CREDIT CARD FRAUD DETECTION WITH AUTOENCODERS, ONE-CLASS SVMS AND ISOLATION FORESTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZDEMİR ÖZKUM

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
STATISTICS

DECEMBER 2023

Approval of the thesis:

CREDIT CARD FRAUD DETECTION WITH AUTOENCODERS, ONE-CLASS SVMS AND ISOLATION FORESTS

submitted by **ÖZDEMİR ÖZKUM** in partial fulfillment of the requirements for the degree of **Master of Science in Statistics, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Özlem İlk Dağ
Head of the Department, **Statistics** ________________

Prof. Dr. Özlem İlk Dağ
Supervisor, **Statistics, METU** ________________

Examining Committee Members:

Prof. Dr. Ceylan Yozgatlıgil
Statistics, METU ________________

Prof. Dr. Özlem İlk Dağ
Statistics, METU ________________

Assistant Prof. Dr. Fatma Yerlikaya Özkurt
Industrial Engineering, Atılım University ________________

Date: 06.12.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Özdemir Özkum

Signature :

ABSTRACT

CREDIT CARD FRAUD DETECTION WITH AUTOENCODERS, ONE-CLASS SVMS AND ISOLATION FORESTS

Özkum, Özdemir
Master of Science, Statistics
Supervisor : Prof. Dr. Özlem İlk Dağ

December 2023, 52 pages

Fraud Detection has gained importance in the last few years due to its cost to the economy and the population in general. However, there are a number of important problems in the modeling process to tackle when detecting fraud. One is the availability of data. Since the financial data produced by the customers is subject to privacy rules, working with synthetic data is a necessity. While this is the case, there are very few data sets that reflect the true nature of the data generation and analysis processes of financial transactions. Another problem is that the fraud or any data that require anomaly detection do not have labeled data. Even labeled, the data points are labeled by rules and there is a need to automate this process. This is why it is deemed important to incorporate an unsupervised collection of methods and let the data speak for itself. For these reasons, this study focuses on the unsupervised learning methods in fraud detection using a synthetic data set which consists of credit card transactions. The learning methods used were the Autoencoders, Sparse Autoencoders, One-Class Support Vector Machines and Isolation Forests. For comparison, a Random Forest Model was also built. It was found that among the unsupervised methods One-Class SVM was the best performing model with Precision=0.68, Recall=0.98, and F1

Score=0.81. However, One-Class SVM did not outperform the supervised Random Forest model which achieved Precision=1.00, Recall=0.95, and F1-Score=0.97.

Keywords: Unsupervised Learning, Supervised Learning, Credit Card Fraud, Autoencoders, One-Class SVM, Isolation Forest, Random Forest

ÖZ

OTOKODLAYICILAR, TEK SINIFLI DESTEK VEKTÖR MAKİNELERİ VE İZOLASYON ORMANLARI İLE KREDİ KARTI DOLANDIRICILIĞI TESPİTİ

Özkum, Özdemir
Yüksek Lisans, İstatistik
Tez Yöneticisi: Prof. Dr. Özlem İlk Dağ

Aralık 2023, 52 sayfa

Dolandırıcılık tespiti, ekonomi ve genel popülasyon üzerinde yarattığı maliyetler sebebiyle, son yıllarda önem kazanmıştır. Ancak, dolandırıcılık tespiti yapılırken, modelleme süreçlerinde çözülmesi gereken birkaç problem söz konusudur. Bunlardan birincisi veriye erişebilirliktir. Şöyle ki, müşteriler tarafından üretilen veriler gizlilik kurallarına tabi olduklarından, yapay veri ile bir gerekliliktir. Ancak, finansal işlemlerin veri üretimi ve analiz süreçlerini yansıtan az sayıda veri bulunmaktadır. Dolandırıcılık tespitinde bir diğer nokta ise, verilerin etiketlenmemiş oluşudur. Hatta etiketlenmiş olsalar bile bu işlem kurallar aracılığıyla yürütüldüğünden bu sürecin otomatize edilmesi gerekmektedir. Bu nedenle denetimsiz öğrenme yöntemleri ile verinin kendi adına konuşmasına izin verilmesi önemlidir. Bu çalışma, bahsedilen bu nedenlerle, yapay kredi kartı verisi kullanarak, dolandırıcılık tespitinde kullanılan denetimsiz öğrenme yöntemlerine odaklanmaktadır. Kullanılan öğrenme yöntemleri Otokodlayıcılar, Seyrek Otokodlayıcılar, Tek Sınıflı Destek Vektör Makineleri ve İzolasyon Ormanları olup, karşılaştırma yapmak üzere Rassal Orman modeli de kurulmuştur. Sonuçlar göstermektedir ki denetimsiz öğrenme yöntemleri arasında en iyi performansı Geri Çağırma= 0.98, Kesinlik=0.68 ve F1 Skoru=0.81 olmak üzere Tek Sınıflı Destek

Vektör Makineleri göstermiştir. Buna karşın, Rassal Orman Modeli ise Tek Sınıflı Destek Vektör Makineleri modelinden daha başarılı bir performans göstererek Geri Çağırma=0.95, Kesinlik=1.00 ve F1=0.97 skorlarını elde etmiştir.

Anahtar Kelimeler: Denetimsiz Öğrenme, Denetimli Öğrenme, Kredi Kartı Dolandırıcılığı, Otokodlayıcılar, Bir Sınıflı Destek Vektör Makineleri, İzolasyon Ormanları, Rassal Ormanlar

To my family, for always being there for me.

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my supervisor Prof. Dr. Özlem İlk Dağ for her guidance and advice in the research; and her trust she placed in me throughout my academic studies and career.

I would also like to thank Prof. Dr. Ceylan Yozgatlıgil for the inspiration and energy she brought in my professional career.

I would like to say a special thank you to Ayşegül Binbaş, for always being there for me every step of my studies.

Finally, I wish to extend my gratitude to Onur İnce, a remarkable friend, with whom I studied, discussed and thought about Statistics.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

ACFE	**:** Association of Certified Fraud Examiners
ADASYN	**:** Adaptive Synthetic Sampling
AE	**:** Autoencoder
ARIMA	**:** Autoregressive Integrated Moving Average
FN	**:** False Negative
FP	**:** False Positive
HMM	**:** Hidden Markov Model
K-L	**:** Kullback-Leibler
MCC	**:** Merchant Category Code
MSE	**:** Mean Squared Error
OCSVM	**:** One-Class Support Vector Machine
RELU	**:** Rectified Linear Unit
SAE	**:** Sparse Autoencoder
SMOTE	**:** Synthetic Minority Over-Sampling Technique
SVM	**:** Support Vector Machine
TANH	**:** Hyperbolic Tangent Function
TN	**:** True Negative
TP	**:** True Positive
TSNE	**:** t-Distributed Stochastic Neighbor Embedding

CHAPTER 1

INTRODUCTION

Fraud detection is a big problem and the detection of it will benefit both the economy and the population. However, when it comes to fraud, there are important distinctions to be made to decide on the course of action in the context of modeling. The first one, since the fraud is an umbrella term, is to decide what type of fraud we are working on. This will cause us to change the modeling approach since each fraud type has its own characteristics with respect to data design and data generating process.

The second one is the availability of data. Considerable amount of work on modeling fraud depends on synthetic data due to privacy concerns. However, this is not to say that there is much synthetic data available to work on. On the contrary, there are limited number of data sets that seem to realistically mimic the real world.

The third one is to choose the appropriate modeling method. Although there are many learning methods used for classification, the lack of labeled data and/or the possible biases that might be produced with the human-labeled data force us to give unsupervised methods more weight in fraud detection. Otherwise, we cannot use supervised models and/or when we want to do so we are just modeling the decisioning mechanisms that are already implemented by the humans.

In this study, a commonly-used feature generation mechanism in fraud detection that captures the spending behavior of customers in the last t hours is followed and by leveraging a realistic credit card data set, published by Erik Altman in 2021, a suite of unsupervised learning methods are employed to show how well these methods perform. To the best of our knowledge this is the first study that explores the performance of unsupervised methods on this data set. Since most of the studies conducted on credit card fraud detection with the same data sets, we believe this study will contribute to the area by showing the performance of unsupervised models on a completely different data set.

The rest of this study is organized as follows: Chapter 2 gives information about fraud, data sets used for fraud detection and the learning methods in the literature. Chapter 3 explains the data used in this study and the learning methods to model the fraudulent behaviour in credit card transactions. Chapter 4 presents the numerical results, and Chapter 5 discusses the findings and lays out the possible research areas.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we present the building blocks of credit card fraud detection, which are the definition of fraud, data used for fraud detection and the learning methods commonly employed by researchers and experts by surveying the related literature.

2.1 Definition of Fraud

Association of Certified Fraud Examiners (ACFE) defines fraud as “any activity that relies on deception in order to achieve a gain” [1]. Needless to say, this act is also classified as a crime one way or another. However, there are different types of fraud where the result is the same but the channels are different. According to the National Fraud & Cyber Crime Center of the UK, the types of fraud include but not limited to

- Credit Card Fraud
- Insurance Fraud
- Customer Fraud

all of which have different characteristics [2].

There are also other examples of fraud, as mentioned in [3]. Money laundering, for example, is to do certain things such as moving funds abroad to conceal the source of proceeds of crime. Here, the key point is to do these acts deliberately. Another example of fraud is, as mentioned in [3], is medical fraud where the offender make false claims at the expense of insurers. As mentioned earlier, fraud is an umbrella term, which has many examples in different contexts. Depending on the nature of the problem, the analysis and modeling choices will change.

In this study, we focus on credit card fraud, where the observations are credit card transactions. However, this is not to say that only one data generation mechanism is at play causing fraudulent transactions. The source of fraud in credit card transactions may vary. As mentioned in [3], the fraudulent credit card transaction may occur due to theft, in which case, the card is present but the beneficiary is different. Another source of fraud in credit card transactions may be because of the lack of due diligence in Know Your Customer principle. The cardholder may have identified herself as someone else and the issuer did not exercised due diligence. Unsupervised modeling in this case is our best effort to detect abnormal behaviours by just looking at the numbers.

2.2 Data for Fraud Detection

Due to data privacy concerns, in most cases, the data used for fraud detection is synthetic data unless the analyst are provided with data by their business partners. There are several synthetic data sets available for fraud detection.

One of these data sets is the Credit Card Fraud Detection Data published on Kaggle [4]. However, the columns of this data set are, except the time -which are shown as steps-, amount and class – which represents the fraud status of the transaction, the result of Principal Component Analysis and this makes it difficult to build an insightful model based on this data set. What is more, the fact that the transactions in this data set seem to span two days stop researchers from being able to produce a model that is generalizable enough.

Another fraud data set is the one that is simulated by PaySim mobile money simulator [5]. The fields of this data set are type of the transaction, amount, name of the originator, old balance of the originator, new balance of the originator, name of the destination, old balance of the destination, new balance of the destination, is fraud and is flagged fraud. The fact that the balances of the parties are included in the data set is a property of this data set that seems to reflect the realistic mechanisms in financial transactions. However, in order to be able to produce a realistic model, the characteristics of the individuals are also important.

