bias because it's a unique identification number. Furthermore, each dataset has three different users; each user has 1000 transactions in the dataset.
2. Username: we used an email address as a username for each user in the dataset. This variable is not used in the ML algorithm integration because it has a string value.
3. Time gap: The system calculates the difference between every consecutive transaction in days. Hence, we generated the values carefully to be normally distributed to avoid any bias in this feature. Figure 4-1 shows that each user has an average time gap of five days, and all users in the Case 6 dataset (user #0, user #1, user #2) have approximately the same data distribution, where Figure 4-1 shows a normal distribution. The statistical analysis for the rest of the cases can be found in the Appendix (A.2).



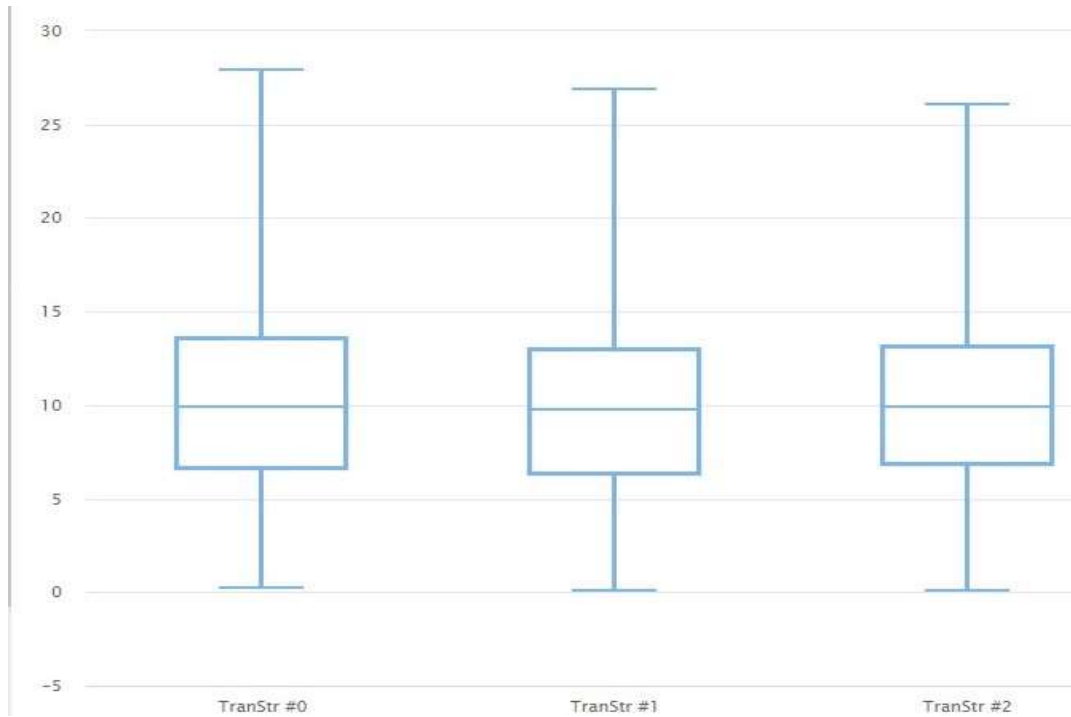
**Figure 4-1:** Time gap

4. Transaction amount: the amount of money of each transaction; this variable's distribution is strongly skewed to the left because most of the transactions have a small amount while a few transactions have larger amounts. According to the case six dataset specifications, as it shows a high consumption behavior, the average transaction amount in Case 6 is \$3000, and the maximum value is \$11,900; the minimum is \$13.5 with a \$2,390 standard deviation. This variable imitates the data distribution of the transaction amount variable in the real dataset. All users (user #0, user #1, user #2) in Case 6 have the same mean and are fairly distributed. Figure 4-2 shows that the transaction amount variable's distribution is skewed to the left since most recorded values were below \$2000. This distribution imitates the skewness of the amount variable in the real dataset. Hence, the data distribution in this dataset shows very similar statistical characteristics to the real dataset.



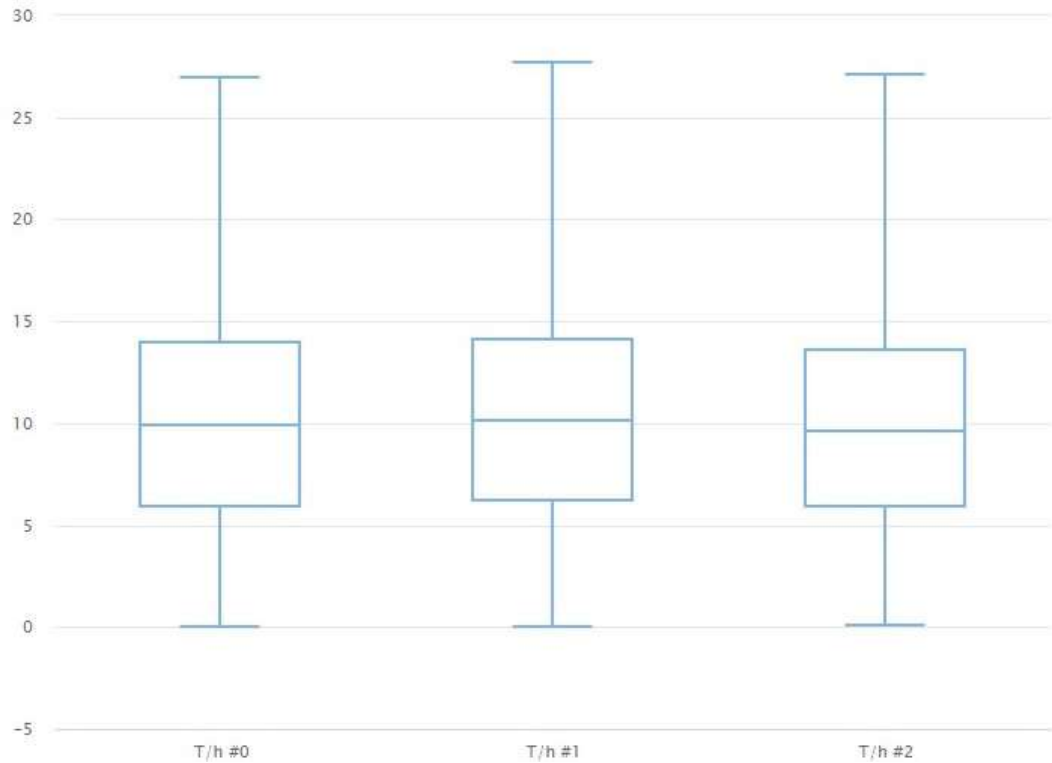
**Figure 4-2:** Transaction amount histogram

- 5. Transaction Store: this variable is a number that represents the merchant’s website of each transaction. This variable has been normalized to avoid any bias in the data generation (see Figure 4-3). These figures show that each user in this dataset has the same data distribution with the same mean =10. The histogram graph shows a normal distribution. Thus, there is no bias in generating this feature.



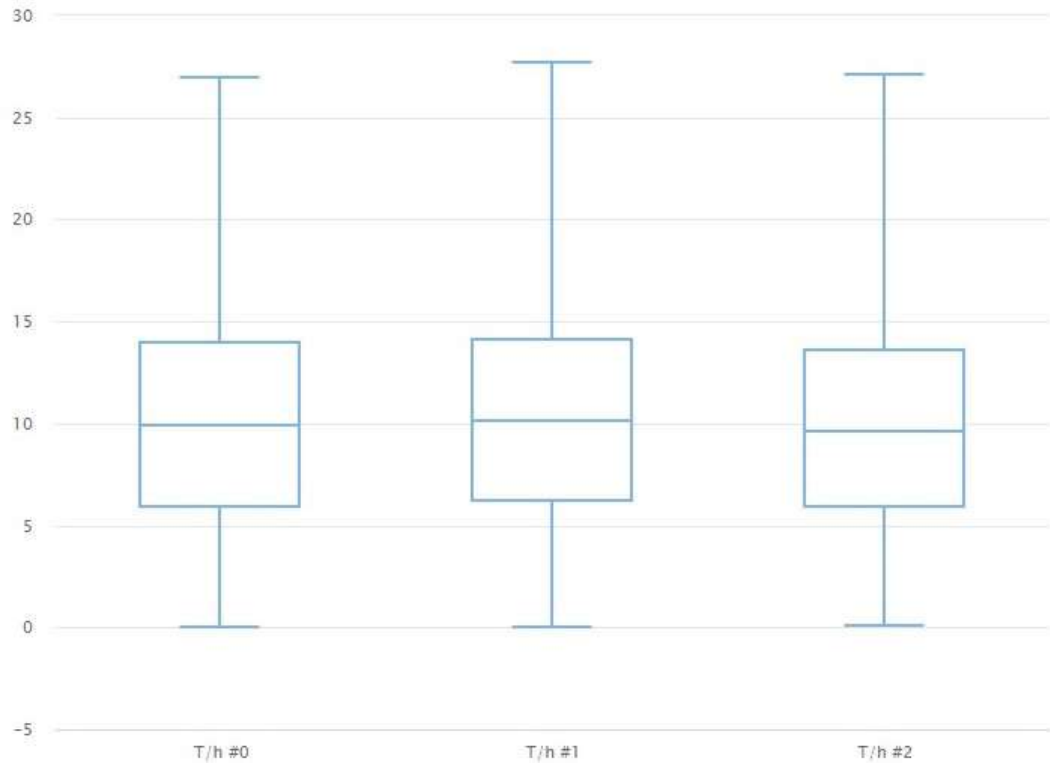
**Figure 4-3:** Transaction store histogram

6. Transaction IP: this variable shows the IP address of each transaction. Since this is just an address, the data distribution is not important because it is considered a string variable.
7. Location (two variables “latitude, longitude”): This variable is to determine the exact location of each transaction. We used real values to fill out these variables. These variables were generated by assigning different locations to different users in the same dataset (see Figure 4-4).



**Figure 4-4:** Transaction store boxplot

8. Time/h: this variable represents the time of the day in hours of each transaction. This variable's value was generated according to each artificial dataset's specifications since we suggested different time ranges in different situations (see Figure 4-5).
9. Status: fraudulent determination variable, "Zero" value indicates non-fraudulent transaction, and "One" value indicates a fraudulent transaction. Note that the statistical analysis includes the fraudulent transactions data as well.



**Figure 4-5: Time/h boxplot**

## 4.2. Research Methodology

This section explains our research methodology and how we will choose the best-fitted ML algorithm for our system based on the proposed features. This research methodology is divided into three steps:

1. Running the chosen machine learning algorithms on the first dataset (the real dataset) and finding the results of each algorithm
2. Running the most efficient machine learning algorithms on the artificially generated datasets and finding the results of each algorithm in each category
3. Finding the results of all experiments and choosing the best-fitted ML algorithm accordingly

### **4.2.1 Machine Learning Algorithms**

This section will discuss the machine learning algorithms used in this research and why we have chosen these algorithms rather than other machine learning algorithms. We use five ML algorithms that help us to detect fraudulent transactions in the real dataset:

1. Decision Tree
2. Support vector machine (SVM)
3. Random Forest
4. Logistic Regression
5. Naïve Bayes

We have chosen the machine learning algorithms because all of them use a binary classifier. Our study looks for two possible cases as an outcome, either a fraudulent transaction or a non-fraudulent transaction. Hence, we have two classes, so we need to distinguish one from another.

### **4.2.2 Machine Learning Criteria**

We have used several evaluation criteria in this study that represent the effectiveness and the efficiency of each ML algorithm. Evaluation criteria help us understand each model's performance and allow us to compare each ML algorithm with others. The most important criteria used in this study are the confusion matrix with many measurements that evaluate the model performance. Confusion Matrix produces a matrix as output and describes the complete performance of the model.

As shown in Figure 4-1, we have four essential measurements used in evaluating each model's analysis in the confusion matrix. These four measurements are True

Positive Ratio (TPR), True Negative Ratio (TNR), False Positive Ratio (FPR), and False Negative Ratio (FNR).

**Table 4-1:** *Confusion Matrix (Sample)*

	<b>Predicted No</b>	<b>Predicted Yes</b>
<b>Actual No</b>	True Negative	False Positive
<b>Actual Yes</b>	False Negative	True Positive

We have also used the performance table with several criteria: accuracy, precision, recall, specificity, sensitivity, classification error, AUC (area under the curve), and others. Accuracy is the ratio of the sum of true positive and true negative to the sum of all the predicted examples as seen in Eq. 4-1.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{Eq. 4-1}$$

Sensitivity, also called recall, is the measure of true positive predictions to the sum of true positive cases and false-negative cases. The recall evaluates the completeness of the program, considering how many true positives were detected as positive. See Eq. 4-2.

$$\text{Sensitivity (recall)} = \frac{TP}{TP+FN} \quad \text{Eq. 4-2}$$

Specificity is the measure of a true negative ratio to the sum of a true negative and false positive. See Eq. 4-3.

$$\text{Specificity} = \frac{TN}{TN+FP} \quad \text{Eq. 4-3}$$



Precision defines the ratio of the number of true positives to the sum of a true positive and false positive, in other words, the measure of the quality of the positive feedback data. The equation of precision is shown below.

$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{Eq. 4-4}$$

AUC (Area Under Curve) represents the probability that a random positive example is positioned to the right of an unexpected negative example.

AUC ranges from 0 to 1. A model whose predictions are 0% incorrect has an AUC of 0, and a model whose predictions are 100% correct has an AUC of 1.0.

AUC is useful for the following two reasons:

- A) AUC is scale-invariant. That means it measures how well predictions are ranked instead of their absolute values.
- B) AUC is classification-threshold-invariant. AUC measures the quality of the model's predictions irrespective of what classification threshold is chosen.

F-Measure provides a combination of both precision and recall as a single measure that captures both properties. Neither precision nor recall tells the whole story. We can have a good precision ratio with a terrible recall ratio or a terrible precision ratio with a good recall ratio. Herein, F-measure provides a way to express both measures with a single score. F-measure is calculated through the following formula.

$$\text{F-Measure} = (2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad \text{Eq. 4-5}$$

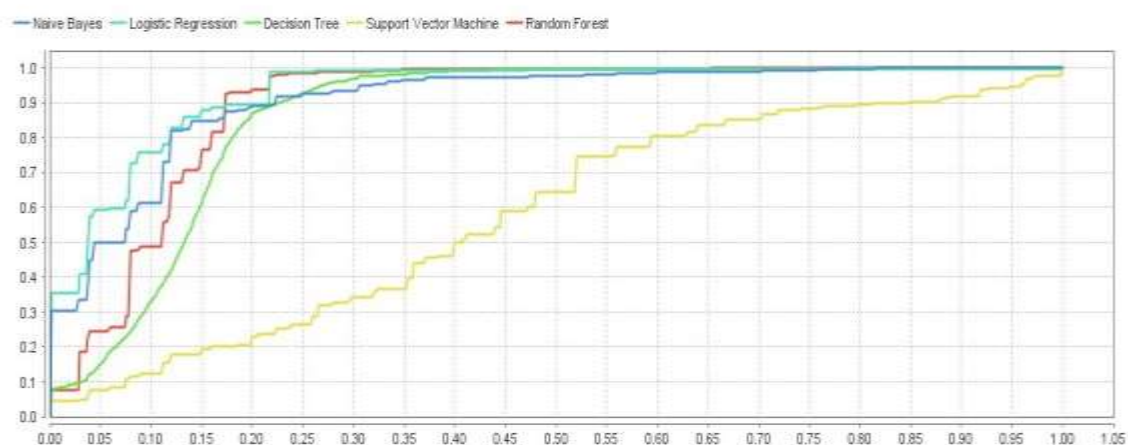
### 4.3 ML Experiment on the Real Dataset (1<sup>st</sup> Experiment)

This section compares each machine learning algorithm's outcomes executed on the real dataset (1<sup>st</sup> experiment). We have integrated five ML algorithms that use a binary

classifier to distinguish fraudulent transactions from non-fraudulent credit card transactions. See Appendix (A.3) for more details.

