Tampere University

Zaffar Zaffar

Credit Card Fraud Detection using One-Class Classification Algorithms

Faculty of Information Technology and Communication Sciences (ITC)
Master's thesis
September 2023

Abstract

Zaffar Zaffar: Credit Card Fraud Detection using One-Class Classification Algorithms
Master's thesis
Tampere University
Master's Degree Program in Statistical Data Analytics
September 2023

Similar to most things in everyday life, the advent of payment cards also has good and bad sides. It undoubtedly, made life easier by bringing the whole payment system to a single card, but it also paved the way for a new set of illegal activities and frauds. The credit card fraud has been carried out since the payment cards came into existence, and since then, the trend in such frauds has been an increasing one. Therefore, a quest to attenuate the losses caused by such frauds began. For this purpose, many preventive and detective measures have been taken in the past, and new ways are sought to further improve the policies. These measures, however, reduce the losses temporarily only and have not yet succeeded in converting the uptrend in the losses by such frauds into a downtrend because fraudsters always come up with a new way of tricking the people and the system. Thus, a new way of solving this ever-existing challenge is needed, which can detect even those fraudulent instances that are executed by techniques and methods that are yet-to-be-invented by fraudsters. Moreover, the occurrence of normal (non-fraudulent) credit card transactions is much more than fraudulent ones, and therefore, the data for credit card fraud detection is highly imbalanced. Another challenge in credit card fraud detection systems is the high dimensionality of datasets. Therefore, to address the imbalance nature of the data, to cope with the curse of dimensionality with a new way of making the model to regulate and extract the discriminative features, and to detect the fraud carried out by yet-to-be-invented techniques, we implemented a set of novel and state of the art subspace learning-based One-Class Classification algorithms. We experimented with integrating a projection matrix and geometric data information in the training phase to improve credit card fraud detection. We also experimented by using a maximization-update rule in updating the projection matrix instead of the classical minimization-update rule in the subspace leaning-based data description. We found that the linear version of Graph-embedded Subspace Support Vector Data Description with kNN graph, gradient-based solution, and minimization-update rule works better than all other models.

Keywords: Credit card fraud detection, one-class classification, financial data processing, subspace learning, graph-embedding.

Preface

I want to express my heartfelt gratitude to my supervisors, Prof. Juho Kanniainen, and Dr. Fahad Sohrab, for their invaluable guidance, support, and encouragement throughout this journey. Their expertise has been instrumental in shaping this research. I also want to extend my sincere gratitude to Prof. Moncef Gabbouj for giving me the opportunity to work under his supervision in the Signal Analysis and Machine Intelligence (SAMI) research group.
I also want to extend my deepest thanks to my family for their unwavering love and encouragement. Your belief in me has been a driving force, and I am grateful for your constant support.
I would like to acknowledge the support of the NSF-Business Finland project AMALIA.
I hope that my work contributes positively to the field and serves as a small token of appreciation for all those who have been a part of this endeavor.

Contents

List of Figures

List of Tables

Nomenclature

$\mathbf{1}$	Vector containing all ones
$\mathbf{1}_c$	Vector with ones for instances belonging to class c, and zeros elsewhere
$\mathbf{a}$	Center of the data description
$\mathbf{A}$	Adjacency matrix
$\bar{\mathbf{A}}$	Diagonal matrix with α values
b	Threshold variable defining the position of the decision boundary
$\mathcal{B}$	Reconstruction vector
C	Percentage of outliers to be considered from the positive class
$\mathcal{C}$	Total number of classes
d	Dimension of data in subspace
D	Dimension of data in original space
$\mathbf{D}$	Diagonal matrix
$\mathbf{E}$	Covariance matrix
$\mathbf{I}$	Identity matrix
$\mathbf{J}$	Matrix with all entities equal to $\frac{1}{N}$
k	Kernel function
$\mathbf{k}$	Kernel vector
$\mathbf{K}$	Kernel matrix
$\hat{\mathbf{K}}$	Centralized kernel matrix
L	Lagrangian equation
$\mathbf{L}$	Laplacian matrix
$\mathbf{L}_{kNN}$	Laplacian matrix in the kNN graph
$\mathbf{L}_t$	Laplacian matrix in the PCA graph
$\mathbf{L}_\alpha$	Laplacian matrix constructed with α values
N	Total number of instances
N_c	Number of instances belonging to class c
$\mathbf{N}$	Neighborhood of a data point
O	Big O notation for complexity analysis
$\mathbf{q}$	Eigenvector in spectral solution
$\mathbf{Q}$	Projection matrix
r	Regularization parameter in GEOCSVM
R	Radius of the data description
R^*	Distance between the test sample and the center of the data description
$\mathbf{s}$	Matrix of support vectors
sgn	Sign function
$\mathbf{S}$	Matrix having the geometric data relationships
$\mathbf{S}_{kNN}$	Matrix having geometric data information in the kNN graph