A data set that is realistic enough is supposed to include the details of the transactions as well as the characteristics of the individuals that are part of the transactions. Also, this close-to-ideal data set should be large enough to be used in the modeling process so that the insights drawn from it could be generalizable.

One such data set that seems to successfully mimic the real credit card transactions is the one that is simulated by Erik Altman, the details of which are given in this study [6]. This data set consists of the details of the credit card transactions as well as the details about the users and the cards they use.

2.3 Learning Methods in Fraud Detection

Learning methods in fraud detection can be broadly classified as rule-based methods and modeling/statistical learning methods. Unfortunately, rule-based methods require considerable amount of expertise in areas such as finance, public finance, accounting and law [7]. What is more, these methods are time-consuming and complicated to fight against fraud in a timely manner.

In the recent years, modeling approaches in fraud detection have been receiving attention due to their ease of automation and accuracy. Modeling is the effort of formulating the behaviour of the fraudsters mathematically and then using these formulations to detect fraud. In modeling, the decisions have to be made considering the following properties of the data sets:

- whether the data is labeled
- whether the observations are dependent
- the proportion of fraudulent cases
- the connectivity of the financial actors
- how quickly should the model produce predictions

In the first context, the modeling framework decision is either supervised methods or the unsupervised methods. As an example, in [8], 5 supervised, namely Support Vector Machines, K-Nearest Neighbors, Extreme Gradient Boosting, Logistic Regression and Random Forest, and 4 unsupervised learning models, namely One-Class Support Vector Machines, Autoencoders, Restricted Boltzmann Machines and Generative Adversarial Networks were used. In supervised methods, the study found that Extreme Gradient Boosting and Random Forest were the best models, whereas, in the unsupervised setting, Restricted Boltzmann Machine was the best model. The results show that the unsupervised models perform almost as well as the supervised models, which is good news since the data is more likely to be unlabeled and the use of unsupervised models in these cases seems to have been justified. In a similar study [9], Local Outlier Factor, Isolation Forest, Support Vector Machine, Logistic Regression, Decision Tree and Random Forest models were employed with SMOTE to detect fraud in the famous credit card data set [4].

The second item tells us if we should use a time series/multivariate time series, a panel data or a Hidden Markov Model (HMM). Examples of these have been studied in the literature [10] [11]. In [10], similar to this study, credit card transactions were studied. Specifically, through the use of an HMM model, the normal behaviour of the customer is modeled and the new transactions are either accepted or rejected depending on the probability of the transaction. It was shown in the study that the model built has an accuracy level of 0.80. In [11], an ARIMA model was employed to mathematically represent the behaviour of the cardholders. Then the model was used to detect deviations from the historical behaviour, thereby detecting the fraud. It was shown that the ARIMA model outperforms the benchmark models such as Isolation Forests and Local Outlier Factor.

The third item implies that a sampling strategy should be chosen to account for the low proportionality of the fraudulent cases. A number of frameworks such as Oversampling, Undersampling, SMOTE and ADASYN have been proposed and succesfully employed to overcome this problem in credit card fraud detection in the literature [12] [13].

The consideration of connectivity between agents in transactional data has led researchers to employ Graph/Network Analysis in fraud detection as well. This type of analysis has raised a lot of attention in the field since, along with the temporal structure of financial data, connectivity is another interesting feature that reveals hidden relationships between agents. In one study, automobile insurance fraud was modeled with social networks [14]. In another study, a social-network based framework was proposed to detect fraudsters and fraudster groups in a financial data set [15].

Finally, the timeliness of the training and predicting leads us to consider real-time models as in [16].

CHAPTER 3

MATERIAL & METHODS

In this section, the data set and learning methods used in the study are introduced, and the setting of the experiment is explained.

3.1 Data

The data set used for this study is part of three synthetic data sets generated by Erik Altman in [6], the transactions data set. This data set contains the credit card transactions of individuals and has the attributes given in Table 3-1

Although the exact mathematical steps are not given in the paper, the ideas that govern the data generation process are explained in detail. More specifically, in [6], the details of the generation of individuals and their states, merchants, the relationship between individuals and merchants, fraudulent transactions and the accuracy checks that show the data are realistic are given.

In light of these, the data generation mechanism starts with the individuals. Their properties conform to the population mean and standard deviation and also their own personal distribution. This way, the data generated for individuals are not only in line with the population but also with their own distribution. An example given in [6] is the spending of a user on restaurant meals. The first two moments of the individual's spending on restaurant meals are from the population; however, her spending is generated based on her own personal distribution. Finally, it is important to mention that the generation of individuals are based on US-based individuals.

Table 3-1 Attribute Table

Attribute	**Description**
User	user no
Card	card no of the user
Year	year of the transaction
Month	month of the transaction
Date	date of the transaction
Time	time of the transaction
Amount	amount of the transaction
Use Chip	one of chip, swipe or online
Merchant Name	the name of the merchant
Merchant City	the city of the merchant
Merchant State	the state of the merchant
Zip	zip code of the merchant
MCC	merchant category code
Errors	if an error has occurred
Is Fraud	Fraud category

The sequential and grouped behavior of individuals is modeled with state machines in [6] since people act in a certain way in some state and states are sequential. For example, on the weekend, which is the weekend state, we buy certain things and on weekdays, which is the weekdays state, we make different purchases.

The generation of merchants depends on both the merchants and the goods and services they sell. The schema first creates goods and services and these are related to merchant category codes and these are related to merchants and merchant locations.

In the generation of fraudulent transactions, there are two schemas at play. The first is the generation of fraudster population, where every fraudster has certain properties and the fraudulent actions are only valid for a certain period. The simulation of fraudsters is also done in a way that the fraudulent behaviour of the individuals are randomized just like in the real world.

Finally, in order to show the data generated are indeed realistic, the data were compared to the population statistics in an automated way to answer questions like do the biographical statistics, transactional statistics, etc. reflect the reality?

3.2 Feature Engineering

Once the data have been obtained for fraud detection, it is important to generate new features that capture the spending habit of each customer. The idea behind this is that the raw transactions cannot give enough information about an individual and this could hinder the fraud detection process.

With this in mind, by leveraging [17] and [18], 36 new variables were generated in addition to the original variables in the data set. The feature generation mechanism calculates the sums and counts of transactions in the last 24, 60 and 168 hours based on different categories, in line with [17] and [18] as only the user-level aggregation cannot capture the spending habits of the individual. This aggregation was first done on the user level, and then, for cards, merchant category codes, merchant names and cities and states.

In addition to these 36 new variables, another set of 14 variables were generated to capture the circular mean and standard deviation of customer's transaction times. The idea behind this is to obtain the expected times and the variability of customers' transactions similar to [18]. The final attribute table is given in Table 3-2.

Table 3-2 Final Attribute Table

Attribute	Attribute	Attribute
User	circstd_state	count_mcc_60
Card	circmean_city	sum_mcc_168
Year	circstd_city	count_mcc_168
Month	circmean_mcc	sum_merchant_24
Day	circstd_mcc	count_merchant_24
Amount	circmean_chip	sum_merchant_60
Use Chip	circstd_chip	count_merchant_60
Merchant Name	sum_user_24	sum_merchant_168
Merchant City	count_user_24	count_merchant_168
Merchant State	sum_user_60	sum_city_24
Zip	count_user_60	count_city_24
MCC	sum_user_168	sum_city_60
Errors?	count_user_168	count_city_60
Is Fraud?	sum_card_24	sum_city_168
time_radians	count_card_24	count_city_168
circmean_user	sum_card_60	sum_state_24
circstd_user	count_card_60	count_state_24
circmean_card	sum_card_168	sum_state_60
circstd_card	count_card_168	count_state_60
circmean_merchant	sum_mcc_24	sum_state_168
circstd_merchant	count_mcc_24	count_state_168
circmean_state	sum_mcc_60	

To illustrate the calculation of the aggregated variables, we can consider the subset given in Table 3-3. First, the table is sorted by Date and MCC; and, for each row, we go 24 hours back. If there are transactions in the last 24 hours, then we sum them up and use it as the current value of sum_mcc_24. In Table 3-3, for MCC 5541, there is a transaction at 7:30 on the 1st of January in 2018. Since there are no transactions in the last 24 hours, sum_mcc_24 is 0. Then, we look at the second record for MCC 5541, which occurred at 7:32 on the same day. We go back 24 hours and there is a transaction with the amount -78. We use this value for sum_mcc_24. One step further, we have another transaction for MCC 5541 at 7:48. We go back 24 hours and there are two transactions which add up to 0. This is the current value for sum_mcc_24. At 11:20 on the same day, there is another transaction for MCC 5541. We look at the transaction in the last 24 hours. We have -78, 78 and 28.86 so the current sum_mcc_24 is 28.86. Next, we have another transaction at 15:00 for the same MCC. We look again for the transactions in the last 24 hours, where we have -78, 78, 28.86 and 15.49, which add up to 44.35. Finally, for the transaction at 15:06 on the same day, for the same MCC, we have another transaction. We have 44.35 from previous transactions and, together with 12.58, we now have 56.93. Counts are calculated in the same way, and the calculations are repeated for other categories such as users, merchant names, merchant cities, merchant states and cards.

Table 3-3 The Calculation of Sum MCC 24

User	**Datetime**	**MCC**	**Amount**	**sum_mcc_24**
262	1/1/2018 7:30	5541	-78	0
262	1/1/2018 7:32	5541	78	-78
262	1/1/2018 7:48	5541	28.86	0
262	1/1/2018 8:06	4784	30.7	0
262	1/1/2018 8:33	4784	33.57	30.7
262	1/1/2018 9:20	7996	139.03	0
262	1/1/2018 11:20	5541	15.49	28.86
262	1/1/2018 11:38	5411	17.33	0
262	1/1/2018 15:00	5541	12.58	44.35
262	1/1/2018 15:06	5541	68	56.93

After feature engineering, the first six observations of the data set with the final attributes is given in Table 3-4. These observations correspond to the first user and the user carried out transactions with card number 3, 0, 1, 3, 1 and 1, respectively. The transactions were made in January in 2018. The days of the transactions are given as well as the amount and the mode of the transactions. We can also see the merchants, cities, states and MCCs to which the transactions were made. Finally, the fraud status of the transaction as well as the values obtained by aggregation are also presented with the use of the first six observations.