We show a comparison model that allows us to understand how each machine learning model has interacted with the real dataset. A good visual comparison among used machine learning models is ROC comparison (Receiver Operating Characteristic curve), representing a performance measurement for the classification problems. ROC gives us a better sense of each model's results in predicting true positive cases (TPR) and false-positive cases (FPR). See Figure 4-6.

### ROC Comparison



**Figure 4-6:** ROC comparison

The ROC curve graph summarizes all models' confusion matrices. The Y-axis represents the true positive ratio (TPR). The X-axis represents the false positive ratio (FPR), which means the closer to the graph's top-left side, the better the prediction ratio than others. The ROC graph in Figure 4-6 shows that the Logistic Regression has performed the best among all tested models. The light blue line representing the logistic regression in the chart is the closest to the top-left side, making fewer false-positive cases

more true positives. In other words, we can say that the sensitivity is the Y-axis, and (1-specificity) is the X-axis.

We observed that all models were close to each other in terms of the sensitivity ratio except Naïve Bayes with 91%. Logistic Regression and Naïve Bayes models have shown the best specificity ratio among all other models since they show the highest specificity rate with 79% and 80%. Table 4-2 shows a review of all measures for all machine learning models.

**Table 4-2: Performance Comparison**

<b>Model/Measure</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Naïve Bayes</b>	89.6%	0.908	96.7%	91.9%	93.8%
<b>LR</b>	96.4%	0.935	96.8%	99.1%	97.9%
<b>Decision Tree</b>	93.9%	0.867	93.9%	99.4%	96.5%
<b>Random Forest</b>	94.6%	0.895	99.5%	99.5%	96.9%
<b>SVM</b>	86.2%	0.582	99.9%	100%	92.6%

According to Table 4-2, the Logistic Regression has shown the best performance overall. However, some models have had a comparable performance in some measures, such as the Random Forest and the Decision Tree model. The Logistic Regression has reached the highest accuracy with 96.4%, and F-Measure with 97.9%. Also, the AUC ratio was the best among all ML models with 0.935. In comparison, the Naïve Bayes and the SVM have shown the worst performance among all tested ML models. Therefore, we will not consider those ML models in the next experiments.

Still, we have another essential criterion that hasn't been mentioned in the table, like time efficiency. Table 4-3 compares the time efficiency among all ML models.

**Table 4-3: Time Efficiency**

<b>Model/Time measure</b>	<b>Training Time (1000 Rows)</b>	<b>Scoring Time (1000 Rows)</b>	<b>Total Time</b>
<b>Naïve Bayes</b>	10 ms	198 ms	17 s
<b>Logistic regression</b>	23 ms	219 ms	17 s
<b>Decision Tree</b>	50 ms	172 ms	18 s
<b>Random Forest</b>	282 ms	1 s	2 min 25 s
<b>SVM</b>	4 s	12 s	27 min 1 s

The Logistic Regression and Naïve Bayes models have the best performance in terms of time efficiency. Hence, the Logistic Regression is considered the best model in terms of both performance and time efficiency. Table 4-3 shows that SVM and Random Forest have shown good performance, but it took a long time to run them. Thus, we should balance the time and the performance results to come up with the best-fitted model. We have executed these machine learning algorithms through RapidMiner Studio on a regular machine with standard specifications (Core i5 1.6 GHz processor, 8 GB RAM, Windows 10). Therefore, we might get different time outcomes by using other machines with better specifications.

#### **4.4 Machine Learning Algorithms Experiment on the Artificial Datasets (2<sup>nd</sup> Experiment)**

This section shows the performance of the most effective machine learning algorithms applied to the real dataset in the first experiment: Decision Tree, Logistic Regression, and Random Forest. See Appendix (A.4) for more details. The performance of the chosen ML algorithms in the second experiment is shown in Table 4-4.

**Table 4-4: Average Performance of the 2<sup>nd</sup> Experiment**

<b>The average performance of 2<sup>nd</sup> experiment</b>					
<b>Model/Measure</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Decision Tree</b>	87.2%	0.5605	93.6%	88.70%	87.00%
<b>Random Forest</b>	90.3%	0.82	95.35%	51.80%	85.20%
<b>LR</b>	94.5%	0.875	94.10%	97%	95.50%

Table 4-5 summarizes all ML algorithms' time efficiency on each case of the artificial dataset. In the table, we observe that the machine learning algorithms behave differently with different inputs and variables.

**Table 4-5: Average Time Efficiency of the 2<sup>nd</sup> Experiment**

<b>2<sup>nd</sup> experiment average time efficiency of the six cases</b>			
<b>Model/Time measure</b>	<b>Training Time (1000 Rows)</b>	<b>Scoring Time (1000 Rows)</b>	<b>Total Time</b>
<b>Logistic regression</b>	<b>177.5 ms</b>	<b>112.5 ms</b>	<b>2 s</b>
<b>Decision Tree</b>	<b>170.6 ms</b>	<b>160 ms</b>	<b>2 s</b>
<b>Random Forest</b>	<b>129.5 ms</b>	<b>601 ms</b>	<b>14 s</b>

All three ML algorithms used in the second experiment use a binary classifier to distinguish fraudulent from non-fraudulent credit card transactions. While different performances of each ML algorithm between the first and the second experiment were expected and observed due to the difference of the dataset sizes, Logistic Regression still reached a 94.5% accuracy and 96% precision ratio as an average of all cases compared with 96.4% in the 1<sup>st</sup> experiment.

In conclusion, the Logistic Regression algorithm still shows the best performance among all ML algorithms. It shows the best accuracy, precision, F-measure and achieved the best time efficiency of all models. Hence, we decided to use the LR algorithm in our system.

Due to the variation of inputs in each case in the second experiment, machine learning spends more time processing data in the more complex cases because of the difficulty of guessing each user's pattern in each dataset. Hence, the more complex the case is, the more time-consuming the model becomes. The Random Forest model's complexity justifies a long time in the processing since it needs to construct many decision trees according to the input variables. Therefore, it has the worst time efficiency among other models.

#### **4.5 First Experiment Vs. the Second Experiment**

This section discusses the results of each experiment and compares all results in terms of performance, efficiency.

##### **4.5.1 Performance Comparison (1<sup>st</sup> Experience Vs. 2<sup>nd</sup> Experience)**

This section compares the first and the second experiment's results with each other. This study uses different metrics to evaluate each machine learning algorithm and how they behave with different variables, dataset size. The best performance ML algorithm is the one that achieves the highest accuracy, precision, F-measure, and AUC percentages in the shortest time.

As shown in Table 4-6, we see that the five ML algorithms have scored an average of 89.23% F-measure 94.35% precision in the first experiment. The Naïve Bayes and the SVM have shown the worst performance compared with other models; therefore,

we haven't used these models in the 2<sup>nd</sup> experiment. Also, we excluded the SVM algorithm in the 2<sup>nd</sup> experiment because it consumes a lot of time detecting fraudulent transactions. According to all performance measures, the best performance shown in the 1<sup>st</sup> experiment was the Logistic Regression model by running 96.4% accuracy and 96.8% for precision with 97.9% F-measure.

**Table 4-6: Performance Comparison (1<sup>st</sup> Experience Vs. 2<sup>nd</sup> Experience)**

<b>Model/Measure</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Naïve Bayes</b>	89.6%	0.908	96.7%	91.9%	93.8%
<b>LR</b>	96.4%	0.935	96.8%	99.1%	97.9%
<b>Decision Tree</b>	93.9%	0.867	93.9%	99.4%	96.5%
<b>Random Forest</b>	94.6%	0.895	99.5%	99.5%	96.9%
<b>SVM</b>	86.2%	0.582	99.9%	100%	92.6%
<b>The average performance of 2<sup>nd</sup> experiment</b>					
<b>Model/Measure</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Decision Tree</b>	87.2%	0.5605	93.6%	88.70%	87.00%
<b>Random Forest</b>	90.3%	0.82	95.35%	51.80%	85.20%
<b>LR</b>	94.5%	0.875	94.10%	97%	95.50%

The 2<sup>nd</sup> experiment examines six different synthetic datasets with different variables. Each dataset comprises 3000 rows designated for three users. Our model is based on unsupervised machine learning, so the system learns from the past transactions and trains itself accordingly.

In comparison, the dataset in the 1<sup>st</sup> experiment has 284,804 rows. On the other side, the 2<sup>nd</sup> experiment has six datasets of 3000 rows each. Thus, obtaining 94.5%

accuracy and 96% precision in the second experiment utilizing the Logistic Regression algorithm is a meaningful achievement because it has shallow data that it can rely on.

#### 4.5.2 Time Efficiency Comparison (1<sup>st</sup> Experience Vs. 2<sup>nd</sup> Experience)

In our study, we have a different number of variables in the two experiments. The 1<sup>st</sup> experiment has 21 different variables included after the data cleaning in the experiment, while the 2<sup>nd</sup> experiment has only 11 variables. Hence, in the first experiment, the ML algorithms are expected to detect fraudulent transactions better than the 2<sup>nd</sup> experiment.

Table 4-7 summarizes the ML algorithms' time efficiency in both first and second experiments. As shown in the table, the SVM showed the worst time efficiency in the first experiment. The Logistic Regression model showed good time efficiency in both experiments, and the decision tree model has a comparative time efficiency to the Logistic Regression.

**Table 4-7: Time Efficiency (1<sup>st</sup> Experience Vs. 2<sup>nd</sup> Experience)**

<b>1<sup>st</sup> experiment time efficiency</b>			
<b>Model/Time measure</b>	<b>Training Time (1000 Rows)</b>	<b>Scoring Time (1000 Rows)</b>	<b>Total Time</b>
Naïve Bayes	10 ms	198 ms	17 s
Logistic regression	23 ms	219 ms	17 s
Decision Tree	50 ms	172 ms	18 s
Random Forest	282 ms	1 s	2 min 25 s
SVM	4 s	12 s	27 min 1 s
<b>2<sup>nd</sup> experiment average time efficiency of the six cases</b>			
<b>Model/Time measure</b>	<b>Training Time (1000 Rows)</b>	<b>Scoring Time (1000 Rows)</b>	<b>Total Time</b>
Logistic regression	177.5 ms	112.5 ms	2 s
Decision Tree	170.6 ms	160 ms	2 s
Random Forest	129.5 ms	601 ms	14 s



The Logistic Regression still proves the best performance and time efficiency in both experiments. According to the time efficiency in both experiments, we can see that the scoring time in the second experiment is 50% less compared with the first experiment, which is faster in predicting fraudulent transactions. We conclude that using fewer critical features leads us to better time efficiency with roughly the same performance.

#### **4.6 Excluding Features (3<sup>rd</sup> Experiment)**

In this section, we excluded the location feature (longitude, latitude) from the synthetic datasets to observe the importance of the integration of features together. We have tested all six cases in this experiment. Logistic Regression ML algorithm will be applied since it showed the best performance among all tested ML algorithms in the previous experiments. We have excluded the location feature in this experiment to observe the influence of excluding one feature from the dataset. After excluding the location feature and applying the LR algorithm, predicting fraudulent transactions was dropped down, as Table 4-8 shows. The table shows a tangible impact by excluding the location feature. Thus, we conclude that location is a critical variable for fraud detection. Furthermore, the integration of all features together leads to better performance.

**Table 4-8: Excluding Location (3<sup>rd</sup> Experiment)**

<b>LR performance with excluding the location</b>				
Case/ criteria	Accuracy	Precision	Recall	F-Measure
Case 1	96.8%	97.3%	98.9%	98.9%
Case 2	86.4%	86.4%	100%	92.6%
Case 3	85.5%	85.4%	100%	92.1%
Case 4	86.8 %	86.8%	100%	91.4%
Case 5	84.3%	84.3%	100%	91.4%
Case 6	76.2%	97.2%	95.3%	86.4%
<b>LR Performance without excluding the location</b>				
Case 1	98.9%	98.8%	100%	99.4%
Case 2	97.5%	97.2%	100%	98.6%
Case 3	93.8%	93.4%	100%	96.5%
Case 4	97.2%	98.8%	97.7%	98.2%
Case 5	88.1%	97.1%	88.7%	92.4%
Case 6	85.5%	89.4%	94.4%	91.6%

As a further experiment, we will depend on the location feature as the only feature in the dataset that we can use to detect fraudulent transactions using only one feature rather than using more than one predictor variable. Table 4-9 shows the result of excluding all variables and keeping the location as the only feature.

**Table 4-9: One Feature ML Integration**

<b>Case/ criteria</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Case 1</b>	81.9%	0.454	96.0%	82.9%	88.7%
<b>Case 2</b>	87.5%	0.674	87.2%	100%	93.1%
<b>Case 3</b>	86.2%	0.26	86.2%	100%	92.6
<b>Case 4</b>	82.0%	0.798	87.2%	93.8%	90.0%
<b>Case 5</b>	88.1%	0.277	88.1%	100%	93.5%
<b>Case 6</b>	77.5%	0.842	79.5%	96.9%	87.2%

According to Table 4-9, the performance of the LR model based on the location feature is poor compared to using many features to detect fraudulent transactions. In conclusion, the performance of the LR model with our six critical features has achieved better performance than depending on few features.

## **CHAPTER 5**

### **DISCUSSION**

In the previous chapter, the design details of the proposed system were covered and discussed. In this chapter, the proposed system's different aspects are evaluated based on the two experiments' performance as well as the one-time credit card payment security analysis. Moreover, the various factors and parameters that play a role in both experiments' performance are analyzed.

#### **5.1 Security Analysis**

As discussed in Chapter 3, the proposed technique improves the currently used systems by adding a new phase that includes several steps before authentication to the user's account. These steps help our system enhance online credit card transactions' security since the user needs to obtain a new one-time credit card number for every online transaction with a predetermined amount of money. Our system is designed to defend against several kinds of attacks, such as man-in-the-middle attacks, database breaches, guessing and cyber-attacks, and unauthorized users. Our system offers secure communication between the cardholder and the server since both sides will be able to generate the same one-time credit card number without sharing it during the user's request.

### 5.1.1 Defend Against Potential Breaches

As explained earlier, this system is designed to defend against several types of attacks. We will assume several attack scenarios and how our system will be able to avoid them.

1. **Man-in-the-middle attack:** is a kind of cyberattack where an unauthorized outsider intrudes into an online correspondence between two users, escaping captured by the two parties. Man-in-the-middle-attack can monitor and change individuals' information until the two users realize it. The proposed system will protect the one-time credit card numbers since both sides of communication don't share the one-time credit card number. Instead, both users and the server will generate the same credit card number locally at their sides using the hashed public key. The only information that any Man-in-the-middle can eavesdrop on is the hashed public key, which is nonreadable without having the secret key that only both sides of communication have. Hence, even if a hacker has obtained the public key, he/she cannot get the one-time credit card number. Thus, our system can overcome Man-in-the-middle attacks by maintaining a secure communication channel.
2. **Database breach:** our approach uses a temporary credit card number for every online transaction, and this number is only valid for one transaction with a predetermined amount of money at a specific online store. Since the credit card numbers stored in an online merchant's database can be used only once, so even if the attacker has access to the entire history of transactions of a user, he/she can't

reuse any of them nor fabricate another credit card number that will pass the server's verification unless he/she knows the shared secret.

For example, potential cybercrimes involving stealing credit card numbers and database breaches will be difficult because of the new phase since all users have to obtain a unique credit card number for each transaction. Hence, the credit card numbers stored in the merchant's databases will not be valid for future uses.

Moreover, the system does not let any credit card numbers be shared between the cardholder and server.

3. Guessing of the random number (N) attack: this threat is impossible in our system because the generated one-time credit card number is generated based on several parameters, and mainly on the secret key that no one knows except both communication ends.

The secret key is a series of hashing functions applied to a random base number concatenated with time and amount variables. Hence, it is impossible to guess the generated random number at the user's side since there is no explicit number being transmitted in the user-server communication. Further, the generated random number will not be used explicitly in the one-time credit card number. The number that is used as a part of the one-time credit card number is a result of another hashing function of different parameters. Thus, the hacker will face another obstacle in finding the actual credit card number.

Furthermore, the actual one-time credit card number will be shared only with the merchant to pay for a service or a product. This operation is done in a matter of seconds, and then the one-time credit card number will not be valid for future transactions. Hence,

the hacker cannot obtain the one-time credit card number and use it for illegal purposes within 1-3 seconds.