Table 3-4 First Six Observations of the Final Data Set

Variable	**First Six Observations**
User	0, 0, 0, 0, 0, 0
Card	3, 0, 1, 3, 1, 1
Year	2018, 2018, 2018, 2018, 2018, 2018
Month	1, 1, 1, 1, 1, 1
Day	1, 2, 2, 3, 3, 4
Amount	123.97, 130.95, 29.84, 130.45, 39.02, 30.54
Use Chip	Swipe, Chip, Chip, Swipe, Chip, Chip
Merchant Name	2027553650310140000, 5817218446178730000, 7945328079774550000, 4060646732831060000, -5475680618560170000, 4060646732831060000
Merchant City	Mira Loma, La Verne, La Verne, La Verne, Monterey Park, La Verne

Table 3-4 (Continued)

Merchant State	CA, CA, CA, CA, CA, CA
Zip	1, 1, 1, 1, 1, 1
MCC	5541, 5912, 4900, 5411, 5942, 5411
Errors?	0, 0, Bad PIN, 0, 0, 0
Is Fraud?	No, No, No, No, No, No
time_radians	1.47, 1.69, 5.48, 1.60, 1.74, 1.75
circmean_user	2.20, 2.20, 2.20, 2.20, 2.20, 2.20
circstd_user	1.24, 1.24, 1.24, 1.24, 1.24, 1.24
circmean_card	2.05, 2.30, 2.29, 2.05, 2.29, 2.29
circstd_card	1.19, 1.23, 1.26, 1.19, 1.26, 1.26
circmean_merchant	2.67, 1.74, 3.29, 1.85, 1.82, 1.85
circstd_merchant	1.03, .46, .94, .61, .36, .61
circmean_state	2.13, 2.13, 2.13, 2.13, 2.13, 2.13
circstd_state	1.13, 1.13, 1.13, 1.13, 1.13, 1.13
circmean_city	2.86, 2.10, 2.10, 2.10, 1.86, 2.10
circstd_city	1.53, 1.04, 1.04, 1.04, .75, 1.04
circmean_mcc	2.70, 1.73, 3.31, 1.87, 1.82, 1.87
circstd_mcc	1.03, .44, .97, .66, .36, .66
circmean_chip	2.03, 2.27, 2.27, 2.03, 2.27, 2.27
circstd_chip	1.15, 1.15, 1.15, 1.15, 1.15, 1.15
sum_user_24	.00, .00, 130.95, 160.79, 160.29, .00
count_user_24	.00, .00, 1.00, 2.00, 2.00, .00
sum_user_60	.00, 123.97, 254.92, 284.76, 415.21, 330.26
count_user_60	.00, 1.00, 2.00, 3.00, 4.00, 4.00
sum_user_168	.00, 123.97, 254.92, 284.76, 415.21, 454.23
count_user_168	.00, 1.00, 2.00, 3.00, 4.00, 5.00
sum_card_24	.00, .00, .00, .00, 29.84, .00
count_card_24	.00, .00, .00, .00, 1.00, .00

Table 3-4 (Continued)

sum_card_60	.00, .00, .00, 123.97, 29.84, 68.86
count_card_60	.00, .00, .00, 1.00, 1.00, 2.00
sum_card_168	.00, .00, .00, 123.97, 29.84, 68.86
count_card_168	.00, .00, .00, 1.00, 1.00, 2.00
sum_mcc_24	.00, .00, .00, .00, .00, .00
count_mcc_24	.00, .00, .00, .00, .00, .00
sum_mcc_60	.00, .00, .00, .00, .00, 130.45
count_mcc_60	.00, .00, .00, .00, .00, 1.00
sum_mcc_168	.00, .00, .00, .00, .00, 130.45
count_mcc_168	.00, .00, .00, .00, .00, 1.00
sum_merchant_24	.00, .00, .00, .00, .00, .00
count_merchant_24	.00, .00, .00, .00, .00, .00
sum_merchant_60	.00, .00, .00, .00, .00, 130.45
count_merchant_60	.00, .00, .00, .00, .00, 1.00
sum_merchant_168	.00, .00, .00, .00, .00, 130.45
count_merchant_168	.00, .00, .00, .00, .00, 1.00
sum_city_24	.00, .00, 130.95, 160.79, .00, .00
count_city_24	.00, .00, 1.00, 2.00, .00, .00
sum_city_60	.00, .00, 130.95, 160.79, .00, 291.24
count_city_60	.00, .00, 1.00, 2.00, .00, 3.00
sum_city_168	.00, .00, 130.95, 160.79, .00, 291.24
count_city_168	.00, .00, 1.00, 2.00, .00, 3.00
sum_state_24	.00, .00, 130.95, 160.79, 160.29, .00
count_state_24	.00, .00, 1.00, 2.00, 2.00, .00
sum_state_60	.00, 123.97, 254.92, 284.76, 415.21, 330.26
count_state_60	.00, 1.00, 2.00, 3.00, 4.00, 4.00
sum_state_168	.00, 123.97, 254.92, 284.76, 415.21, 454.23
count_state_168	.00, 1.00, 2.00, 3.00, 4.00, 5.00

3.3 Learning Methods

The unsupervised anomaly/outlier detection algorithms for fraud detection employed in this study are Isolation Forests, Autoencoders, Sparse Autoencoders and One-Class SVMs. Also, in order to compare these unsupervised learning methods with a supervised algorithm, a Random Forest model is employed. The reason why a Random Forest model is chosen for this comparison is this model is one of the best performing models in classification problems out-of-the-box [19] [20]. In this section we introduce the unsupervised learning algorithms used in the study.

3.3.1 Isolation Forests

Isolation Forests for anomaly detection was first introduced in 2008 by Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou [21]. The method works in a similar way to Random Forests, that is, it recursively partitions a data set based on a sample of the observations and the features to isolate the observations. The ones that are easily isolated are considered to be anomalous since they are different from the rest of the data.

The anomaly scores are produced by the following equation as seen in Equation (2) in [21],

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \qquad (1)$$

where

- h(x) is the path length, which is the number of edges from the root node to the external node to isolate an observation as stated in [21],
- $E(h(x))$ is the expected value of the path length from the isolation trees,
- $c(n) = 2H(n-1) - (2(n-1)/n)$, which is the expected path length of unsuccessful search in Binary Search Trees. Here, H(i) is estimated by ln(i) + Euler's constant as stated in [21].

If the average path length from the isolation trees is less than the expected path length of unsuccessful search, then the exponent becomes less than 1. As the exponent gets less than 1, the anomaly score gets closer to 1, in which case the observation is considered to be anomalous.

The tuning parameters for Isolation Forests are

- sampling rate,
- number of randomly selected features,
- contamination and
- number of trees.

The effect of these parameters on the performance of the model has been studied in the literature. In the original paper of the Isolation Forest, the experiments were conducted on eleven real and one synthetic data sets. It was found that increasing the sampling rate after a threshold value, which was found to be 256, does not seem to yield much better results [21]. As for the default value for the number of trees, it was found to be 100 in the same study, by which the average path lengths seem to converge [21]. Contamination gives the model the proportion of the outliers to be detected.

3.3.2 Autoenconders

Autoencoders are a type of deep learning algorithm that basically estimates the original data from the data itself, which is called reconstruction. Then the algorithm calculates reconstruction errors to decide if an observation is anomalous or not. If the reconstruction error is high, then the observation is considered to be anomalous.

The estimation of the original data from itself is done with encoding and decoding steps. First, the data is encoded to a lower dimension and is decoded back to the original dimension. This lower dimension is referred to as the bottleneck layer. In the encoding step, the algorithm learns the important structures and summarizes the data, and in the decoding step the data is reconstructed back to the original dimension. This is illustrated in the Figure 3-1.

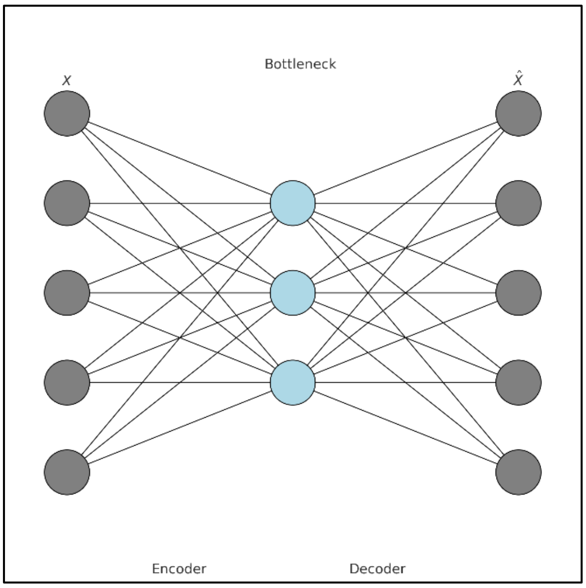

Figure 3-1 Autoencoder Architecture

Mathematically, as seen in Equations (12) and (13) in [8], the reconstruction and the optimization processes can be formulated as

$$\hat{X} = D(E(X)) \tag{2}$$

$$min_{D,E}||X - D(E(X))|| \tag{3}$$

The process of obtaining the values in the encoding and decoding parts is the same as the usual working of neural networks. Specifically, in the encoding part, following the notation from [22], the values are obtained by,

$$E(X) \;=\; f_e(W_e X + B_e) \tag{4}$$

where

- E(X) represents the values in the encoding layer,
- f_e is the activation function for the encoding layer,
- W_e represents the weights for the encoding layer,

- B_e represents the biases for the encoding layer,

Similarly, the values of $\hat{X}$ as a result of the application of the decoding function is given by,

$$\hat{X} = D(E(X)) = f_d(W_d X + B_d) \tag{5}$$

where

- $\hat{X}$ represents the reconstructed input data,
- f_d is the activation function for the decoding layer,
- W_d represents the weights for the decoding layer,
- B_d represents the biases for the decoding layer.

Activation functions for the layers are chosen from a set of possible ones, as in the application of deep networks. When choosing an appropriate activation function, the first consideration is the type of prediction problem. For example, in the case of predicting a binary output, the sigmoid activation is used. Another example is the softmax activation function, which is used in predicting a multi-level categorical outputs. These activation functions are used in the output layer to get the desired output. Another consideration in the selection of activation functions is the ability to capture nonlinearities in the data. The most well-known activation functions used in hidden layers that have this property is the Hyperbolic Tangent Function (tanh) and Rectified Linear Unit (ReLU). In some cases, linear activation function is also used. The activation functions and the corresponding formulae are given in Table 3-5, based on the notations in [23].

Table 3-5 Activation Functions

Activation Function	**Formula**
Linear	$f(x) = x$
ReLU	$\mathrm{f(x) = max(0, x)}$
Hyperbolic Tangent Function (tanh)	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$
Softmax	$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$

There are a number of variants of Autoencoders; however, the first classification could be made based on the sizes of the hidden layers. When the hidden layer has a lower dimension than the input data, $dim(hidden\, layer) \leq dim(input\, data)$, we have an "Undercomplete Autoencoder". Whereas, when the hidden layer has a higher dimension than the input data, $dim(hidden\, layer) \geq dim(input\, data)$, we have an "Overcomplete Autoencoder". A recent debate which should be mentioned at this point on the overcompleteness in Autoencoders is the fact that a bottleneck might not be required in the use of Autoencoders when detecting fraud. This is considered to be a problem in the literature since the overparameterization leads to the learning of the identity function and prevents the model from learning the useful structures in the data [24]. However, in [24], it was found that overparameterized models can outperform the undercomplete ones.

The reconstruction error can be measured with a few different ways, one of which is the mean squared error. Depending on the problem definition other metrics can be used as well.

3.3.3 Sparse Autoenconders

Sparse autoencoders are a variant of autoencoders that allow the sizes of the hidden layers to have a larger dimension, which is called the overcomplete case [25]. It is known that if the sizes of the hidden layers are larger than the input, the model might not learn anything useful [26]. However, in the context of sparse autoencoders, some neurons, due to regularization, approach zero and because of this the model learns the useful structures while not imposing a smaller dimension on the data. This is illustrated with the use of white neurons in Figure 3-2. In the hidden layer, the use of regularization causes some of the neurons approach zero, which controls the sparsity of the hidden layer.

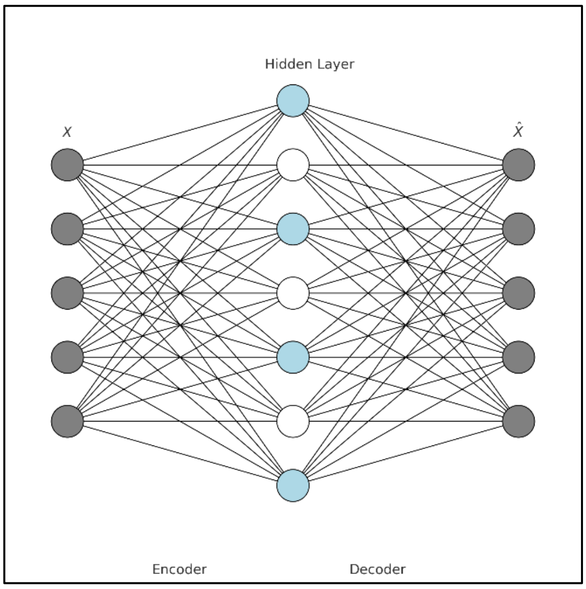

Figure 3-2 Sparse Autoencoder Architecture

In Sparse Autoencoders, loss function becomes, as seen in [25]

$$L_{Sparse} = L + \beta \sum_j KL(\rho || \hat{\rho}_j) \tag{6}$$

where L is the loss from the Autoencoder and KL is the Kullback-Leibler Divergence of the activations of the neurons, which compare the distributions of the activation functions. Here, ρ is the average activation of the neurons and $\hat{\rho}_j$ is the average activation of neuron j. The sparsity is controlled by β [25]. The KL Divergence, as given in [25], is calculated by

$$KL(\rho||\hat{\rho}) \;=\; \rho \, log \, \frac{\rho}{\hat{\rho}_j} \,+\, (1-\rho) \, log \, \frac{1-\rho}{1-\hat{\rho}_j} \tag{7}$$

$$\hat{\rho}_j = \frac{1}{n}\sum_i [a_j(x^{(i)})] \tag{8}$$

where $x^{(i)}$ is the ith sample, a_j is the activation of the hidden unit and n is the size of the training set.

3.3.4 One-Class Support Vector Machines

One-Class Support Vector Machines are first introduced in [27]. The idea behind One-Class SVMs is to create a boundary around the normal instances and use it to identify anomalies/outliers. As given in [8] the objective function in One-Class SVMs is

$$\min \Phi\,(w) = \frac{1}{2}\, w^T w + \frac{1}{\upsilon n}\sum_{i=1}^{n} \xi_i - \rho \tag{9}$$

s.t.

$$\begin{aligned} y_i(\langle w, \phi(x_i)\rangle + b) &\geq \rho - \xi_i, & i &= 1, \ldots, n \\ \xi_i &\geq 0, & i &= 1, \ldots, n \end{aligned} \tag{10}$$

where

- w is the weight vector
- υ is the upper bound on the proportion of training errors and a lower bound of the fraction of support vectors
- ξ_i is the slack variable for observation i
- ρ is the minimum distance from the origin to any data point in the transformed space [28].

In the objective function, the first part, $\frac{1}{2}\, w^T w$ represents the minimization of the norm of the weight vector, which also means the maximization of the margin. The second part, $\frac{1}{\nu n}\sum_{i=1}^{n}\xi_i$, represents the penalization of the incorrectly classified observations. Finally, in the third part, the minimization of $-\rho$ means the maximization of the distance from the origin to the decision boundary.

3.4 Experiment

The credit card data set used in the study has more than 20 million observations. However, in our experiment, the data set was subsetted to obtain the observations for 2018 and 2019 for the first 1,000 users. The data set was then further sampled to obtain normal instances from the data set. 100,000 normal instances were obtained from the data set randomly to make up the training set and for the test set another 10,000 normal observations were sampled and brought together with the 2,186 fraudulent transactions to form the test set. All the numeric features were standardized using Min-Max Scaler[1]. The categorical variables were binary encoded due to their high cardinality. This data setting was used in Autoencoder, Sparse Autoencoder and One-Class SVM model since the training is carried out on normal instances in these models.

For the Autoencoder and Sparse Autoencoder models, train-validation-test approach was used to tune the hyperparameters and measure the performance of the models on the test set. For both models, the architecture contains two encoder layers and one decoder layer. Moreover, the models were built with an overcomplete design where the dimensions of the hidden layers are greater than the input layer with a similar approach in [24].

---

[1] $XTR = \frac{X_i - min\,(X)}{max(X) - min(X)}$

For the Autoencoder and Sparse Autoencoder models, the hyperparameter spaces that were evaluated are given in Table 3-6. The first thing to point out in the selection of encoder and decoder units. The input layer has a size of 64. However, since an overcomplete design is followed based on [24], the sizes of encoder and decoder layers were set to 128 and 256 in the hyperparameter search. Also, batch sizes are set to 128 and 256.

Table 3-6 AE and SAE Hyperparameter Spaces

AE Hyperparameter	**Search Space**
encoder units	128 and 256
encoder activation	relu, tanh
decoder layer	same as the encoder
learning rate	0.0001- 0.001 with log sampling
optimizer	adam, sgd, rmsprop
weight initializer	glorot normal, glorot uniform, he normal and he uniform
batch size	128 and 256
SAE Hyperparameter	**Search Space**
encoder units	128 and 256
activation	relu, tanh
decoder	same as encoder
learning rate	0.0001 to 0.001 with log sampling
regularization	0.00001 to 0.0001 with log sampling
optimizer	adam, sgd, rmsprop
weight initializer	glorot normal, glorot uniform, he normal and he uniform
batch size	128 and 256

For the One-Class Support Vector Machine model, since the model is not meant to reconstruct the original input, instead of tuning the hyperparameters, the effect of different kernel, gamma, nu and degree hyperparameters was shown on the performance of the model on the test set. The hyperparameter setting is given in Table 3-7.

Table 3-7 One-Class Support Vector Machine Hyperparameter Space

Hyperparameter	**Values**
kernel	linear, poly
Gamma	0.001, 0.01, 0.1, 1
Nu	0.001, 0.01, 0.1
degree	3, 4

For the Isolation Forest Model, since the model is not expected to be trained only on normal instances the data set with users 1-1,000 with transactions in 2018 and 2019 was used. The train-test split was done randomly, where 80% of the data were set as the training set and 20% of the data were set as the test set. Same data setting as the Isolation Forest was used in the Random Forest model when building the supervised model for fraud detection. The hyperparameter space of the Isolation Forest model is given in Table 3-8.

Table 3-8 Isolation Forest Hyperparameter Space

Hyperparameter	**Values**
contamination	0.001, 0.01, 0.1
max features	0.5, 1
max samples	128, 256, 512
n estimators	50, 100, 200

For the measurement of the performance of the models, Recall, Precision and F1 scores were used as these metrics reflect the balance of false positives and false negatives in fraud detection cases.

$$Precision = \frac{TP}{TP+FP} \tag{11}$$

$$Recall = \frac{TP}{TP+FN} \tag{12}$$

$$F1 = 2 * \frac{(Precision*Recall)}{(Precision+Recall)} \tag{13}$$

where

- TP is True Positives, that is, correctly classified as fraudulent transactions
- FP is False Positives, that is, incorrectly classified as fraudulent transactions
- FN is False Negatives, that is, incorrectly classified as non-fraudulent transactions.

All the experiments were conducted in the Python programming language, version 3.11.5, distributed with Anaconda, version 23.7.4. on a Windows-10 machine. The important packages deployed include Pandas(v2.0.3), Matplotlib(v3.7.2), scikit-learn(v1.3.0) and Tensorflow(v2.14.0).

CHAPTER 4

RESULTS

4.1 Summary of the Data Set

The subsetted data set for user 1-1,000 with the transactions in 2018 and 2019, as explained in 3.4. has 64 columns and 1,708,950 observations.

The summary statistics for the numeric columns are given in Table 4-1. According to the statistics, the median of "Amount" for fraudulent transactions are slightly higher than the nonfraudulent ones. Similarly, when we compare the days and months for the fraudulent and nonfraudulent transactions, there seem to be slight differences. As for the variables created for the expected time and standard deviations in the times of the transactions calculated by User, Card, MCC, Merchant, State, City and Use Chip, the standard deviations seem to have more interesting patterns. For City, State and Merchant, the standard deviations in the times are less for the fraudulent transactions.

The assessment of the aggregated variables can be done focusing on three aspects of the statistics. First, we can focus on the categories to see the which ones seem to be interesting in explaining fraud. Second, we can focus on the aggregation interval to see the interesting patterns in intervals. Finally, we can focus on the type of statistics.

The sums and counts based on MCC and Merchant Name are all zero for all intervals for fraudulent transactions. Also, while count statistics for user are close for fraudulent and nonfraudulent transactions, there is information in the sums for user-based statistics; that is the amounts are greater for fraudulent transactions. Card-related statistics follow the same pattern as the user-based statistics. Finally, city-based and state-based statistics follow a similar pattern. Sums and counts are lower for the fraudulent statistics in both categories across all aggregation intervals.

Finally, time in radians for fraudulent and nonfraudulent transactions have also close values.