### **5.1.2 Authentication**

As discussed earlier, every user must have an account that facilitates the online transaction operations by signing into the system and filling out a form that includes all required information.

Our system uses the provided account information by the user to authenticate him/her to log in to his/her account safely. Furthermore, the system uses the phone number to send a verification message that indicates the requested transaction details to verify risky transactions as an additional security layer.

For example, suppose the server receives a fraudulent request to activate a one-time credit card number from an unauthorized user with an unknown location, IP/IMEI address. In that case, the server will not be able to obtain the generated one-time credit card number on the user's device. Hence, the server sends a verification message to the user's device, informing him/her of the suspicious transaction and stopping it. As another use case scenario, if the server received a fraudulent request with the same user's information, the server should still detect an abnormal behavior by applying the ML fraud detection algorithm. The server sends a verification message to the user's phone number to authenticate the transaction or stop it if the user didn't verify the transaction.

## **5.2 Overview on ML Algorithms' Results**

In this section, we discuss what happens behind the scenes. In fact, the primary operations are implemented at the server's side without the user's acknowledgment. The server applies the fraud detection operations through the machine learning algorithm. In

this study, we proposed several added features to develop the current online credit card system. The proposed features are:

1. User's location (longitude, latitude)
2. IP address or the IMEI number
3. Transaction's store
4. Transaction time period (time difference between every two consecutive transactions)
5. Time (the time in hours of the day)
6. Transaction's amount

Our system uses the above features combined to make fraud detection much more efficient. In order to support our assumption, we have chosen the most appropriate machine learning algorithms in our case of detecting fraudulent transactions. We have used several datasets to evaluate our model. The first dataset is a real dataset, and the other datasets are artificially generated datasets that imitate the real dataset. We have tested all datasets by integrating several ML algorithms on all datasets to observe the differences among them and determine how ML algorithms interact with different situations or users' behaviors separately. Therefore, we have chosen the Logistic Regression ML algorithm to be integrated into our approach due to both experiments' great performance.

A comparative study by Trivedi, Naresh Kumar, et al. [30] investigated the same real dataset we used in this research. Their results show that the Random Forest obtained the best performance with 95% precision, but they ignored how time-consuming this



model is. On the other hand, our system shows better performance utilizing the Logistic Regression algorithm by achieving 96% precision with a very time-efficient model.

In another study by Lakshmi et al. [67], they have implemented three ML algorithms, Decision Tree, Logistic Regression, and Random Forest, on different variables. The first dataset has only five variables, and the obtained accuracy of each model was as follows: LR 87.2%, DT 89%, and RF 90.1%. The second dataset has 10 variables, and the result was as follows: LR 88.6%, DT 92.1%, and RF 93.6%. In comparison, our system utilizes only six features, and the achieved accuracy shows better results compared with their approach by obtaining 94.5% accuracy for the Logistic Regression model.

### **5.3 One-Time Credit Card Number Analysis**

One promising future direction of this work is designing a new credit card system that does not use permanent credit card numbers. The proposed system is expected to ensure a secure online payment system with a high capability of catching fraudulent transactions. The one-time credit card approach is used by several companies/banks.

For example, Capital One's bank provides this service for all its customers. In comparison, our system generates a unique one-time credit card number for every transaction, and the generated number will not be usable again at any merchant's websites. The user needs to obtain a new number every time. Hence, our system will provide higher security to the cardholder.

On the other hand, our system takes further steps before generating the one-time credit card number. We can verify the user's location by retrieving the longitude and the latitude and the users' spending behavior by storing every transaction's detail, including

the IP/IMEI number, in a database. Thus, we know all users' activities that help us detect any suspicious activity and stop it. Furthermore, our system generates the one-time credit card number at both sides (user and server's side) to ensure higher security.

Another example of a credit card issuer that offers virtual credit card numbers is City Bank. But this service is limited to a specific customer's category who uses "Only Select Citi cards." In comparison, our system is willing to work at any internet browser and smartphone application since the customer has to log in to his/her account and obtain a one-time credit card number with a predetermined amount at the user's side. The customer needs to specify the merchant's website and the transaction amount to set up a limitation to the credit card number for a specific online store with a certain amount of money. The one-time credit card number will be discarded automatically by the system right after the transaction completion, and this credit card number will not be valid for future uses, but the one-time credit card number will be stored in the server's database for a limited time in case there is a product returned. The generated number will be attached to the genuine user's account.

Many one-time credit card approaches have been adopted for a long time. One of the systems developed by Saxena and Ponnappalli [16] uses a one-time credit card system that generates the credit card number at the user's side offline without contacting the server or being online. On the other hand, our approach also generates the one-time credit card number at the user's side. The user sends the transaction information, including a public key. The generated one-time number is made of multiple variables; it doesn't depend only on the secret key.

Furthermore, the server receives information implicitly to use for the second security layer (ML fraud detection algorithm). The server activates the one-time number if the transaction is considered legitimate. Also, it should be able to generate the same one-time credit card number based on the provided information. First, the user checks for duplication, while Ponapalli's approach doesn't pay attention to this issue because the number will not be stored in a database. This is a weakness in their system because their approach cannot handle product return issues.

Another comparative study by Rajasekaran and Varadarajan [18] developed a new model to reduce the potential credit card fraud by a one-time credit card number generator and single round-trip authentication. In comparison, our system generates the one-time credit card number at the user's side, without the need for further operations as described in this literature. The server should be able to verify the generated one-time number immediately if both numbers generated at both sides are matched and pass the ML fraud detection algorithm. Moreover, the generated number is designed to meet a particular online transaction characteristic in terms of the transaction amount and specific online store.

Our system has extended the current systems by combining several features such as location, IMEI/IP address, the time difference between transactions, and the online store. This study enhances credit card security systems by integrating two levels of protection, one-time credit card number, and integrating machine learning algorithms using the new critical features.

### **5.3.1 One-Time Credit Card Number Generation**

This section analyzes how our system generates a unique one-time credit card number. We designed our system using different programming languages such as HTML, JavaScript, PHP, and CSS. The one-time credit card number has been designed to meet the current credit card numbering structure. The first 6-digits are reserved to identify the card issuer. Our system focuses on generating the following 9-digits using PHP programming language. The last digit is a check number generated using the Luhn algorithm.

The generated number is then stored in the server's database to reference each online transaction and to afford any product return. The server assures non-duplicate credit card numbers by matching every generated one-time credit card number with the stored ones in the database to avoid any transactions collision. This step comes to ensure that the generated number is unique, although this case is almost impossible since the system uses several variables to generate the one-time credit card number, including the transaction time in microseconds. The transaction time is always an increasing variable, which ensures a unique number every time.

The PHP and MySQL code to generate the one-time credit card number and the CVV security code is provided in Appendix A.6.

## **5.4 Conclusions**

This section concludes the work in this dissertation. Our approach combines the machine learning (ML) algorithm with unique temporary credit card numbers in one integrated system, which is the first approach in the online credit card protection system. Our system proposes secure communication between the cardholder and server. Both

sides of the communication collaborate to generate the same one-time credit card number in a secure channel. The one-time credit card number is generated at the user's side and verified by the server using a secret key and a verification token (public key). Our approach integrates the ML fraud detection algorithm as a second security layer.

We have investigated several ML algorithms to find the best-fitted algorithm for our system based on six hypothetical cases. Five ML algorithms were used in the 1<sup>st</sup> experiment, Decision Tree, Logistic Regression, Random Forest, Naïve Bayes, and SVM, which were applied to a real dataset. In contrast, we picked the best three models to be used in the 2<sup>nd</sup> experiment. In conclusion, we found that the Logistic Regression model has the best performance. The Logistic Regression shows 96.4% accuracy with the best time efficiency in the 1<sup>st</sup> experiment. On the other side, it shows 94.5% accuracy in the second experiment with great time efficiency. The second experiment showed an outstanding time efficiency in reducing the scoring time in the first experiment by more than 50% utilizing only six critical features compared with the first experiment. The third experiment shows that the six features combined lead to better performance compared to using some of them.

The one-time credit card system is designed to be compatible with the current online payment infrastructure since it meets the existing credit card numbering structure. The one-time credit card number will be stored in the server's database for a month. Hence, we can deal with any refunds to a particular online transaction. In comparison, other one-time credit card systems cannot deal with refunds to the used one-time credit card number, such as the Capital One virtual credit card system and Citi bank's temporary credit card system.

This dissertation intends to minimize cyber threats such as database breaches and Man-in-the-middle attacks in electronic commerce by combining both the one-time credit card number approach and the ML fraud detection algorithm.

## **CHAPTER 6**

### **CONCLUSIONS AND FUTURE WORK**

Credit card security is essential for both consumers and credit card issuers due to increased financial fraud and database breach issues. Therefore, this study addresses this issue in order to improve the security of current credit card systems.

We have achieved a working strategy to overcome the security issues in permanent credit card numbers by proposing a secure method utilizing one-time credit card numbers. This study combines the one-time credit card approach with machine learning algorithms to ensure a robust online payment system. Although our approach causes a little overhead on the customer by requesting a one-time credit card number every time before proceeding to the regular process of making an online transaction, this approach protects customers' money from being stolen by fraudsters.

Moreover, every generated number is unique and will be used for only one online payment transaction with a predetermined amount of money assigned to every transaction. Several factors have been used to integrate the Logistic Regression ML algorithm, such as IMEI/IP address, location (longitude, latitude), average consumption, transaction store, and time to obtain a consumption behavior for every cardholder. These factors enhance the system's performance to detect fraudulent transactions. Furthermore, our approach uses the current online payment infrastructure; hence, there is no need for extra equipment to implement our approach in any credit card company immediately.

This research proposed a secure online credit card payment system that can be immediately deployed utilizing existing infrastructure. Hence, one future project can be a practical deployment of the proposed system. Below are some potential future projects:

1. Exploring significant factors that might improve the credit card payment system
2. Trying to enhance the in-store payment systems using new approaches
3. Considering improving the online payment systems with different approaches
4. Presenting our proposed system to various credit card companies for adoption
5. Diving deeper into the machine learning algorithms as an effort of improving these models or developing a new approach to detect fraudulent transactions
6. Enhancing the one-time credit card system by obtaining the credit card number without involving the user in this process



**APPENDIX A**  
**REAL DATASET ANALYSIS AND RESULTS**

### A. 1 Real Dataset

The real dataset is highly imbalanced; 0.172% of all the transactions were fraudulent in nature. This dataset has 30 features. In the dataset, we are not provided with original features and background. We are provided with a PCA transformed version of the features due to confidentiality and privacy issues. This dataset has the following features: V1, V2, V3, ..., V28 are PCA transformed for customer's privacy issues. We have some other features that are not subjected to PCA transformation, such as 'time' and 'amount' [56].

Due to this dataset's nature, we have applied several operations on this dataset to be able to use it properly. The following operations have been applied to the dataset:

1. Cleaning the Dataset:

First, we need to clean the dataset since it has many inappropriate and missing values in order to be able to use it to train our model. In general, this step involves deleting the rows that have missing or inappropriate values using RapidMiner studio [64].

2. Balancing the dataset:

Imbalance in a dataset is usually reflected by the asymmetrical distribution of classes within the dataset. For ease, we can call the class that makes up a massive proportion of the dataset as majority classes and the class that makes up a smaller proportion as minority classes [57]. We need to apply a sampling technique before performing a classification task on the imbalanced dataset. It is not easy to train a model with an imbalanced dataset.

We solved the problem of dealing with an imbalanced dataset by oversample the minority class. We have used RapidMiner Studio software [58] to balance our dataset.

In the feature space, RapidMiner studio selects close samples by drawing a boundary between the samples in the feature space and picking a new sample at any point along that line. Also, RapidMiner studio helped us to generate the data visualization [59].

### 3. Data analysis:

Our approach focuses on understanding the credit card transactions dataset and developing an effective model to detect fraudulent transactions. To better understand the dataset, we have performed exploratory data analysis (EDA) using widely used open-source libraries such as NumPy, Pandas, Matplotlib, and Seaborn using PyCharm community edition 2020.2 [65].

Matplotlib and Seaborn are excellent libraries for visualization in Python. We have obtained several visualizations such as histograms, bar graphs, density plots, and box plots using RapidMiner studio to get a better sense of the dataset.

For a better understanding of this dataset, see the following visualizations.



**Figure A- 1:** Transactions amount frequency

From Figure A-1, we can tell that most of all credit card transactions' amounts are below \$2,500 with an average of \$88.35. Hence, we can rely on this feature combined with other features to detect fraud if the upcoming transaction is not in the rage.

4. Features' correlation:

This dataset has 28 PCA transformed variables and two natural variables, time and the Transaction amount. The following figure shows the correlation heatmap of all variables in this dataset:

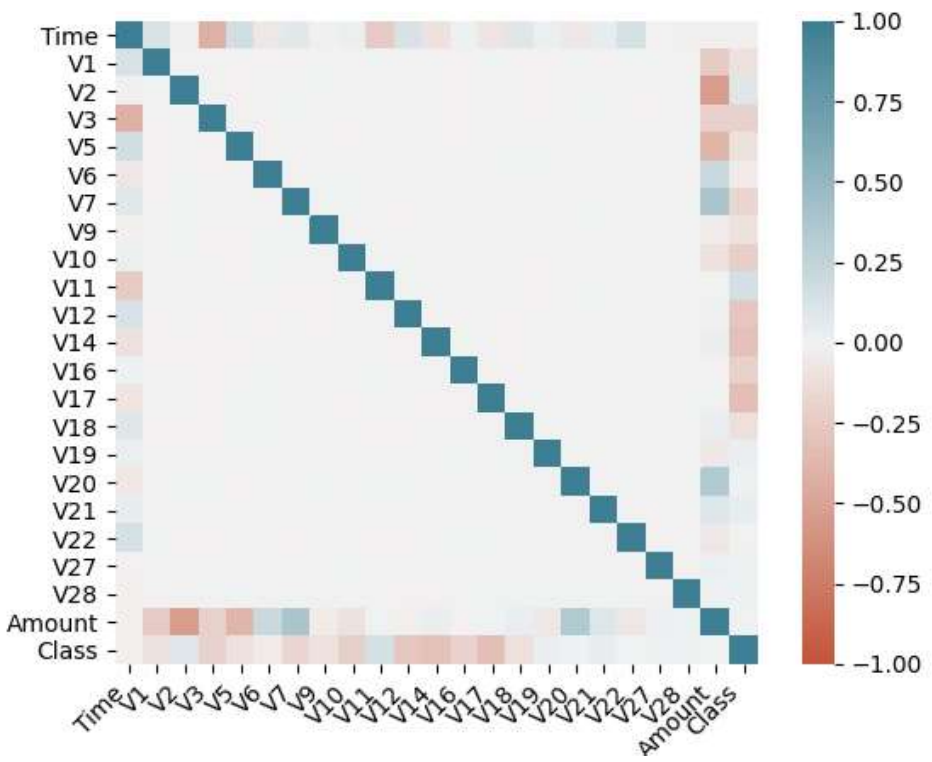


Figure A-2: Correlation heatmap (Exp1)

Based on the correlation result, we conclude that some variables have no influence on the result of detecting fraudulent transactions because they have a low correlation ratio. We found that V4, V8, V13, V15, V22, V23, V24, V25, and V26 have no tangible correlation, and they have no influence on the results. Therefore, we have

excluded these variables in the analysis of detecting fraudulent transactions. The following figure shows the weights by correlation that we need to consider in our study.

### A. 2 Artificial Datasets

#### A.2.1 Statistical Analysis of Datasets

This section shows the data distribution in the first five synthetic datasets and the last dataset (Case 6 is shown in the dissertation).