Table 4-1 Summary Statistics for the Numeric Columns

	Fraud 1 (n=2,186)	**Fraud 0 (n=1,706,764)**
Variable	Median (SD)	Median (SD)
Amount	35.97 (147.44)	29.72 (81.70)
Day	14.50 (8.83)	16.00 (8.79)
Month	6.00 (3.25)	7.00 (3.45)
circmean_card	3.28 (0.85)	3.30 (0.79)
circmean_chip	3.24 (0.87)	3.29 (0.86)
circmean_city	3.92 (0.33)	3.31 (0.97)
circmean_mcc	3.57 (0.98)	3.33 (1.19)
circmean_merchant	3.80 (0.78)	3.33 (1.20)
circmean_state	3.92 (0.32)	3.29 (0.88)
circmean_user	3.29 (0.87)	3.29 (0.78)
circstd_card	1.19 (0.33)	1.19 (0.35)
circstd_chip	1.18 (0.35)	1.13 (0.40)
circstd_city	0.69 (0.21)	1.05 (0.47)
circstd_mcc	0.72 (0.51)	0.61 (0.52)
circstd_merchant	0.12 (0.45)	0.55 (0.53)
circstd_state	0.69 (0.19)	1.13 (0.40)
circstd_user	1.19 (0.32)	1.19 (0.34)
count_card_168	10.00 (7.58)	11.00 (11.35)
count_card_24	2.00 (1.97)	2.00 (2.13)
count_card_60	4.00 (3.74)	4.00 (4.51)
count_city_168	2.00 (2.38)	10.00 (12.57)
count_city_24	0.00 (1.23)	1.00 (2.28)
count_city_60	1.00 (1.81)	4.00 (4.96)
count_mcc_168	0.00 (1.88)	2.00 (6.33)
count_mcc_24	0.00 (0.58)	0.00 (1.23)
count_mcc_60	0.00 (1.01)	1.00 (2.53)
count_merchant_168	0.00 (0.74)	1.00 (5.68)

Table 4-1 (Continued)

count_merchant_24	0.00 (0.34)	0.00 (1.14)
count_merchant_60	0.00 (0.46)	0.00 (2.31)
count_state_168	2.00 (2.38)	18.00 (13.96)
count_state_24	0.00 (1.23)	3.00 (2.55)
count_state_60	1.00 (1.81)	6.00 (5.51)
count_user_168	23.00 (12.54)	25.00 (15.38)
count_user_24	4.00 (2.66)	3.00 (2.82)
count_user_60	9.00 (5.56)	9.00 (6.09)
sum_card_168	492.18 (519.94)	410.81 (618.59)
sum_card_24	59.77 (189.67)	41.22 (140.17)
sum_card_60	174.00 (311.32)	132.97 (266.09)
sum_city_168	130.66 (296.59)	337.05 (506.62)
sum_city_24	0.00 (148.22)	31.86 (111.32)
sum_city_60	23.09 (219.09)	111.63 (215.67)
sum_mcc_168	0.00 (119.24)	41.54 (251.99)
sum_mcc_24	0.00 (51.16)	0.00 (54.88)
sum_mcc_60	0.00 (85.23)	2.26 (103.10)
sum_merchant_168	0.00 (72.34)	13.64 (225.12)
sum_merchant_24	0.00 (34.44)	0.00 (50.76)
sum_merchant_60	0.00 (49.06)	0.00 (93.02)
sum_state_168	130.66 (296.59)	627.20 (665.76)
sum_state_24	0.00 (148.22)	76.42 (146.84)
sum_state_60	23.09 (219.09)	217.02 (282.70)
sum_user_168	1,046.68 (738.83)	973.55 (813.29)
sum_user_24	140.66 (229.08)	116.78 (187.41)
sum_user_60	370.60 (393.53)	324.21 (354.35)
time_radians	3.89 (0.74)	3.34 (1.32)

The summary statistics for the categorical variables are given in Table 4-2. The categories are sorted based on the frequency of fraudulent transactions in a descending manner. Due to high cardinality only the first three levels are given in the table. The MCC code which has most of the fraudulent transactions is 5311 and it is followed by 5411 and 5310. Most of the fraudulent transactions occur in Rome, Italy, as the results show. Although the numbers are close, 2018 has the larger number of fraudulent transactions. When a transaction is fraudulent, the zip code of the merchant is missing 15% of the time.

Table 4-2 Summary Statistics for the Categorical Columns

		Fraud 1 (n=2,186)	**Fraud 0 (1,706,764)**
Variable		n (%)	n (%)
MCC*			
	5311	516 (24%)	58,113 (3.4%)
	5411	189 (9%)	198,918 (12%)
	5310	176 (8%)	30,376 (1.8%)
Merchant City*			
	Rome	2,137 (97%)	1,155 (0.07%)
	Online	41 (2%)	220,215 (13%)
	Marion	8 (0.4%)	231 (0.01%)
Merchant State*			
	Italy	2,137 (98%)	296 (0.02%)
	Online	41 (2%)	220,215 (13%)
	Ohio	8 (0.4%)	67,925 (4%)
Year			
	2018	1,228 (56%)	851,711 (50%)
	2019	958 (44%)	855,053 (50%)
Zip			
	0	1,861 (85%)	1,501,707 (88%)
	1	325 (15%)	205,057 (12%)

*Only the first 3 levels are given due to high cardinality.

4.2 Visual Analysis

In this section, a series of visualization steps are followed in order to gain more insight into what we are looking for in the detection of fraud in the context of the data set used in the study. We especially focus on the relationships between the total amounts and counts based on different intervals to examine the contributions of historical behaviours of the customers in the separation of the classes. This is considered a necessary step to enforce the findings achieved in the summarization of the data set in Section 4.1. We also examine the correlation structure of the data set in the section.

The first set of variables to consider is the total amount in the last 24 hours vs 60 hours. This is because pairwise comparisons give similar characteristics in the comparison of total amounts based on duration. From the examination of Figures 4.1 – 4.4, the main characteristic of fraudulent transactions based on Total Amounts is that a transaction is likely to be fraudulent when the total in the last 24 hours is fixed close to 0 in the untransformed data. Also, if the total in the last 24 and 60 hours are increasing proportionally on the lower limit, the transaction is likely to be fraudulent. Finally, when both have small and positive values, the transaction is more likely to be fraudulent.

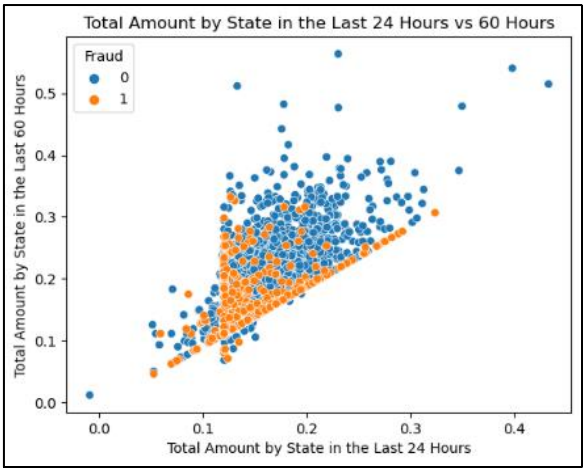

Figure 4-1 Total Amount by State in the Last 24 Hours vs 60 Hours

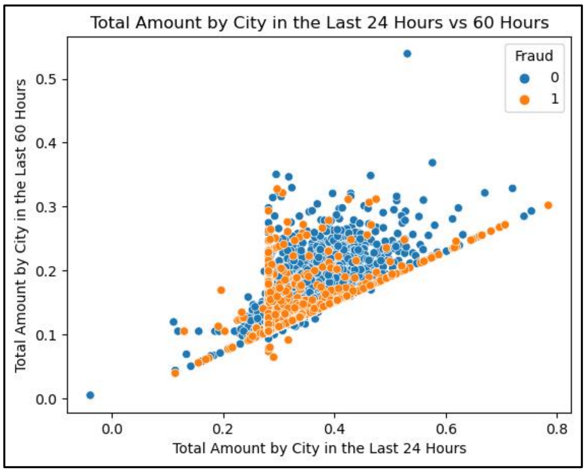

Figure 4-2 Total Amount by City in the Last 24 Hours vs 60 Hours

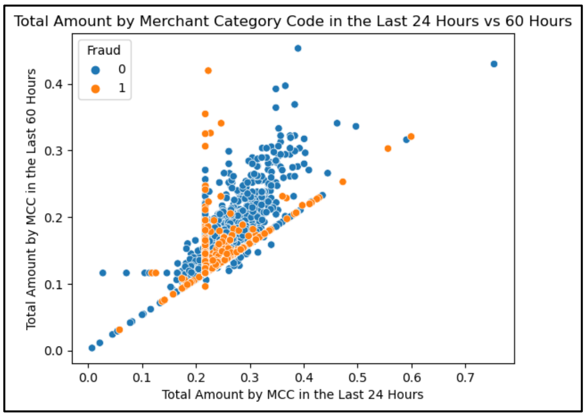

Figure 4-3 Total Amount by MCC in the Last 24 Hours vs 60 Hours

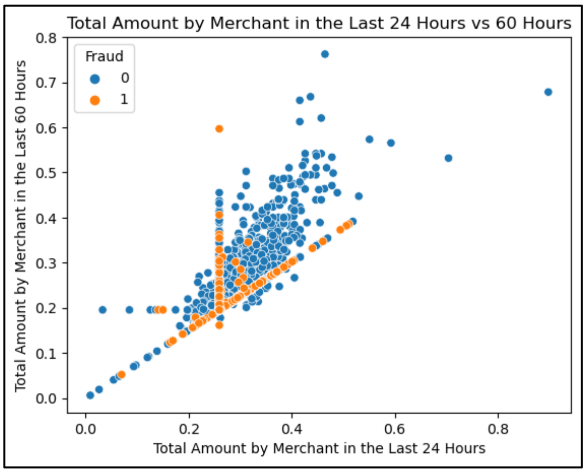

Figure 4-4 Total Amount by Merchant in the Last 24 Hours vs 60 Hours

The second set of variables to consider is the count groups in the last 24 Hours vs 60 Hours. For a fixed total count in the last 24 hours, a transaction is more likely to be fraudulent when the total count in the last 60 hours is small.