##### 1. Case 1 statistical analysis

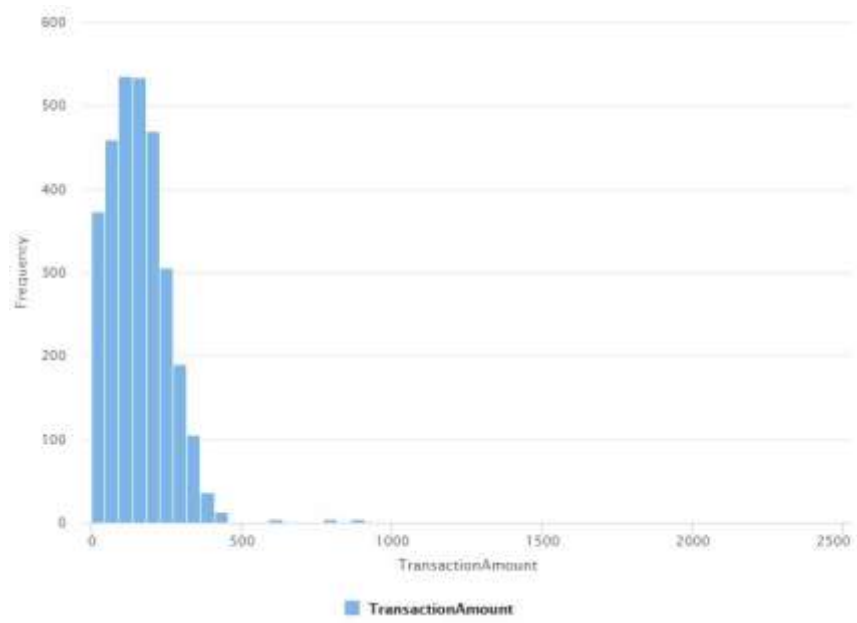
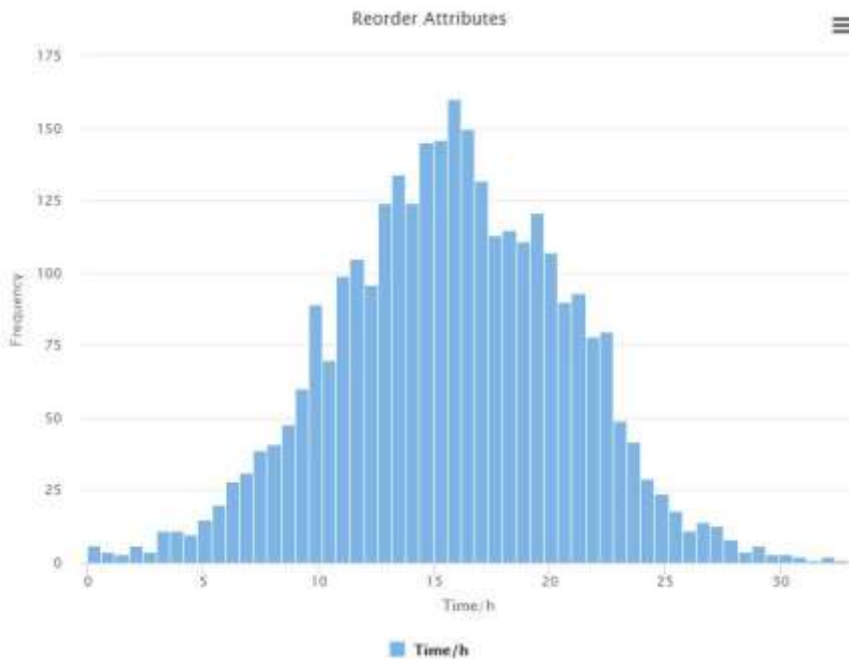
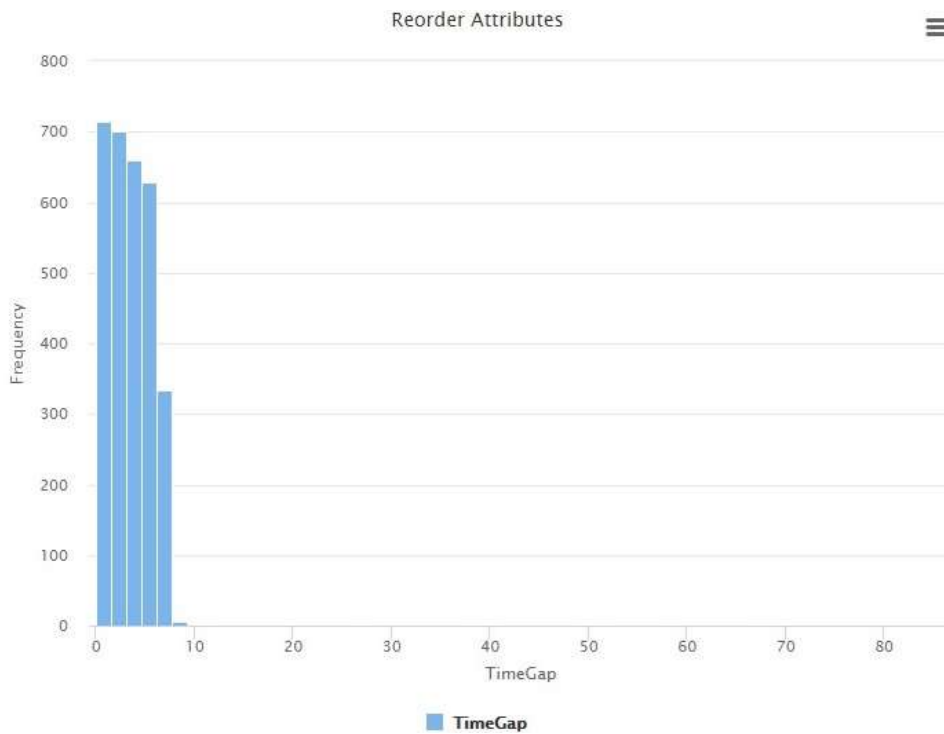


Figure A-3: Transaction amount

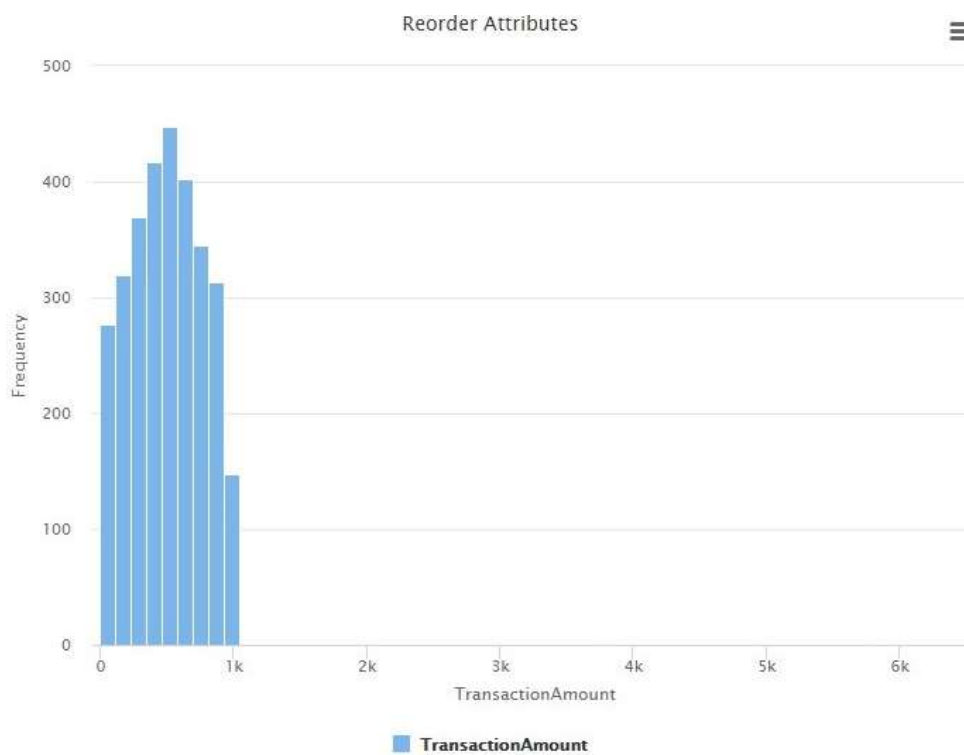
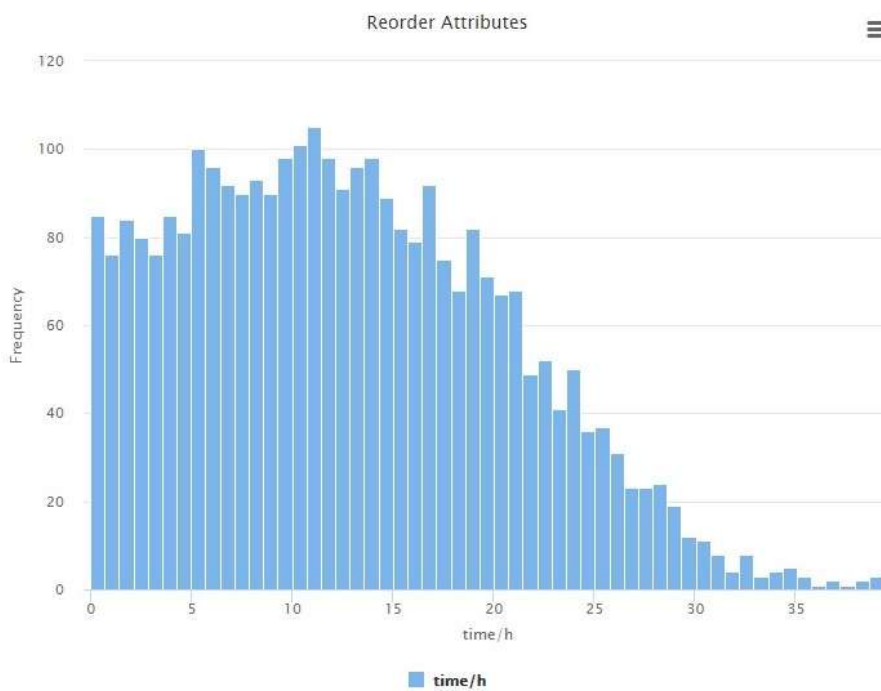


**Figure A-4: Time in hours**



**Figure A-5: Time gap**

## 2. Case 2

**Figure A-6: Transaction amount****Figure A-7: Time in hours**

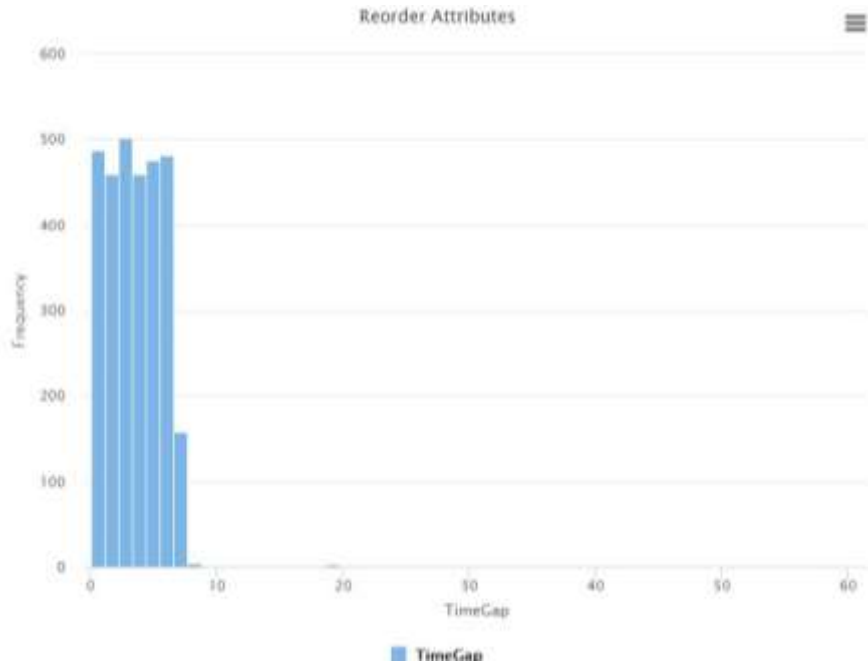


Figure A-8: Time gap

3. Case 3

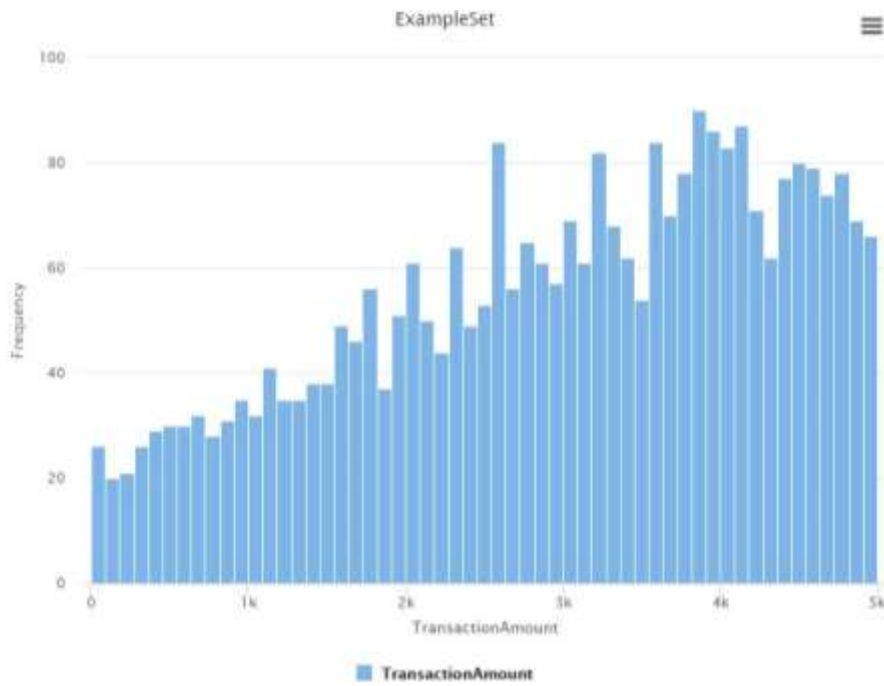
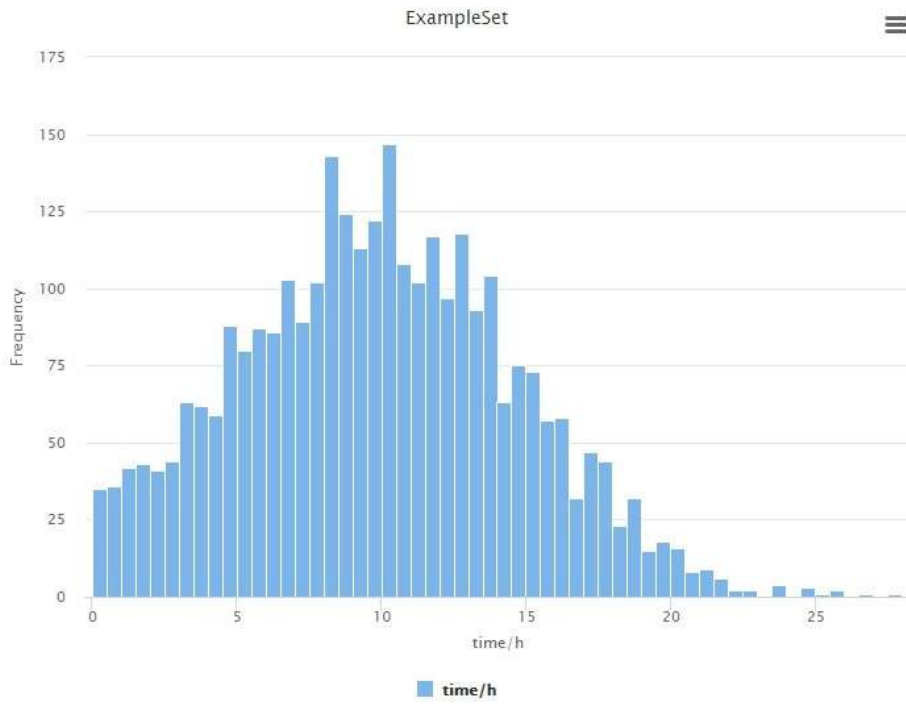
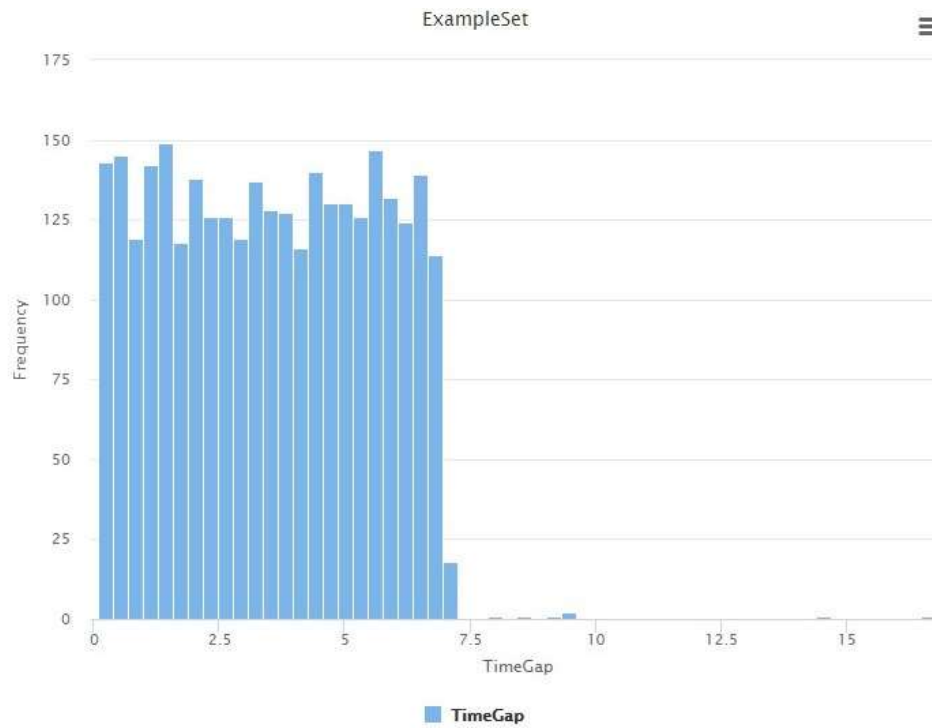


Figure A-9: Transaction amount



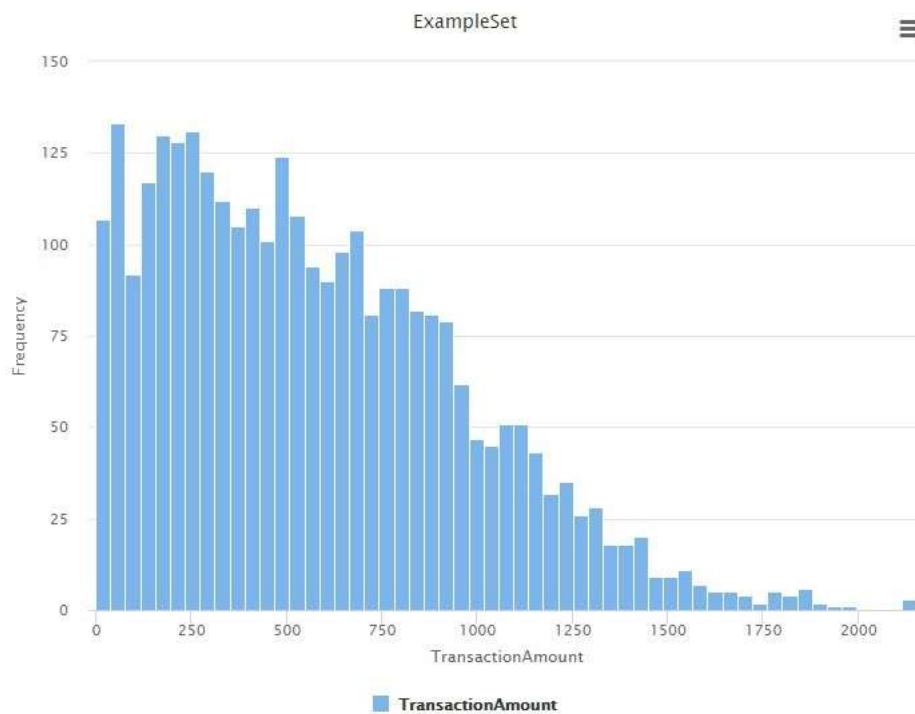
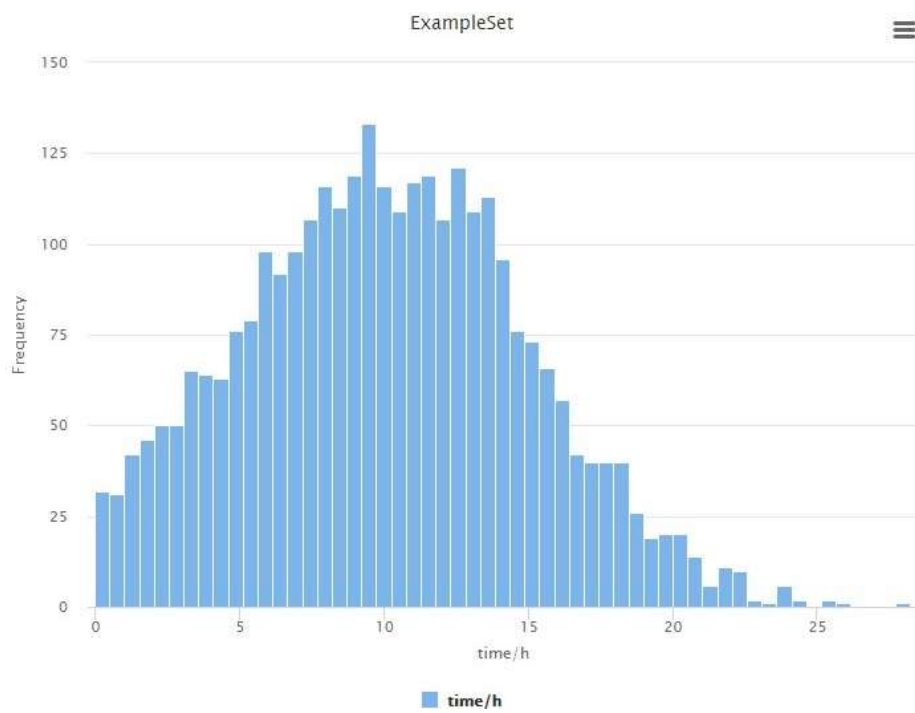


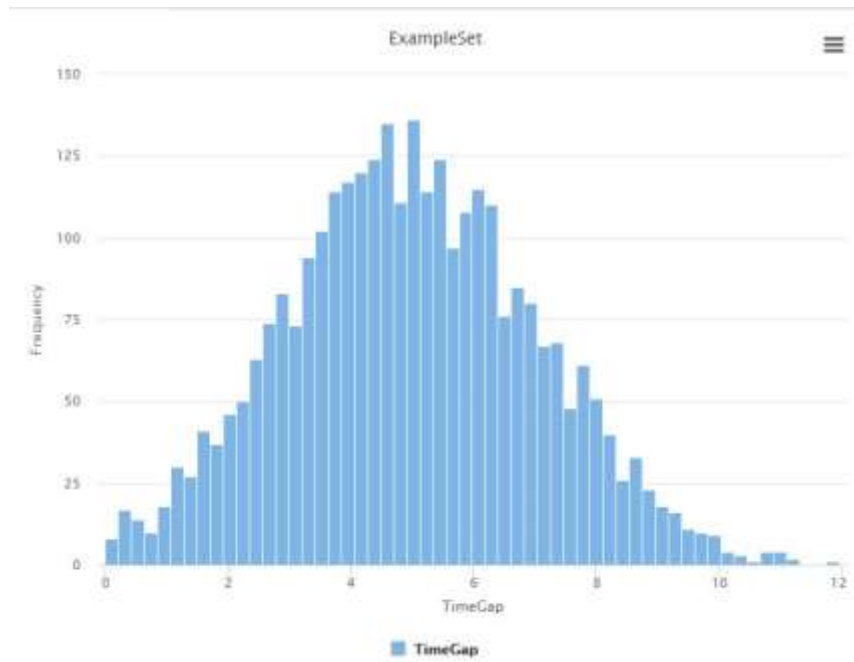
**Figure A-10: Time in hours**



**Figure A-11: Time gap**

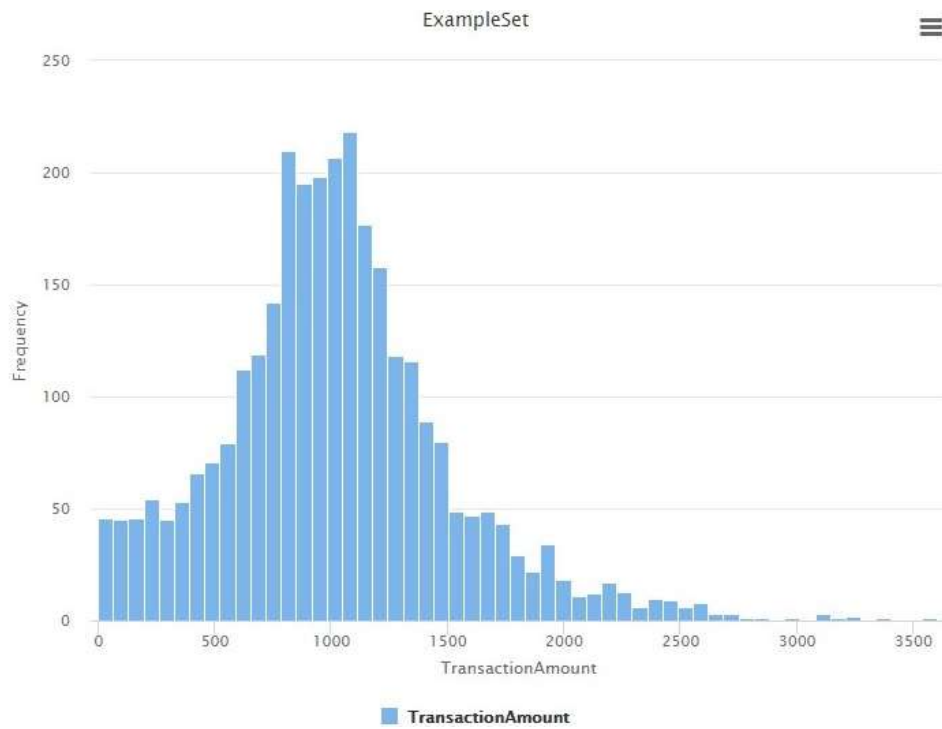
## 4. Case 4

**Figure A-12: Transaction Amount****Figure A-13: Time in hours**

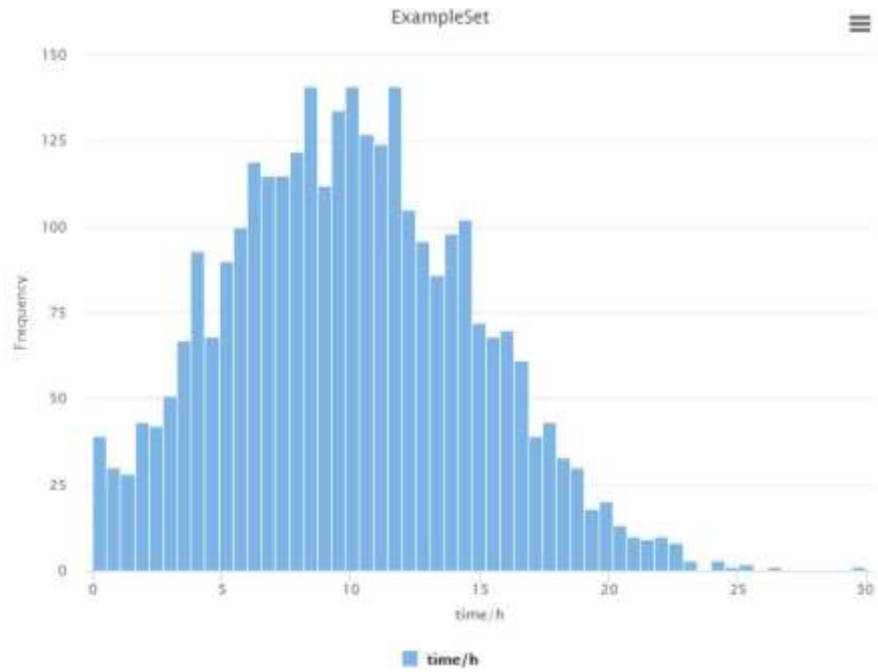


**Figure A-14: Time gap**

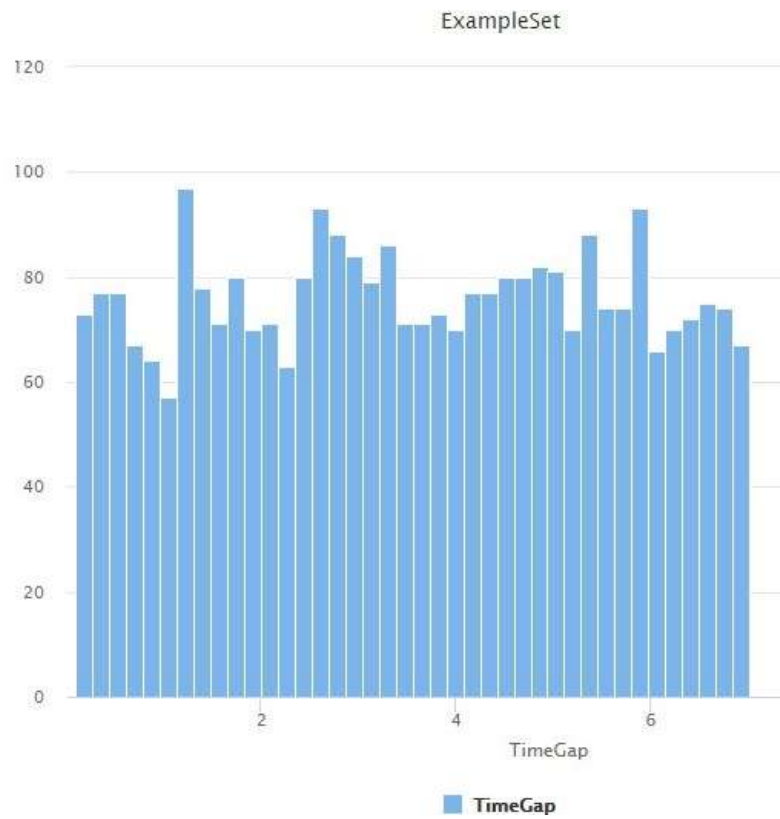
5. Case 5



**Figure A-15: Transaction amount**



**Figure A-16: Time in hours**



**Figure A-17: Time gap**

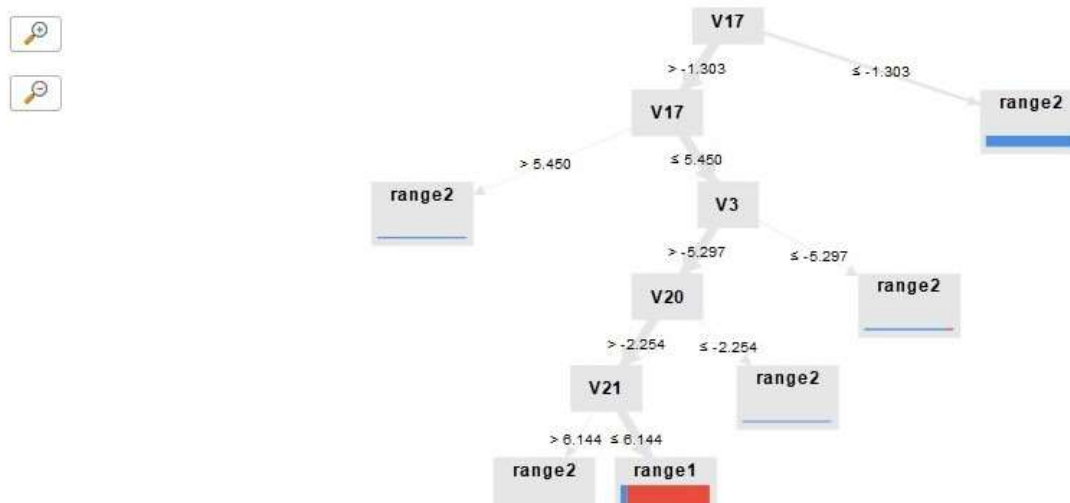
### A.3 Machine Learning Algorithms Experiment on the Real Dataset (1<sup>st</sup> Experiment)

#### A.3.1 Decision Tree (1<sup>st</sup> experiment)

The Decision Tree traces the path from the root node to the leaf node. A classification rule is obtained to be used for further analysis.