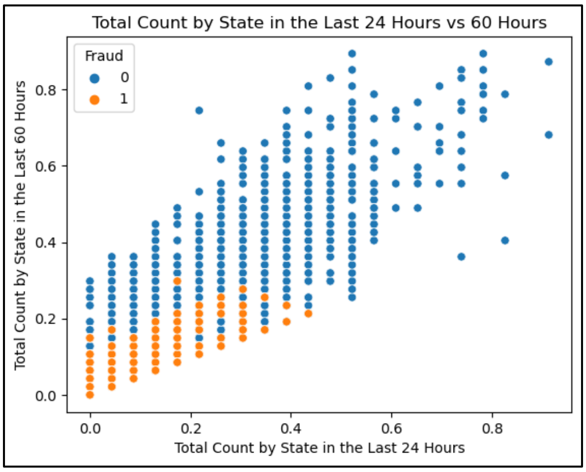

Figure 4-5 Total Count by State in the Last 24 Hours vs 60 Hours

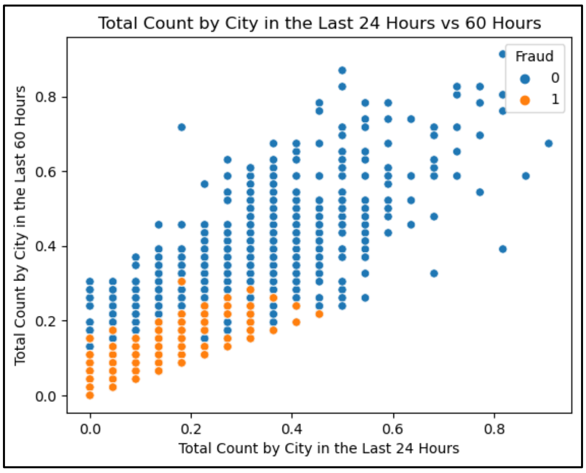

Figure 4-6 Total Count by City in the Last 24 Hours vs 60 Hours

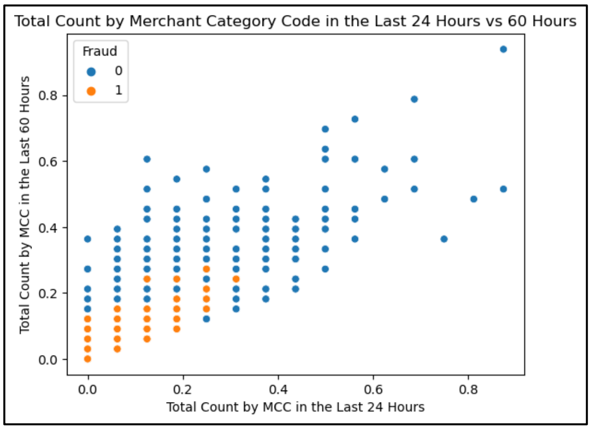

Figure 4-7 Total Count by MCC in the Last 24 Hours vs 60 Hours

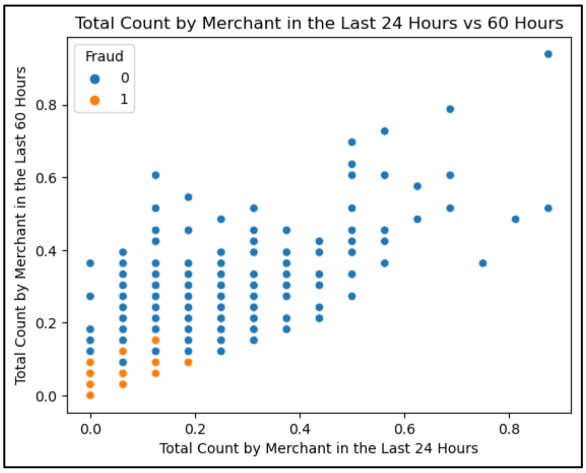

Figure 4-8 Total Count by Merchant in the Last 24 Hours vs 60 Hours

When we compare current amounts with the aggregated data, things get a little complicated. Two example illustrations are given in Figure 4-9 and Figure 4-10 to clarify this statement. When the total amount in the last 24 hours is fixed around 0.21, which is 0 in the untransformed data, a transaction is more likely to be fraudulent; however, this is not the only pattern in the data. As a matter of fact, the rest of the relationship does not exhibit a clear pattern; hence, the amount, in the existence of the aggregated variables does not contribute much to the separation of the classes.

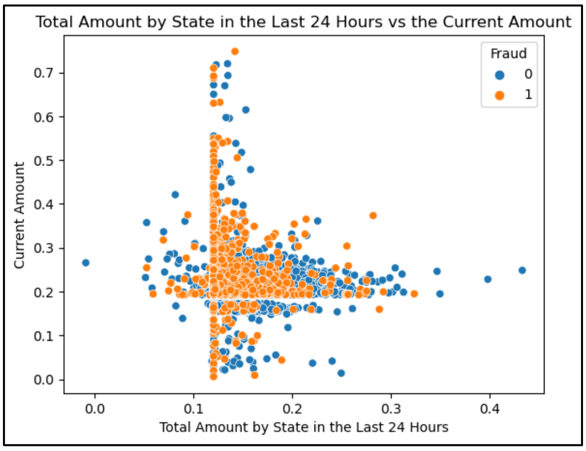

Figure 4-9 Total Amount by State in the Last 24 Hours vs the Current Amount

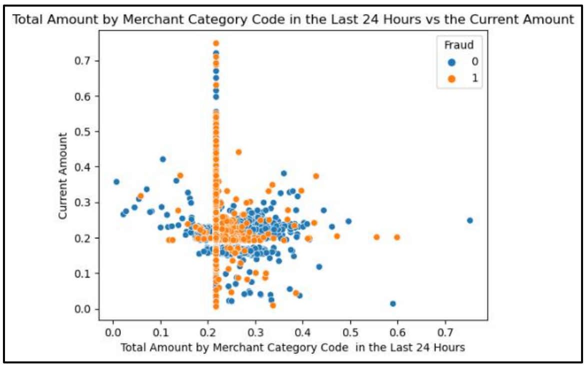

Figure 4-10 Total Amount by MCC in the Last 24 Hours vs the Current Amount

Also, using the t-SNE algorithm the data set was transformed into a 2-dimensional data to obtain another basis visualization in the detection of fraud. This is given in Figure 4-11.

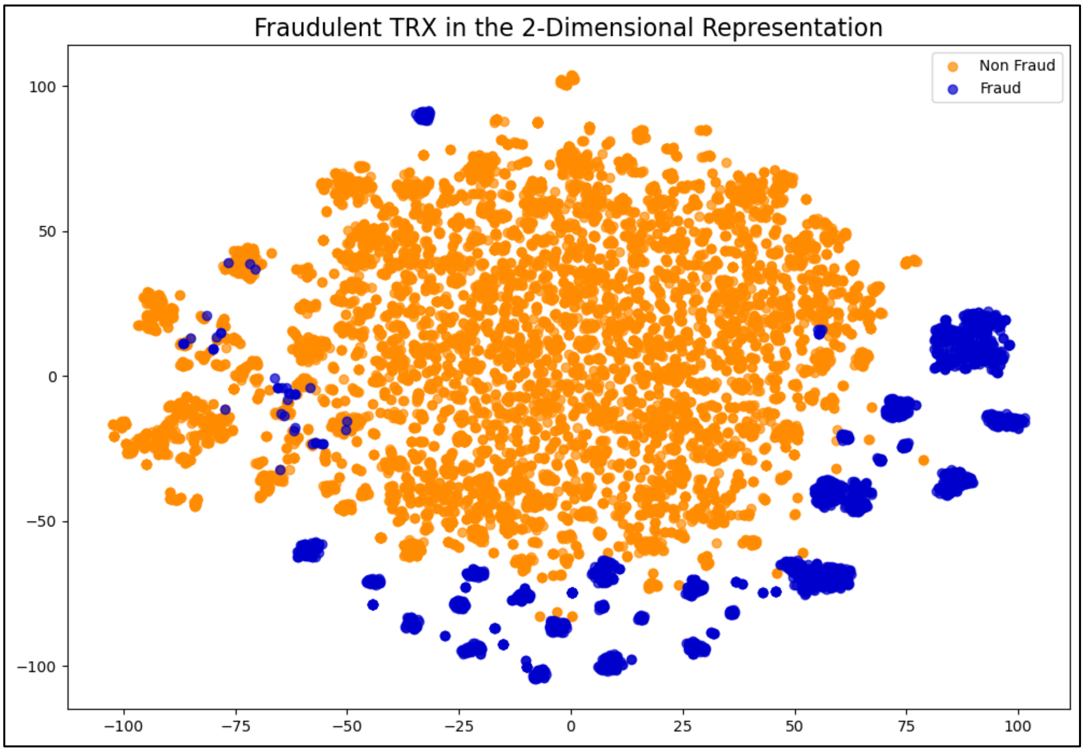

Figure 4-11 2-Dimensional Representation of the Fraudulent TRX

Due to the aggregated variables, it is expected have correlated variables in the data set. The heat map given in Figure 4-12 illustrates the correlation structure of the data. Based on the heat map, there seem to be three main correlation structures. The first one results from the relationships between the expected times of the transactions, the second one results from the circular standard deviations of the times of the transactions, and the third one results from the aggregated variables.

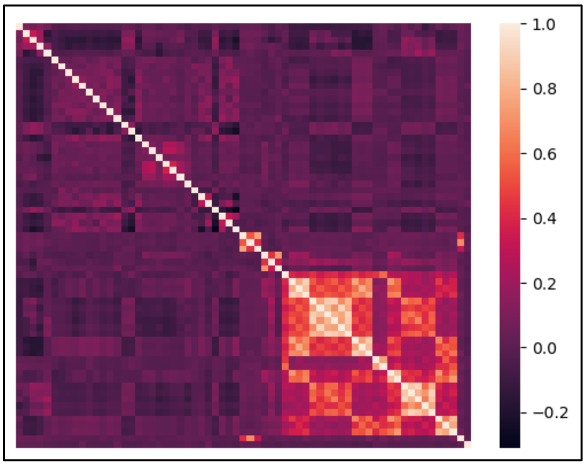

Figure 4-12 Correlation Map

4.3 Model Results

Based on the tuning process, the hyperparameter values of the best Autoencoder model are glorot normal weight initializer, 256 units and relu activation in the encoding and decoding layers and linear activation in the output layer and 0.0001 learning rate with adam optimizer and 256 batch size.

The hyperparameters of the best Sparse Autoencoder model are he normal weight initializer, 256 units and relu activation in the encoding and decoding layers, linear activation in the output 0.00003 L1 regularization for the first encoding layer, 0.00001 L1 regularization for the second encoder layer and 0.00003 in the decoding layer, 0.0001 learning rate with 128 batch size and adam optimizer.

One-Class SVM gives the best F1 score with gamma 1, nu 0.1 with polynomial kernel with 4 degrees.

The model results on the test set show that out of all the unsupervised models employed, the best performance belongs to One-Class SVM with an F1-score of 0.81. However, the Random Forest Model outperformed the One-Class SVM model, being the best model among the employed models in the study. Finally, the Isolation Forest gives the worst performance among the models employed.

Table 4-3 Model Results on the Test Set

Model	**Precision**	**Recall**	**F1-Score**
Autoencoder	0.51	0.57	0.54
Sparse Autoencoder	0.65	0.73	0.69
One-Class SVM	0.68	0.98	0.81
Isolation Forest	0.00	0.03	0.00
Random Forest	1.00	0.95	0.97

4.4 Comparison of Models

As the results show, the models show different performances in the detection of fraudulent transactions. When comparing the models' performances, we should first consider the patterns in the fraudulent transactions and the patterns models capture when detecting these and how they compare.

First, we examine the 2-dimensional representation of the fraudulent transactions with the test and the predicted data sets. Figure 4-13 shows that the Autoencoder model was able capture some of the patterns in the fraudulent transactions. However, the Autoencoder model produces fraudulent predictions in the area almost completely comprised of normal transactions.

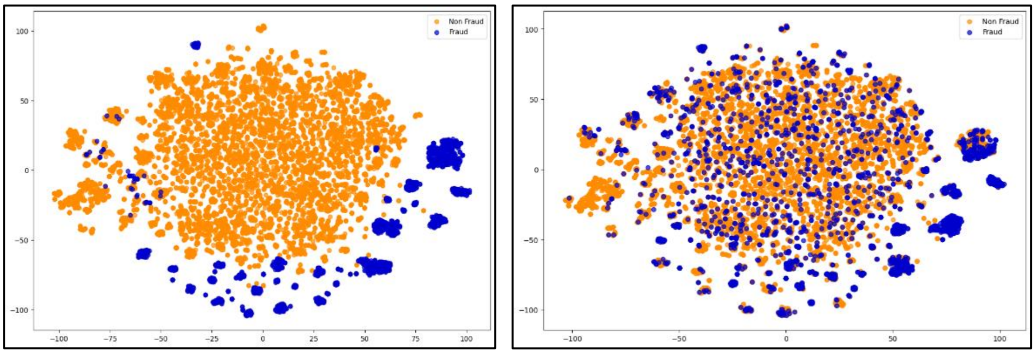

Figure 4-13 2-Dimensional Representations of the Test Set and Autoencoder Predictions

This is because Autoencoders try to reconstruct the input data without any focus on the separation of the classes, obviously. When we examine the reconstruction errors on the test set, we see that the input data was reconstructed almost perfectly, which can be seen from Figure 4-14.

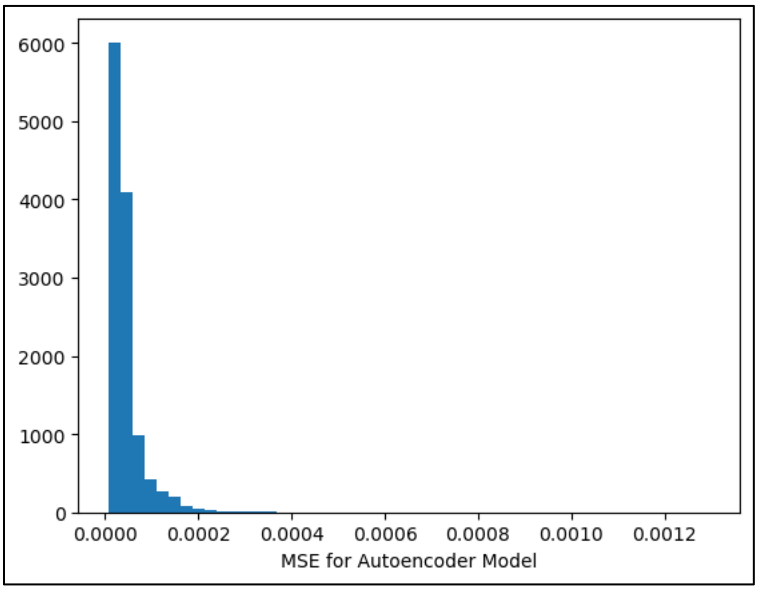

Figure 4-14 MSE for Autoncoder Model

We know that the Autoencoder model labels an observation as fraudulent when its reconstruction error is high. We also know that the reconstruction error is low overall for the data set. Therefore, the model has successfully learnt the input data. However, we need to dig a little deeper to understand how the model labels fraudulent transactions and which variables are important in the process. In order to answer this question, we follow a feature elimination strategy to see how much the performance metrics decreases in the absence of these variables. The rest of the variables do not change the model performance significantly.

Table 4-4 AE Model Performance with Removed Variables

Removed Variable	**Precision**	**Recall**	**F1-Score**
-Merchant State	0.17	0.19	0.18
-Merchant City	0.18	0.21	0.19

In the same fashion, Figure 4-15 shows that the Sparse Autoencoder model was able capture some of the patterns in the fraudulent transactions. However, this model, too, produces fraudulent predictions in the area almost completely comprised of normal transactions.

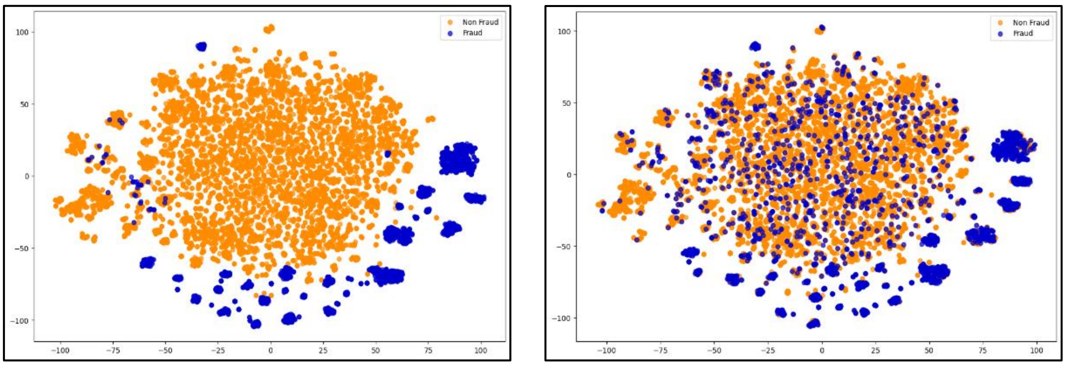

Figure 4-15 2-Dimensional Representations of the Test Set and Sparse Autoencoder Predictions

The reconstruction errors for the Sparse Autoencoder model also show that the model has successfully learnt the input data. The errors are as can be seen from Figure 4-16 are mostly zero.

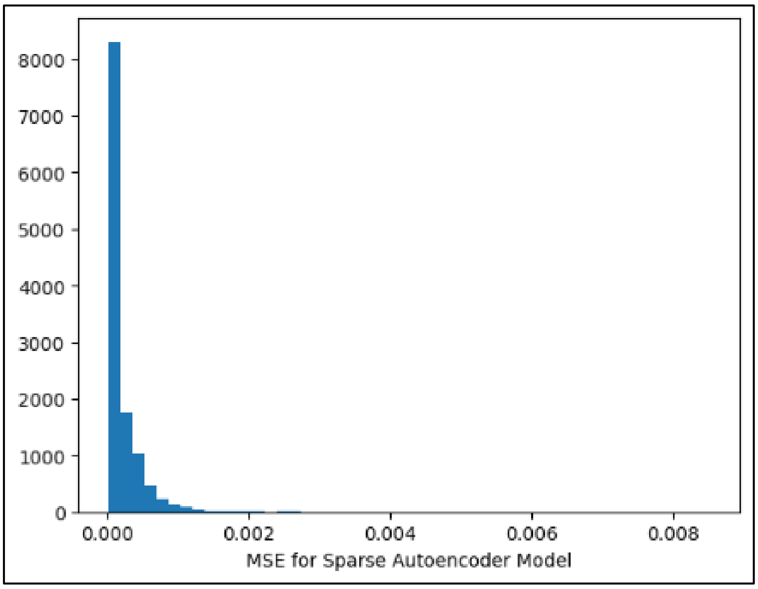

Figure 4-16 MSE for Sparse Autoencoder ModelTo summarize the Autoencoder-based models, it can be said that the "overcomplete design", where the hidden layers have higher dimensions are successful in the detection of fraudulent transactions. The Sparse Autoencoder model performs better than the regular Autoencoder model, which means the regularization of the hidden layers improve the model performance. For both models, removal of location-related variables worsen the model performances.

As for the OCSVM model, the 2-dimensional representations show that the patterns in the test set are mostly captured with a number of false positives in the normal area.

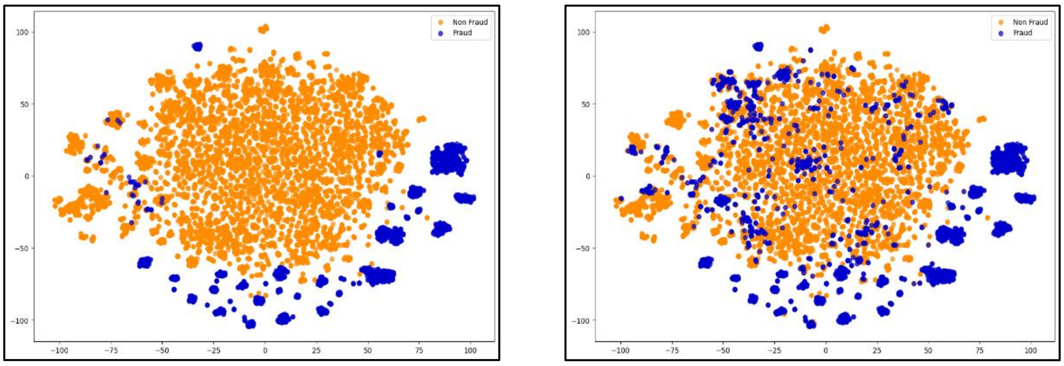

Figure 4-17 2-Dimensional Representation of the Test Set and OCSVM Predictions

The OCSVM model, unlike the Autoencoder-based models does not reconstruct the data. The anomaly scores of OCSVM are obtained with the distances to the separating hyperplane. The transactions with anomaly scores that are less than 0 are considered to be fraudulent based on the logic of the OCSVM model. The anomaly scores are given in Figure 4-18.

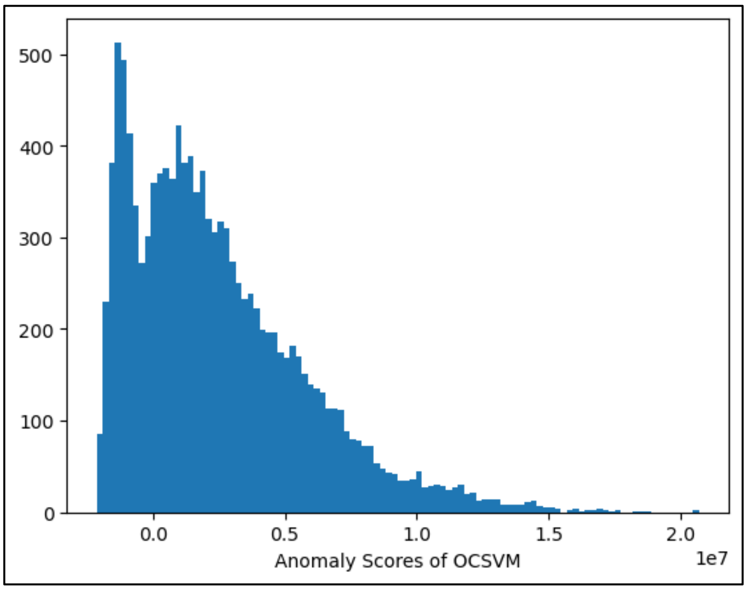

Figure 4-18 Anomaly Scores of OCSVM

The examination of what contributed to the labeling of fraudulent transactions is carried out with a feature elimination strategy similar to the strategy followed for Autoencoder-based models. Several models were built in order to see the elimination of which variables worsens the model performance. The variables whose removal worsened the model are given in Table 4-5. The results show that the main determinants of the fraudulent transactions are the locations of the merchants. The removal of the rest of the variables do not seem to affect the model performance significantly.