The following figure shows the constructed decision tree after applying the steps to the preprocessed real dataset.

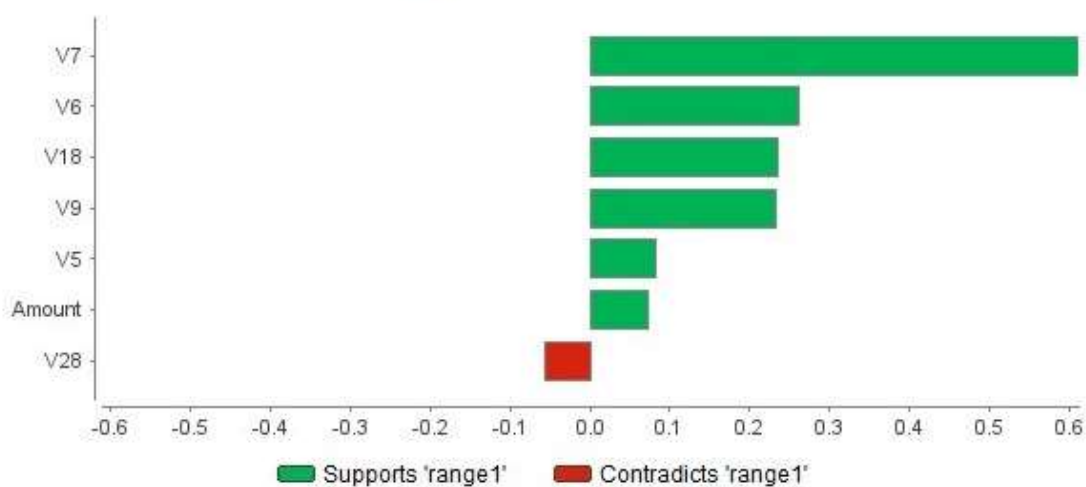
#### Decision Tree - Model



**Figure A-18:** The constructed decision tree model

By applying the Decision Tree fraud detection algorithm on the real dataset, we found that some variables have supported the fraudulent transaction's prediction. Some variables contradict the prediction. Figure A-19 explains the effect of each variable.

### Important Factors for range1



**Figure A-19:** Decision tree correlation factors

This model has excellent performance compared with other used machine learning algorithms such as the SVM. The Decision Tree can obtain results faster than SVM. We used the model performance that consists of the model's prediction accuracy and other performance criteria based on the type of classification problem. The performance is computed on a 40% holdout set based on the default setup for SVM as Rapidminer studio recommended, which has not been used for any performed model optimizations (optimizing the tree depth and minimal leaf size). The most comprehensive and the highest performance are removed, and the remaining performances are reported in Figure A-20. Although this validation is not as precise as full cross-validation, this approach strikes a good balance between runtime and model validation quality, such as automated data slicing for model validation [61].

## Decision Tree - Performance

### Performances

Criterion	Value	Standard Deviation
Accuracy	93.9%	± 1.5%
Classification Error	6.1%	± 1.5%
AUC	86.7%	± 3.6%
Precision	93.9%	± 1.9%
Recall	99.4%	± 0.6%
F Measure	96.5%	± 0.9%
Sensitivity	99.4%	± 0.6%
Specificity	59.1%	± 7.8%

	True Fraudulent	True Legitimate	Class precision
Pred. Fraudulent	80	5	94.12%
Pred. Legitimate	56	865	93.85%
Class recall	58.82%	99.42%	

**Figure A-20:** Decision Tree performance and confusion matrix

Figure A-20 shows both the performance and the confusion matrix of the Decision Tree on the real dataset.

This model has shown good performance in credit card fraud detection by obtaining 93.9% accuracy.

### A.3.2 Support Vector Machine (SVM) (1<sup>st</sup> experiment)

We have used the Support Vector Machine model for fraud detection because our approach is based on supervised learning.

By applying the SVM model, 1880 vectors have been produced. Range1 support vectors (legitimate transactions) were 1619, while 261 vectors were produced for Range2 (fraudulent transactions). The following figures show the SVM model (Kernel Model) structure.

## Support Vector Machine - Model

## Kernel Model

Total number of Support Vectors: 1880  
Bias (offset): -0.725

w[Time] = 73687251.288  
w[V1] = -1540.515  
w[V2] = 1217.595  
w[V5] = -1009.214  
w[V6] = -582.653  
w[V7] = -1807.493  
w[V8] = 332.797  
w[V9] = -888.182  
w[V13] = -50.944  
w[V15] = -3.804  
w[V18] = -770.605  
w[V19] = 227.469  
w[V20] = 142.413

w[V13] = -50.944  
w[V15] = -3.804  
w[V18] = -770.605  
w[V19] = 227.469  
w[V20] = 142.413  
w[V21] = 235.878  
w[V23] = -63.001  
w[V24] = -26.274  
w[V25] = 33.348  
w[V26] = 24.779  
w[V27] = 92.794  
w[V28] = 43.575  
w[Amount] = 91377.264

number of classes: 2  
number of support vectors for class range1: 1619  
number of support vectors for class range2: 261

Figure A-21: Kernel model

Figure A-22 shows each variable's influence in the real dataset on the fraudulent transaction prediction.

## Important Factors for range1

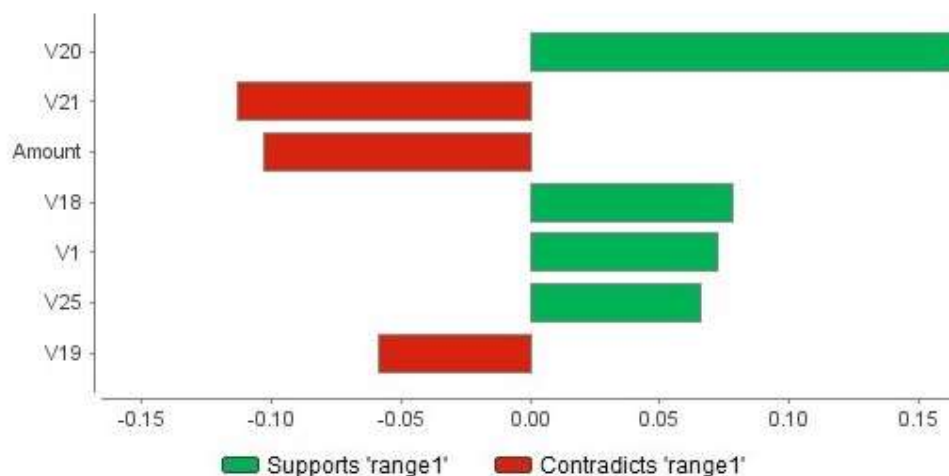


Figure A-22: SVM factors of prediction

The Support Vector Machine model has not shown a good performance compared to other ML models in predicting fraudulent transactions. There are some more measurements to evaluate this model, such as accuracy, precision, and recall percentages.



As in Figure A-23, SVM performs very poorly compared to the decision tree by showing 86.2% accuracy, precision ratios. Although the F-Measure metric shows 92.6%, still the other models have performed better than SVM.

### Support Vector Machine - Performance

Criterion	Value	Standard Deviation
Accuracy	86.2%	± 1.0%
Classification Error	13.8%	± 1.9%
AUC	58.2%	± 5.4%
Precision	86.2%	± 1.0%
Recall	100.0%	± 0.0%
F Measure	92.6%	± 1.1%
Sensitivity	100.0%	± 0.0%
Specificity	0.0%	± 0.0%

	True Fraudulent	True Legitimate	Class precision
Pred. Fraudulent	0	0	0.00%
Pred. Legitimate	138	861	85.16%
Class recall	0.00%	100.00%	

**Figure A-23:** SVM performance and confusion matrix

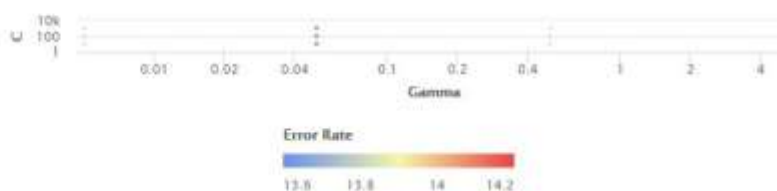
As in Figure A-24, the SVM model has shown bad results in detecting fraudulent credit card transactions with an approximately 13.8% error.

### Support Vector Machine - Optimal Parameters

#### Optimal Parameters

Gamma: 0.005  
C: 10

#### Error Rates for Parameters



**Figure A-24:** SVM error rates for parameters

The SVM model helps us differentiate fraudulent transactions from legitimate transactions by setting up a threshold line. For SVM, a high value of the parameter Gamma leads to more accuracy but biased results and vice-versa. Similarly, a significant value of Cost parameter (C) indicates poor accuracy but low bias and vice-versa. A large C gives you low bias and high variance. Low bias because you penalize the cost of misclassification. As a result of applying SVM on the real dataset, we found that the optimal parameters are kernel gamma= 0.005 and C= 10. The following table shows different C and kernel gamma values and the error percentage corresponding to each value.

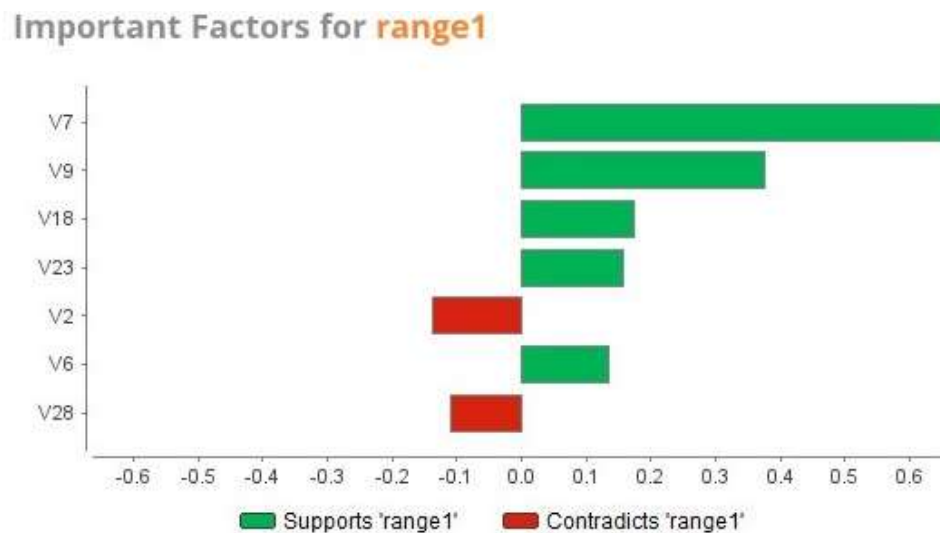
**Table A-1:** SVM Error Rate

<b>Gamma (RBF)</b>	<b>C</b>	<b>error</b>
0.005	10	13.7%
0.050	10	14.2%
0.500	10	13.7%
5	10	13.8%
0.005	100	13.7%
0.050	100	14.2%
0.500	100	13.7%
5	100	13.8%
0.005	1000	13.7%
0.050	1000	14.2%
0.500	1000	13.7%
5	1000	13.8%

Also, we need to figure out the values for true positive, true negative, false positive, and false negative. These values represent the model efficiency in detecting fraudulent transactions, and we can evaluate this model upon the result of the SVM model's confusion matrix. See Figure A-25. Note that range1 represents legitimate transactions, and range2 represents fraudulent transactions.

### A.3.3 Random Forest (1<sup>st</sup> experiment)

Figure A-25 shows the correlation of the features to the prediction result.



**Figure A-25:** Random forest factors for prediction

### Random Forest - Performance

Criterion	Value	Standard Deviation
Accuracy	94.6%	± 1.8%
Classification Error	5.4%	± 1.8%
AUC	89.5%	± 4.8%
Precision	94.5%	± 1.6%
Recall	99.5%	± 0.5%
F Measure	96.9%	± 1.0%
Sensitivity	99.5%	± 0.5%
Specificity	62.9%	± 7.8%

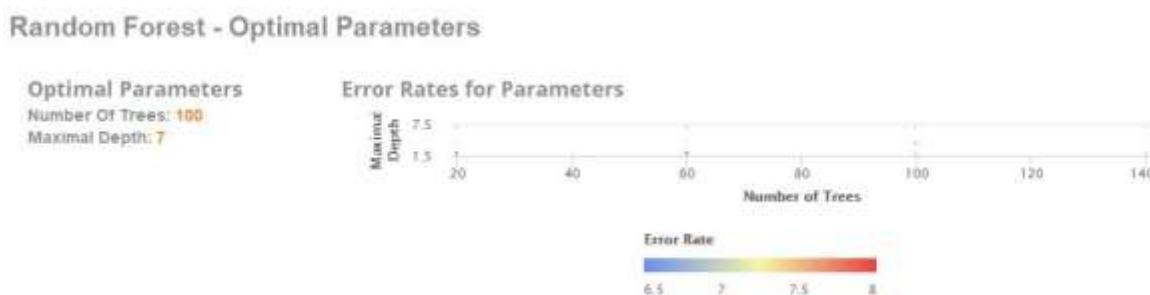
  

	True Fraudulent	True Legitimate	Class precision
Pred. Fraudulent	84	4	95.45%
Pred. Legitimate	50	860	94.51%
Class recall	62.69%	99.54%	

**Figure A-26:** Random forest performance and confusion matrix

As in Figure A-26, this model has performed the best in accuracy and precision ratios.

The RF model has built 100 trees with a maximal depth of seven. The model forecasting shows 5.4% error rates for parameters, as the following figure indicates.



**Figure A-27:** RF error rate

#### A.3.4 Logistic Regression (1<sup>st</sup> experiment)

The Logistic Regression algorithm uses both the LR function and the sigmoid function to present a binary classification based on the dataset's various factors. The sigmoid function is shown below [62]:

$$Y(z) = \frac{1}{1+e^{-z}} \quad \text{Eq. A-1}$$

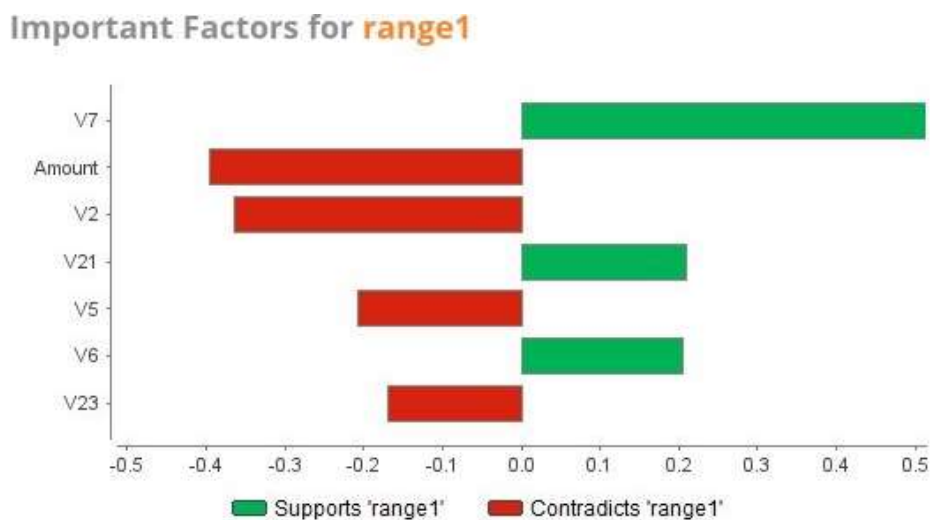
The Sigmoid function is used to find a binary classification probability. In this equation, y represents the probability of the output, and z represents the input to the function.

$$z = b + m_1x_1 + m_2x_2 + \dots + m_nx_n, \quad \text{Eq. A-2}$$

Where b is the linear regression intercept and m is the weighted values and bias, and x is the values featured. We have used the LR model because it presents a binary classification of either 1 or 0, fraudulent transactions and non-fraudulent transactions.

This model uses a threshold of 0.5, and any value higher than this threshold is considered

1, and any value lesser than a threshold of 0.5 is automatically considered 0 [70]. Figure A-28 shows the correlation analysis of features with the prediction results.



**Figure A-28:** Important factors for LR model

### Logistic Regression - Performance

Criterion	Value	Standard Deviation
Accuracy	96.4%	± 0.8%
Classification Error	3.6%	± 0.8%
AUC	93.5%	± 5.5%
Precision	96.8%	± 0.5%
Recall	99.1%	± 0.8%
F Measure	97.9%	± 0.5%
Sensitivity	99.1%	± 0.8%
Specificity	78.5%	± 5.7%

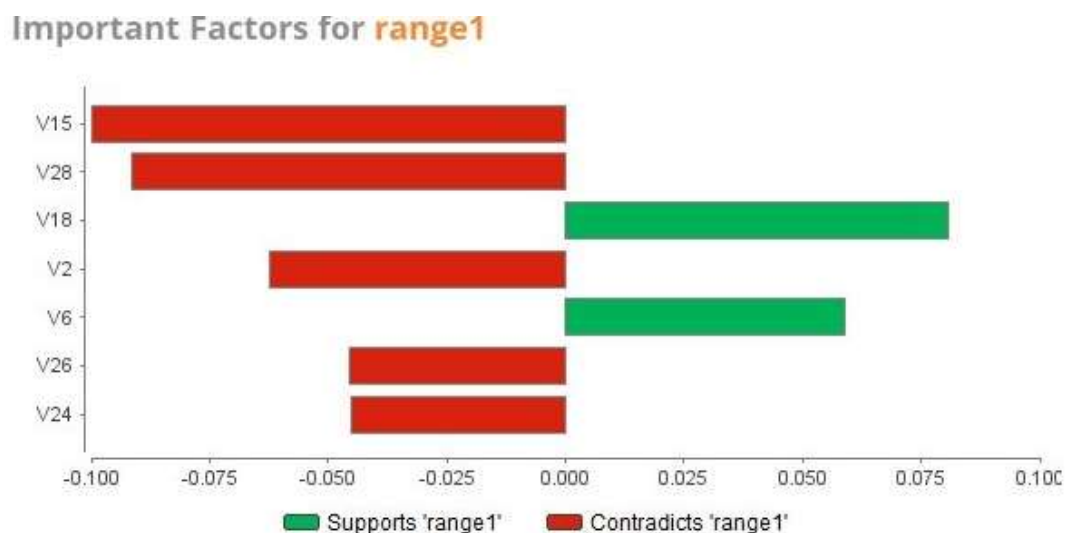
	True Fraudulent	True Legitimate	Class precision
Pred. Fraudulent	106	8	92.98%
Pred. Legitimate	28	856	96.83%
Class recall	79.10%	99.07%	

**Figure A-29:** LR performance and confusion matrix

As shown in Figure A-29, so far, this model has performed the best in terms of accuracy, precision, and F-measure percentages.

### A.3.5 Naïve Bayes (1<sup>st</sup> experiment)

The Naïve Bayes classifier is also a way to distinguish fraudulent credit card transactions. Figure A-30 shows the correlation analysis of features with the prediction results.



**Figure A-30:** Naïve Bayes important factors

Table A-31 shows the performance and confusion matrix of Naïve Bayes on the real dataset. As shown in the table, this model is one of the weakest performances among all tested ML despite a good precision ratio.

## Naive Bayes - Performance

Criterion	Value	Standard Deviation
Accuracy	89.6%	± 0.8%
Classification Error	10.4%	± 0.8%
AUC	90.8%	± 2.1%
Precision	96.7%	± 1.6%
Recall	91.1%	± 1.6%
F Measure	93.8%	± 0.6%
Sensitivity	91.1%	± 1.6%
Specificity	80.4%	± 8.1%

	True Fraudulent	True Legitimate	Class precision
Pred. Fraudulent	109	77	58.60%
Pred. Legitimate	27	784	96.67%
Class recall	80.15%	91.06%	

**Figure A- 31:** Naïve Bayes performance and confusion matrix

### A.4 Machine Learning Algorithms Experiment on the Artificial Datasets (2<sup>nd</sup> Experiment)

This section will integrate the most effective machine learning algorithms applied to the real dataset in the first experiment: Decision Tree, Logistic Regression, and Random Forest.

#### A.4.1 Decision tree (2<sup>nd</sup> experiment)

By applying the DT fraud detection algorithm on the first case dataset, we found the following results: accuracy, precision, recall percentages, as shown in Table A-2.

**Table A-2: Decision Tree Performance (2<sup>nd</sup> experiment)**

<b>Case/ criteria</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Case 1</b>	85.1%	0.825%	85.1%	100%	91.8%
<b>Case 2</b>	87.6%	0.581	87.6%	79.7%	83.1%
<b>Case 3</b>	88.8%	0.668	88.6%	100%	93.9%
<b>Case 4</b>	86.8%	0.893	86.8%	100%	92.8%
<b>Case 5</b>	90.8%	0.342	90.2%	100%	94.8%
<b>Case 6</b>	83.5%	0.54	85.2%	97.7%	90.9%

The performance table shows that the obtained average accuracy of all cases was 85.6%, and approximately the same percentage for the precision. The Decision Tree model shows consistency in performance with all cases in this experiment, no matter the users' consumption behavior participating in this experiment.

We can notice that all metrics ratios decrease in Cases 5 and 6 according to the nonexistence of clear patterns. Note that the F-measure shows an average of 90.3%. Due to the shallow data in these synthetic datasets, the F-measure represents the performance much better than the accuracy because it reflects the false-positive, false-negative in these cases, which is more necessary in this experiment.

#### **A.4.2 Random Forest (2<sup>nd</sup> experiment)**

Table A-3 summarizes the performance of Random Forest on the artificial dataset. The Random Forest has behaved differently with each case, as shown in the table. Overall, this model has achieved a 93.7% F-measure and 90.3% accuracy with 92.8% precision as an average of all case performances. The precision metric and F-Measure



represent the model's performance more accurately than the accuracy metric due to the small size of the synthetic datasets compared with the real dataset that comprises 284,804 dataset size. Thus, this model has performed well but not as well as the Logistic Regression's performance.

**Table A-3: Random Forest Performance (2<sup>nd</sup> Experiment)**

<b>Case/ criteria</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Case 1</b>	91.5%	0.951	97.1%	91.9%	94.4%
<b>Case 2</b>	92.7%	0.744	92.7%	90.9%	91.4%
<b>Case 3</b>	90.0%	0.703	89.9%	100%	94.6%
<b>Case 4</b>	92.4%	0.962	97.5%	93.4%	95.3%
<b>Case 5</b>	90.8%	0.672	93.3%	100%	96.4%
<b>Case 6</b>	84.5%	0.765	86.5%	96.7%	91.2%

The worst obtained performance was in Case 6 with 91.2% F-measure and 84.5% accuracy because this case doesn't show a clear pattern for each user due to all users having a more scattered spending behavior in Case 6 than the normal spending behavior's users. RF provides features importance, but it does not provide complete visibility into the coefficients as linear regression. Still, there are more metrics to judge the model's performance. Even when the ML algorithm predicts a high accuracy, our model is also susceptible to other error types.

Our study classifies an online credit card transaction, whether it is fraudulent (the positive class) or a non-fraudulent transaction (the negative class). While 99% of the time, the credit card transaction is non-fraudulent, possibly 1% of the time, it is fraudulent. If we train a machine learning model and always predict a transaction as non-

fraudulent (negative class), it would be accurate 99% despite never catching the fraudulent transactions (positive class).

F-measure, precision, and recall metrics are helpful to evaluate the performance of the ML model in our scenario because they provide a percentage of the ML model prediction of catching the fraudulent transactions (positive class). Herein, precision is a measure of how often the ML predicts the positive class as true. The recall is the measure of how often the actual positive class is predicted as such. Therefore, a low precision situation appears when very few of our positive predictions are actually true. A low recall percentage occurs when most of your positive values are never predicted.

The Random Forest model has obtained an 86.5% precision ratio in Case 6, which better represents the model's performance. Furthermore, the F-measure shows an average of 91.2%, which means that this model has produced a few false positives and negatives.

#### **A.4.3 Logistic Regression (2<sup>nd</sup> experiment)**

The Logistic Regression model is one of the best models used in our study because it presents a binary classification of the cases. As discussed before in chapter 2, Logistic Regression sets up a threshold of 0.5 between all cases, and then it starts to change this threshold based on the distribution of both classes (0, 1).

Table A-4 summarizes the performance of Logistic Regression on the artificial dataset. The Logistic Regression model has performed the best among all ML models. The Logistic Regression shows great performance in predicting the regular user's behavior. On the other hand, this model shows a better performance in predicting both Cases 5 and 6 than other ML models, but not as good as the simple cases. LR model

shows a good performance with other cases due to the consistency of the user's behavior in the other cases.

Logistic Regression has obtained 94.5% average accuracy, with a slightly higher percentage for precision with 96% precision. We can conclude that this model showed the best performance among other used ML models.

**Table A-4: Logistic Regression Performance (2<sup>nd</sup> Exp)**

<b>Case/ criteria</b>	<b>Accuracy</b>	<b>AUC</b>	<b>Precision</b>	<b>Recall</b>	<b>F-Measure</b>
<b>Case 1</b>	98.9%	0.6%	98.8%	100%	99.4%
<b>Case 2</b>	97.5%	0.973	97.2%	100%	98.6%
<b>Case 3</b>	93.8%	0.89	93.4%	100%	96.5%
<b>Case 4</b>	97.2%	0.972	98.8%	97.7%	98.2%
<b>Case 5</b>	94.0%	0.747	98.2%	95.0%	96.4%
<b>Case 6</b>	85.5%	0.866	89.4%	94.4%	91.6%

#### **A.4.4 Time Efficiency of the 2<sup>nd</sup> Experiment**

Table A-5 summarizes each ML algorithms' time efficiency on each case of the artificial dataset. In the table, we observe that the machine learning algorithms behave differently with different inputs and variables. Due to the variation of inputs in each case in the second experiment, machine learning spends more time processing data in the more complex cases because of the difficulty of guessing each user's pattern in each dataset. Hence, the more complex the case is, the more time-consuming the model becomes. The Random Forest model's complexity justifies a long time in the processing since it needs to construct many decision trees according to the input variables. Therefore, it has the worst time efficiency among other models.

**Table A-3: Time Comparison (2<sup>nd</sup> Exp)**

<b>Case 1</b>			
<b>Model/Time</b>	<b>Training</b>	<b>scoring</b>	<b>total</b>
<b>Decision tree</b>	67 ms	295 ms	1 s
<b>Logistic regression</b>	216 ms	106 ms	1 s
<b>Random forest</b>	82 ms	303 ms	4 s
<b>Case 2</b>			
<b>Model/Time</b>	<b>Training</b>	<b>scoring</b>	<b>total</b>
<b>Decision tree</b>	104 ms	116 ms	2 s
<b>Logistic regression</b>	239 ms	161 ms	1s
<b>Random forest</b>	96 ms	366 ms	4 s
<b>Case 3</b>			
<b>Model/Time</b>	<b>Training</b>	<b>scoring</b>	<b>total</b>
<b>Decision tree</b>	104 ms	125 ms	1 s
<b>Logistic regression</b>	179 ms	71 ms	898 ms
<b>Random forest</b>	136 ms	304 ms	4 s
<b>Case 4</b>			
<b>Model/Time</b>	<b>Training</b>	<b>scoring</b>	<b>total</b>
<b>Decision tree</b>	142 ms	141 ms	3 s
<b>Logistic regression</b>	118 ms	174 ms	3 s
<b>Random forest</b>	196 ms	2 s	22 s
<b>Case 5</b>			
<b>Model/Time</b>	<b>Training</b>	<b>scoring</b>	<b>total</b>
<b>Decision tree</b>	121 ms	123 ms	1 s
<b>Logistic regression</b>	186 ms	66 ms	959 ms
<b>Random forest</b>	178 ms	377 ms	4 s
<b>Case 6</b>			
<b>Model/Time</b>	<b>Training</b>	<b>scoring</b>	<b>total</b>
<b>Decision tree</b>	96 ms	160 ms	2 s
<b>Logistic regression</b>	127 ms	97 ms	3 s
<b>Random forest</b>	89 ms	257 ms	4 s

### A.5 Artificial Datasets Generation Code

We have used Python to generate artificial datasets with different constraints to differentiate the six cases. The following is the Python code.

```

import csv
import random
import string
import pandas as pd
from random import randrange
from datetime import timedelta
from datetime import datetime
data = pd.read_csv("uscities.csv")
print(data['city'][0])
print(data['lat'][0], data['lng'][0])
# Number of instances
instances = 150000
# Function for .....
def random_date(start, end):
    delta = end - start
    int_delta = (delta.days * 24 * 60 * 60) + delta.seconds
    random_second = randrange(int_delta)
    return start + timedelta(seconds=random_second)
#Function for .....
def random_char(y):
    return ''.join(random.choice(string.ascii_letters) for x in
range(y))
#print (random_char(7)+"@gmail.com")
#=====
# generate 100 IDs
IDs = []
for i in range(instances):
    IDs.append(i)
#print(IDs)
#=====
# generate 100 IDs
userNames = []
for i in range(instances):
    userNames.append(random_char(7)+"@gmail.com")
#print(userNames)
#=====
# Create Accounts numbers
accountNums = []
for i in range(instances):
    accountNums.append(i)
#print(accountNums)
stores = ["Walmart", "Target", "Amazon", "eBay"]
#Ips
Ips = []
for i in range(instances):
    Ips.append('.'.join('%s'%random.randint(0, 255) for i in
range(4)))
#=====
dl = datetime.strptime('1/1/2020 12:00 AM', '%m/%d/%Y %I:%M %p')
from datetime import datetime, timedelta
nine_hours_from_now = datetime.now() + timedelta(hours=9)
print (nine_hours_from_now)
nine_hours_from_now = datetime.now() + timedelta(days=9)
print (nine_hours_from_now)

```

```

    while start < stop:
        yield float(start)
        start += decimal.Decimal(step)
#=====
import random
from functools import reduce
#1,804 total
def gen_avg(expected_avg=89, n=3, min=1, max=3000):
    while True:
        l = [random.randint(min, max) for i in range(n)]
        avg = reduce(lambda x, y: x + y, l) / len(l)

        if avg == expected_avg:
            return l
with open('ccFraudDataSet.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=',', quotechar='|',
quoting=csv.QUOTE_MINIMAL)
    writer.writerow(["UserID", "UserAccountNumber", "UserName",
"TransactionTime", "TransactionAmount", "TransactionStore",
"TransactionIP", "latitude", "longitude", "Status"])
    total = 0
    for i in range(instances):
        # Constants
        print("\users: " + str(i))
        uID = random.choice(IDs)
        index = IDs.index(uID)
        maxNoTransactions = 3
        for j in range(random.randint(1, maxNoTransactions)):
            # Variables
            uName = userNames[index]
            latitude = data['lat'][index]
            longitude = data['lng'][index]
            tTime = random_date(d1, d2)
            uAccNum = accountNums[index]
            tStore = random.choice(stores)
            tAmount = random.choice(gen_avg())
            total = total + 1
            print("# of transactions " + str(total))
            tIp = Ips[index]
            writer.writerow([uID, uAccNum, uName, tTime, tAmount,
tStore, tIp, latitude, longitude, '0'])