Table 4-5 OCSVM Performance with Removed Variables

Removed Variable	**Precision**	**Recall**	**F1-Score**
-Merchant State	0.59	0.68	0.63
-Merchant City	0.48	0.43	0.45

The Random Forest model shows the best performance in detecting fraud (F1 Score=0.97). In order to illustrate what drives Random Forest model in the detection of fraudulent transactions, the variable importance plot in Figure 4-19 is used. The plot shows that the five most important variables in the detection of fraud are circstd_time_state, Merchant State_6, circstd_time_city, Merchant State_3, Merchant State_4 and Merchant State_2.

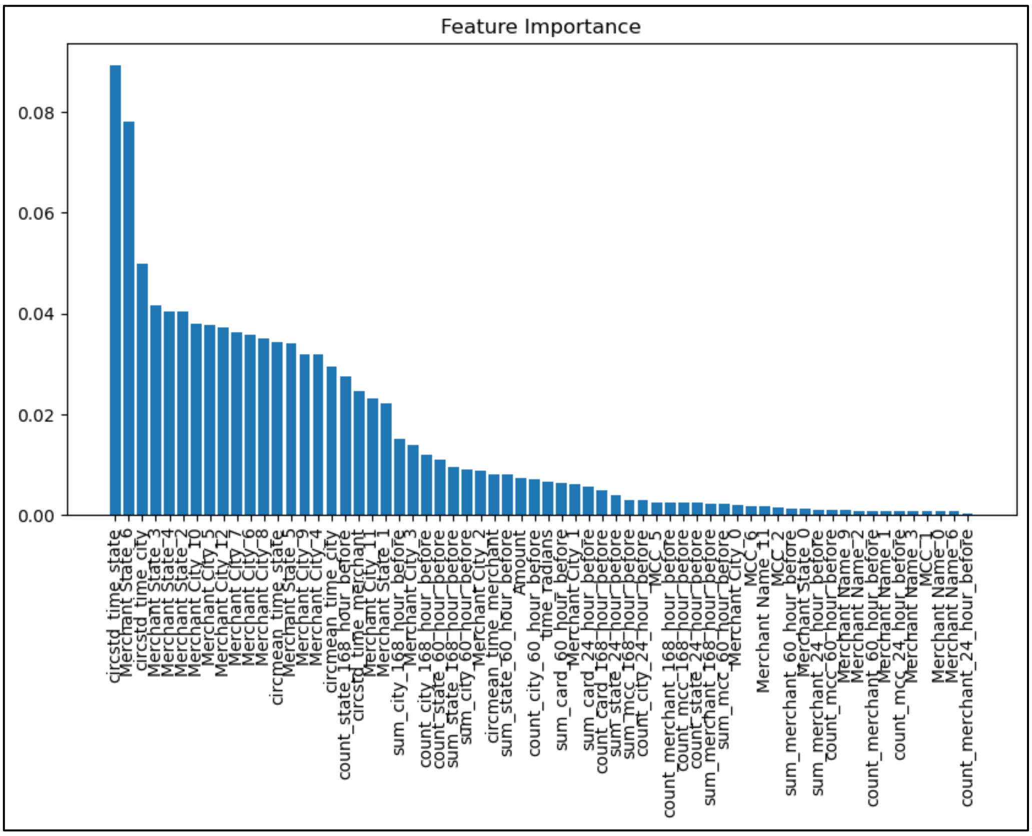

Figure 4-19 Feature Importances

The examination of the model performances has revealed that the most important variables in the detection of fraud for the data set used in this study are the location-related ones. The removal of these variables seem to worsen the performance of the models. The data set, based on the analyses conducted, do not react much to the behavioral patterns captured by the generated features via the aggregation strategies followed in the study.

CHAPTER 5

DISCUSSION

One of the most important problems in fraud detection is that the data come in an unlabeled form. In this case, the labeling of the data is done by rules/expert opinion. However, this approach comes with a few but serious problems. First, it is nearly impossible to measure the false negatives. The rules that do not cover unknown patterns will cause not being able to know the cases that are fraudulent. On top of this, the analytical models will be built on these labels which will lead to the propagation of this problem. Another problem is the temporal nature of the patterns. Depending on new financial technologies, the patterns in fraud detection may change. These labeling mechanism and analytical model building processes should also take this fact into account. Therefore, the model building processes should first focus on high recall rates with a compromise on the precision rates. Second, the models should bear the novelty detection characteristic as well. This is necessary because of the models' need to adapt to new fraudulent patterns.

With these in mind, several unsupervised learning methods were applied to detect credit card fraud in the study. First of all, the consolidation of the results in a unified approach has some complexities. This results from the fact that each method has its own way of working and it is difficult to translate these results into unified insights with respect to fraud detection. Autoencoders work with the reconstruction of the input data; One-Class SVMs work in a way that the fraudulent transactions are best separated from the nonfraudulent ones. Also, in these models, normal transactions are used in the training. However, Isolation Forests use normal and abnormal observations and attempt to isolate the abnormal observations and the anomaly score generation process is also different from the other two approaches. As for the performances of the models, One-Class SVMs exhibit the best performance among the unsupervised models. However, Random Forest Model outperformed this model, showing that the unsupervised models still have room for improvement and in a commercial setting, these should be the options to go with.

Another important point in fraud detection is the availability of benchmark data that will enable researchers to exhibit their insights with a special focus on typologies in credit card fraud. In most of the studies, either open data sets that have no meaningful attributes are used or the studies cannot reveal their insights, apart from the numerical ones, due to data privacy issues. In this study, a synthetic data set was used to be able to reach insights from meaningful variables. However, it is concluded after working with this data set, it has other problems. First of all, the data set does not follow meaningful patterns in the fraudulent transactions, such as commonly encountered typologies. More specifically, the typologies in credit card fraud such as card not present, card replication, or suspiciously recurring transactions in small amounts do not exist in the data set. For this reason, most of the fraudulent transactions can be explained with the location-related variables, which do not allow us to reach meaningful insights.

As future work, this study will build on three fundamental aspects in fraud detection in financial transactions. Temporal nature of the data will need to be incorporated in the learning methods used, Autoencoders, One-Class SVMs and Isolation Forests. Also, the models will take into account the data generation mechanisms, which are the different types of fraud mechanisms in the same category, e.g. credit card fraud. Finally, social network analysis will also be a part of the modeling process to better detect fraud.

REFERENCES

[1] Association of Certified Fraud Examiners;, "acfe.com," 2023. [Online]. Available: https://www.acfe.com/fraud-resources/fraud-101-what-is-fraud. [Accessed 25 7 2023].

[2] National Fraud & Cyber Crime Reporting Centre;, "National Fraud & Cyber Crime Reporting Centre," 2023. [Online]. Available: https://www.actionfraud.police.uk/a-z-of-fraud. [Accessed 25 7 2023].

[3] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," *Statistical Science,* vol. 17, no. 3, pp. 235-255, 2002.

[4] Machine Learning Group, ULB;, "Kaggle," 2023. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. [Accessed 7 25 2023].

[5] E. A. Lopez-Rojas, A. Elmir and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," *The 28th European Modeling and Simulation Symposium-EMSS, Larnaca, Cyprus.,* 2016.

[6] E. Altman, "Synthesizing Credit Card Transactions," *Proceedings of the Second ACM International Conference on AI in Finance [Preprint],* 2021.

[7] A. Abdallah, M. A. Maarof and A. Zainal, "Fraud detection system: A survey," *Journal of Network and Computer Applications,* vol. 68, pp. 90-113, 2016.

[8] X. Niu, L. Wang and X. Yang, "A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised," *arXiv:1904.10604,* 2019.

[9] V. N. Dornadula and G. S, "Credit Card Fraud Detection using Machine Learning Algorithms," *International Conference on Recent Trends in Advanced Computing,* vol. 165, pp. 631-641, 2019.

[10] A. Srivastava, A. Kundu, S. Sural and A. K. Majumdar, "Credit Card Fraud Detection Using Hidden Markov Model," *IEEE Transactions on Dependable and Secure Computing,* vol. 5, no. 1, pp. 37-48, 2008.

[11] G. Moschini, R. Houssou, J. Bovay and S. Robert-Nicoud, "Anomaly and Fraud Detection in Credit Card Transactions Using the ARIMA Model," *Eng. Proc.,* vol. 5, no. 1, 2021.

[12] T. C. Tran and T. K. Dang, "Machine Learning for Prediction of Imbalanced Data: Credit Fraud Detection," *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Korea (South),* pp. 1-7, 2021.

[13] G. Haixianga, L. Yijinga, J. Shang, G. Mingyuna and H. Yuanyuea, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems With Applications,* vol. 73, pp. 220-239, 2017.

[14] L. Šubelj, Š. Furlan and M. Bajec, "An expert system for detecting automobile insurance fraud using social network analysis," *Expert Systems with Applications,* vol. 38, pp. 1039-1052, 2011.

[15] R. A. Leite, T. Gschwandtner, S. Miksch, E. Gstrein and J. Kuntne, "Network Analysis for Financial Fraud Detection," EUROVIS 2018, Austria, 2018.

[16] A. Thennakoon, C. Bhagyani, S. Premadasa, S. Mihiranga and N. Kuruwitaarachchi, "Real-time Credit Card Fraud Detection Using," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2019.

[17] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston and N. M. Adams, "Transaction aggregation as a strategy for credit card," *Data Mining and Knowledge Discovery,* vol. 18, pp. 30-55, 2008.

[18] A. C. Bahnsen, D. Aouada, A. Stojanovic and B. Ottersten, "Feature Engineering Strategies for Credit Card Fraud Detection," *Expert Systems with Applications,* vol. 51, pp. 134-142, 2016.

[19] L. Grinsztajn, . E. Oyallon and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2022, pp. 507-520.

[20] S.-Z. Ravid and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion,* vol. 81, pp. 84-90, 2022.

[21] F. T. Liu, K. M. Ting and Z.-H. Zhou, "Isolation Forest," *2008 Eighth IEEE International Conference on Data Mining,* pp. 413-422, 2008.

[22] S. Misra, S. Thakur, M. Ghosh and S. K. Saha, "An Autoencoder Based Model for Detecting Fraudulent Credit Card Transaction," *Procedia Computer Science,* vol. 167, pp. 254-262, 2020.

[23] C. E. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," arXiv:1811.03378 [cs.LG], Glasgow, 2018.

[24] B. X. YONG and A. BRINTRUP, "Do Autoencoders Need a Bottleneck for Anomaly Detection?," *IEEE Access,* vol. 10, pp. 78455-78471, 2022.

[25] J. Chen, S. Yao and R. Ali, "Credit Card Fraud Detection Using Sparse Autoencoder and Generative Adversarial Network," in *2018 IEEE 9th Annual*

Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2018, pp. 1054-1059.

[26] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[27] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation,* vol. 13, no. 7, p. 1443–1471, 2001.

[28] Scikit-Learn, "Scikit-Learn," 12 20 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html. [Accessed 12 20 2023].