```

## A.6 One-Time Credit Card Generation Full Code

This is the full code that generates the one-time credit card numbers using PHP.

```

<?php
require "connection.php";
session_start();
function getRandomNumber($len = "5")// 4 random credit card number (user identifier)
function
{
    $better_token= str_pad(rand(0, 99999), $len, '0', STR_PAD_LEFT);
    return $better_token;
}
function getRandomCVV($len = "3")// generate random CVV number function
{
    $better_token= str_pad(rand(0, pow(10, $len)-1), $len, '0', STR_PAD_LEFT);
    return $better_token;
}
// Luhn algorithm to generate the check number which is the last digit of the CC number
function Luhn($gen_num) {
    $stack = 0;
    $gen_num = str_split(strrev($gen_num));
    foreach ($gen_num as $key => $value)
    {
        if ($key % 2 == 0)
        {
            $value = array_sum(str_split($value * 2));
        }
        $stack += $value;
    }
    $stack %= 10;

    if ($stack != 0)
    {
        $stack -= 10;    $stack = abs($stack);
    }
    $gen_num = implode("", array_reverse($gen_num));
    $gen_num = $gen_num . strval($stack);

    return $gen_num;
}
function microseconds() {
    $smt = explode(' ', microtime());
    return ((int)$smt[1]) * 1000000 + ((int)round($smt[0] * 1000000));
}

```

```

function getUserIpAddr(){
  if(!empty($_SERVER['HTTP_CLIENT_IP'])){
    //ip from share internet
    $ip = $_SERVER['HTTP_CLIENT_IP'];
  }elseif(!empty($_SERVER['HTTP_X_FORWARDED_FOR'])){
    //ip pass from proxy
    $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
  }else{
    $ip = $_SERVER['REMOTE_ADDR'];}
  return $ip;}
function getMAC (){
  $d = explode('Physical Address. . . . .',shell_exec ("ipconfig/all"));
  $d1 = explode(':', $d[1]);
  $d2 = explode(' ', $d1[1]);
  return $d2[1];}
function chain($x){
  return hash('sha256', $x);}
$iss_identifier = 123456;// credit card issuer identifier
$amount_1= rand (1,1000);// transaction amount
$currtime=microseconds();// time in microseconds
$rand_5=getRandomNumber();// random 5-digit
$zz= $amount_1*$currtime;
$yy= intval(($amount_1 * $currtime)/($rand_5));
$com_num= substr($yy,-4);
$cc_hash= hash('sha256', $rand_5*$amount_1+$currtime);
$int = filter_var($cc_hash, FILTER_SANITIZE_NUMBER_INT);
$num_salt= substr($int,-5);
$format= '%d%d%d';
$gen_num= sprintf($format, $iss_identifier, $num_salt,$com_num);
$luhn= Luhn($gen_num);
$luhn_num= substr($luhn,-1);// Luhn digit
echo "This is the random number generated at the user's side = $rand_5";
$x= hash('sha256', "password.$zz");
$g= $rand_5;
  while ($g!= 0){
    $x= chain($x);
    $g--;}
echo "<br>";
echo "This is the hashed code to the power of the random number = $x";
echo "<br>";
echo "<br>";
$secret=hash('sha256', "password.$zz");
for($j=0; $j<99999;$j++){
  if ($x == $secret){echo "This is the power value that found by the server which matched
the generated random value at the user's side =$j". "<br>";
  $rr= hash('sha256', $j*$amount_1+$currtime);
  $int2 = filter_var($rr, FILTER_SANITIZE_NUMBER_INT);
  $xyz= $iss_identifier.substr($int2,-5).$com_num.$luhn_num;
  echo "the obtained number in server is: $xyz" . "<br>";}
  $secret= chain($secret);}

```



```

$user_MAC = getMAC();
$user_ip = getUserIpAddr();
$time_curr= date("Y-m-d H:i:s");
$scvv= substr((getRandomCVV()*$currtime),-3);
$ccnum= $iss_identifier. $num_salt.$com_num. $luhn_num;
$mysqli = new mysqli("localhost", "root", "", "register");
$result = $mysqli->query("SELECT id FROM exp1 WHERE ccNumber = '$ccnum'");
if($result->num_rows == 0) {
    echo "row not found";
    echo "<br>";
    echo "Verified";
    echo "<br>";
    echo "credit card number: $ccnum";
    echo "<br>";
    echo "CVV number: $scvv";
} else {
    echo "row found";
    echo "<br>";
}
$sql_new= "INSERT INTO exp1 (id, ccNumber, CVV, time_curr, time_milli, time_mod,
amount,time_amount, comb_num, IP, MAC, hashed_key)
VALUES
(',$ccnum','$scvv','$time_curr','$currtime','$rand_5','$amount_1','$yy','$com_num','$user_ip','$user_MAC','$x')";
$query=mysqli_query($link,$sql_new);
    $mysqli = new mysqli("localhost", "root", "", "register");
    $id= mysqli_insert_id($link);
    echo "<br>";
    echo $id;
    mysqli_close($link);
echo "<br>";
echo "<br>";
?>

```

## REFERENCES

- [1] Britannica, The Editors of Encyclopaedia. Credit card. *Encyclopedia Britannica*, 4 Mar. 2021, <https://www.britannica.com/topic/credit-card>
- [2] C. Silver, Biggest credit card fraud in history. *Investopedia*. <https://www.investopedia.com/capital-one-reveals-massive-hack-exposing-millions-4707235>
- [3] Nilson Report, Reuters reported lost money due to credit card fraud. *Cission PR Newswire*. <https://www.prnewswire.com/news-releases/payment-card-fraud-losses-reach-27-85-billion-300963232.html>
- [4] M. Krishna, The second biggest cybercrime in history. *Investopedia*. <https://www.investopedia.com/news/5-biggest-credit-card-data-hacks-history/>
- [5] M. Jewell, TJX breach could top 94 million accounts. *NBC NEWS*. [http://www.nbcnews.com/id/21454847/ns/technology\\_and\\_science-security/t/tjx-breach-could-top-million-accounts/#.XldJ3GhKjIU](http://www.nbcnews.com/id/21454847/ns/technology_and_science-security/t/tjx-breach-could-top-million-accounts/#.XldJ3GhKjIU)
- [6] Nilson Report, Card fraud losses reach \$27.95 billion. <https://nilsonreport.com/mention/407/1link/>
- [7] Shift Credit Card Processing, Credit card fraud statistics. Jan. 2021. <https://shiftprocessing.com/credit-card-fraud-statistics/>
- [8] J. Kiernan, Credit card and debit card fraud statistics. *Wallethub*, 2 Feb 2017. <https://wallethub.com/edu/cc/credit-debit-card-fraud-statistics/25725/>
- [9] K. Chaudhary, J. Yadav, and B. Mallick, "A review of fraud detection techniques: Credit card." *International Journal of Computer Applications*, vol. 45, no. 1, pp. 39-44. 2012.
- [10] J. Pieprzyk, H. Ghodosi, and E. Dawson, "Information security and privacy," *Proceedings of 12th Australasian Conference*, vol. 4586, pp. 1-11. Springer. 2007.
- [11] P. Richhariya and P. Singh, "A survey on financial fraud detection methodologies," *International Journal of Computer Applications*, vol. 45, no. 22, pp. 15-22. 2012.

- [12] L. Delamaire, A. Hussein, and J. Pointon, "Credit card fraud and detection techniques: A review." *Banks and Bank Systems*, vol 4, no.2, pp. 57-68. 2009
- [13] Pago reports the successful year 2005, *Businesswire*.  
<https://www.businesswire.com/news/home/20060130005607/en/Pago-Reports-Successful-Year-2005-International-Acquiring>
- [14] K. Joshi, "A review of credit card fraud detection techniques in e-commerce," *Academic Journal of Forensic Sciences*, vol. 01, no. 01, 2018.
- [15] C. Phua, R. Gayler, V. Lee, and K. Smith, "On the approximate communal fraud scoring of credit applications." *Proceedings of Credit Scoring and Credit Control*, pp.1-10. 2005.
- [16] A. Saxena and H. Ponnappalli, *U.S. Patent Application No. 13/109,946*. 2012.
- [17] S. Meredith, D. Kent, D. Patterson, and E. Abrahamian. "Fraud detection via mobile device location tracking." *U.S. Patent No. 9,858,575*. 2 Jan. 2018.
- [18] S. Rajasekaran, and R. Varadarajan, *U.S. Patent No. 6,908,030*. Washington, DC: U.S. Patent and Trademark Office. 2005.
- [19] J. Lynam, E. Meyer, M. Snycerski, and B. Singer, *U.S. Patent Application No. 13/841,318*. 2014.
- [20] P. Bentley, J. Kim, G. Jung, and J. Choi, Fuzzy Darwinian detection of credit card fraud. In the *14th Annual Fall Symposium of the Korean Information Processing Society*, vol. 14, 2000.
- [21] T. Behera, and S. Panigrahi, "Credit card fraud detection: A hybrid approach using fuzzy clustering & neural network." In 2015 Second International Conference on Advances in Computing and Communication Engineering, pp. 494-499. *IEEE*.2015.
- [22] P. Save, P. Tiwarekar, K. Jain, and N. Mahyavanshi, "A novel idea for credit card fraud detection using a decision tree." *International Journal of Computer Applications*, vol. 161, no. 13. 2017.
- [23] J. Essebag, S. Pochic, and C. Lalo, *U.S. Patent No. 10,032,169*. Washington, DC: U.S. Patent and Trademark Office. 2018.
- [24] J. Ashfield, *U.S. Patent No. 9,251,637*. Washington, DC: U.S. Patent and Trademark Office. 2016.
- [25] N. Patel, *U.S. Patent Application No. 13/311,262*. 2012.

- [26] G. McDonald, *U.S. Patent Application No. 12/408,325*. 2010.
- [27] P. Barbour, 2004, "Method and system for conducting financial transactions using single-use credit card numbers." *Patent number: US20040133507A1*. 2004.
- [28] S. Gupta and R. Johari, "A new framework for credit card transactions involving mutual authentication between cardholder and merchant," *2011 International Conference on Communication Systems and Network Technologies*, 2011, pp. 22-26, 2011. doi: 10.1109/CSNT.2011.12.
- [29] Y. Li and X. Zhang, "A security-enhanced one-time payment scheme for a credit card," 14th International Workshop Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications, *Proceedings*, Boston, MA, USA, 2004, pp. 40-47, doi: 10.1109/RIDE.2004.1281701.
- [30] N. Trivedi, "An efficient credit card fraud detection model based on machine learning methods." *International Journal of Advanced Science and Technology* vol. 29, no.5. 2020, pp. 3414-3424.
- [31] Nilson Report, "Card fraud losses reach \$28.65 billion," <https://nilsonreport.com/mention/1313/1link/>.
- [32] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naïve Bayes," *Advances In Neural Information Processing Systems*, vol. 2, pp. 841-848, 2002.
- [33] S. Maes, K. Tuyls, B. Vanschoenwinkel and B. Manderick, "Credit card fraud detection using Bayesian and neural networks," *Proceedings of the 1<sup>st</sup> International Naiso Congress on Neuro Fuzzy Technologies*, pp. 261-270, 2002.
- [34] C. Phua, D. Alahakoon and V. Lee, "Minority report in fraud detection: classification of skewed data," *Acm Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 50-59, 2004.
- [35] M. Szmiglera, "Fraud losses per 100 U.S. dollars of total card sales worldwide from 2010 to 2018, with forecasts to 2027," *Statista*, <https://www.statista.com/statistics/1080685/global-card-fraud-losses-forecast/>
- [36] A. Shen, R. Tong and Y. Deng, "Application of classification models on credit card fraud detection," *Service Systems and Service Management* 2007, pp. 1-4, 2007.
- [37] S. Panirahi, A. Kundu, S. Sural and A. Majumdar, "Credit card fraud detection: A fusion approach using Dempster-Shafer theory and Bayesian learning," *Information Fusion*, vol. 10, no. 4, pp. 354-363, 2009.

- [38] Y. Sahin and E. Duman, "Detecting credit card fraud by decision trees and support vector machines," *Proceedings of International Multi-Conference of Engineers and Computer Scientists (IMECS 2011)*, vol. 1, pp. 1-6, Mar. 16-18 2011.
- [39] S. Bhattacharyya, S. Jha, K. Tharakunnel and J. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602-613, 2011.
- [40] Y. Sahin and E. Duman, "Detecting credit card fraud by ANN and logistic regression," *Innovations in Intelligent Systems and Applications*, pp. 315-319, 2011.
- [41] K. Sherly, "A comparative assessment of supervised data mining techniques for fraud prevention," *Int. J. Sci. Tech. Res.*, vol. 1, no. 16, 2012.
- [42] J. Pun and Y. Lawryshyn, "Improving credit card fraud detection using a meta-classification strategy," *International Journal of Computer Applications*, vol. 56, no. 10, 2012.
- [43] O. Adepoju, J. Wosowei, and H. Jaiman. "Comparative evaluation of credit card fraud detection using machine learning techniques." *2019 Global Conference for Advancement in Technology (GCAT)*. IEEE, 2019.
- [44] CapitalOne. What is a virtual card number, and how does it work? *CapitalOne*, <https://www.capitalone.com/learn-grow/money-management/what-are-virtual-card-numbers/>
- [45] G. John, and P. Langley. "Estimating continuous distributions in Bayesian classifiers." *arXiv preprint arXiv:1302.4964* (2013).
- [46] J. Awoyemi, O. Adetunmbi, and S. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis." *2017 International Conference on Computing Networking and Informatics (ICCNI)*. IEEE, 2017.
- [47] J. James, "Bayes' theorem." *Stanford Encyclopedia of Philosophy*, " 28 Jun 2003.
- [48] R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 1-106. 1986.
- [49] S. Kalyan Krishnan, D. Singh, R. Kant, 2014, "On building decision trees from large-scale data in applications of online advertising," *Proceedings of the 23<sup>rd</sup> ACM International Conference on Information and Knowledge Management*, Nov 2014, pp. 669-678.

- [50] J. Gaikwad, A. Deshmane, H. Somavanshi, S. Patil, and R. Badgujar, "Credit card fraud detection using decision tree induction algorithm." *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol.4, no. 6, 2014.
- [51] E. Altam, G. Macro, and F. Varetto, Corporate distress diagnosis: a comparison using linear discriminant analysis and neural networks. *Journal Banking and Finance*, vol. 18, pp. 505-529, 1994.
- [52] Y. Sahin, and E. Duman, "Detecting credit card fraud by ANN and logistic regression." In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pp. 315-319, IEEE. 2011.
- [53] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, and C. Jiang, "Random forest for credit card fraud detection," *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, 2018, pp. 1-6.
- [54] Wikipedia, "Payment card industry data security standard," *JSTOR*, 2017. [https://en.wikipedia.org/wiki/Payment\\_Card\\_Industry\\_Data\\_Security\\_Standard](https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard).
- [55] H. Stevens, "Hans Peter Luhn and the birth of the hashing algorithm." *IEEE Spectrum*, vol. 55, no. 2, pp. 44-49, 2018.
- [56] Kaggle, "Credit Card Fraud Detection Dataset," Mar 2018, [online] Available: <https://www.kaggle.com/mlg-ulb/creditcardfraud>
- [57] RapidMiner, "Imbalanced Data," *Google Developers*, January 2019, <https://developers.google.com/machine-learning/dataprep/construct/sampling-splitting/imbalanced-data>
- [58] RapidMiner, "Studio data preparation," *Google Developers*, 2019, <https://rapidminer.com/glossary/data-preparation/>
- [59] RapidMiner, "Data visualization," *Google Developers*, 2019, <https://rapidminer.com/products/turbo-prep/>
- [60] T. Sabanaglu, "Online shopping frequency according to online shoppers worldwide as of October 2018," <https://www.statista.com/statistics/664770/online-shopping-frequency-worldwide/>
- [61] T. Wu, M. Ribeiro, J. Heer, and D. Weld, "Erudite: Scalable, reproducible, and testable error analysis." *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Pp. 747-763. 2019.
- [62] JavaPoint, "Logistic regression in machine learning," <https://www.javatpoint.com/logistic-regression-in-machine-learning>

- [63] PyCharm, “Install using the toolbox app,” *PyCharm*,  
<https://www.jetbrains.com/help/pycharm/installation-guide.html#toolbox>
- [64] S. Gupta, and R. Johari. “A new framework for credit card transactions involving mutual authentication between cardholder and merchant.” *2011 International Conference on Communication Systems and Network Technologies. IEEE*, 2011.
- [65] S. Lakshmi, and S. Kavilla, “Machine learning for credit card fraud detection system.” *International Journal of Applied Engineering Research* vol. 13, no. 24, pp.16819-16824, 2018.