$\mathbf{S}_{\mathbf{Q}}$	Matrix having the geometric relationships of data in the subspace
$\mathbf{S}_t$	Total scatter matrix
$\mathbf{S}_w$	Within class scatter matrix
$\mathbf{S}_\alpha$	Matrix having geometric data information constructed with $\mathbf{L}_\alpha$
$\bar{\mathbf{S}}$	Regularized $\mathbf{S}$ matrix
$\mathbf{t}$	Eigenvector in spectral regression solution
$tr(\cdot)$	Trace operator
$\mathbf{u}$	Center of the data description in the projected space
$\mathbf{U}$	Matrix with eigenvectors corresponding to $\mathbf{\Lambda}$ in each column
v	Eigenvalue in spectral and spectral regression solutions
$\mathbf{W}$	Vector normal to the hyperplane in OCSVM model
$\mathbf{x}$	D-dimensional input vector
$\mathbf{x}^*$	D-dimensional test data point
$\mathbf{X}$	Matrix of all training data
$\mathbf{y}$	d-dimensional input data transformed from the original D-dimensions
$\mathbf{y}^*$	d-dimensional test sample transformed from the original D-dimensions
$\mathbf{z}$	Data vector in d-dimensional space with embedded geometric information
$\mathbf{z}^*$	Test data vector in d-dimensional space with embedded geometric information
$\boldsymbol{\alpha}$	Vector of α values
α_i	Lagrange multiplier associated with i^{th} data point
β	Weight assigned to the regularization term
γ_i	Lagrange multiplier associated with $i^{th} datapoint$
ΔL	Gradient of L
η	Hyper-parameter specifying the step of gradient ΔL
λ	Parameter used to select different Ψ
$\mathbf{\Lambda}$	Diagonal matrix with non-negative eigenvalues
ξ	Slack variable
σ	Hyper-parameter that defines the width of the kernel
ϕ	Function which maps $\mathbf{x}$ to the kernel space
$\mathbf{\Phi}$	Matrix containing training data in kernel space
Ψ	Regularization term expressing the class variance
$(\cdot)^T$	Transpose of the given vector or matrix

List of abbreviations

Acc	Accuracy
ANN	Artificial neural network
ATM	Automated teller machine
CVC	Card verification code used by MasterCard
CVV	Card verification value
CVV2	Card verification code used by Visa
ESVDD	Ellipsoidal support vector data description
F1	F1-measure
fMRI	Functional magnetic resonance imaging
fn	False negatives
fp	False positives
GANN	Genetic algorithm with neural network
GEOCSVM	Graph-embedded one-class support vector machine
GESSVDD	Graph-embedded subspace support vector data description
GESVDD	Graph-embedded support vector data description
HMM	Hidden Markov model
kNN	k-nearest neighbor
LR	Logistic Regression
LTSM	Long short-term memory
M-SSVDD	Multi-modal subspace support vector data description
ML	Machine Learning
NPT	Non-linear projection trick
OCC	One-Class Classification
OCSVM	One-class support vector machine
PCA	Principal component analysis
Pre	Precision
RBF	Radial basis function
SOM	Self organizing maps
SSVDD	Subspace support vector data description
SVDD	Support vector data description
SVM	Support vector machine
tn	True negatives
tnr	True negative rate
tp	True positives
tpr	True positive rate
ULB	Université Libre de Bruxelle

1 Introduction

Fraud detection has been carried out since the beginning of modern technology, but still, billions of dollars are annually lost by different types of fraud. The methods, as well as the fraudsters, are evolving with time and technology, and hence, the quest for better ways and algorithms to detect, reduce and eventually stop fraud never ceases. This master's thesis, like many others, is another attempt to investigate, implement and compare a few (novel) machine learning (ML) methods (One-Class Classification algorithms) to find a more efficient way to save the huge amount of dollars lost by fraud.

1.1 Fraud

Fraud is a significant and ever-evolving problem that impacts people, businesses, and society as a whole. It is a broad term encompassing a spectrum of deceitful practices intended to exploit others and gain an unfair advantage. It can take many different forms, depending on the channel adopted to carry out the fraudulent activity. Some common forms of fraud are described below:

- Financial Fraud: This involves schemes such as accounting fraud and insider trading, where individuals manipulate financial information or misuse funds for personal gain.
- Identity Theft: In this type of fraud, criminals steal personal information to impersonate individuals and engage in unauthorized transactions or activities, often resulting in financial loss and damage to the victim's reputation.
- Insurance Fraud: This form includes staged accidents, false damage claims, fake injuries, and many other activities.
- Credit Card Fraud: Criminals obtain and use someone else's credit card information without permission, leading to unauthorized purchases and financial loss for both individuals and financial institutions.

Among the various kinds of fraud, our focus in this study is on credit card fraud and its detection. The credit card transaction can take place in two ways: either physically, which is done physically by being at the payment terminal, or virtually which takes place over the internet or telephone by using some PIN number and or proof of identification [1]. In order for someone to make a fraudulent credit card transaction in any of the two modes, either the credit card has to be stolen (in case of physical transactions), or the information necessary to use the credit card online has to be leaked or stolen (for online transaction) [1]. With the invention of such a luxury to shop online with a few clicks, online credit card fraud also got invented [2]. Figure 1.1 shows the patterns and trends in fraud cases reported in the US for different categories of fraud. This detailed graph of registered fraudulent reports clearly shows the increasing trend in credit card fraud over the span of 5 years.

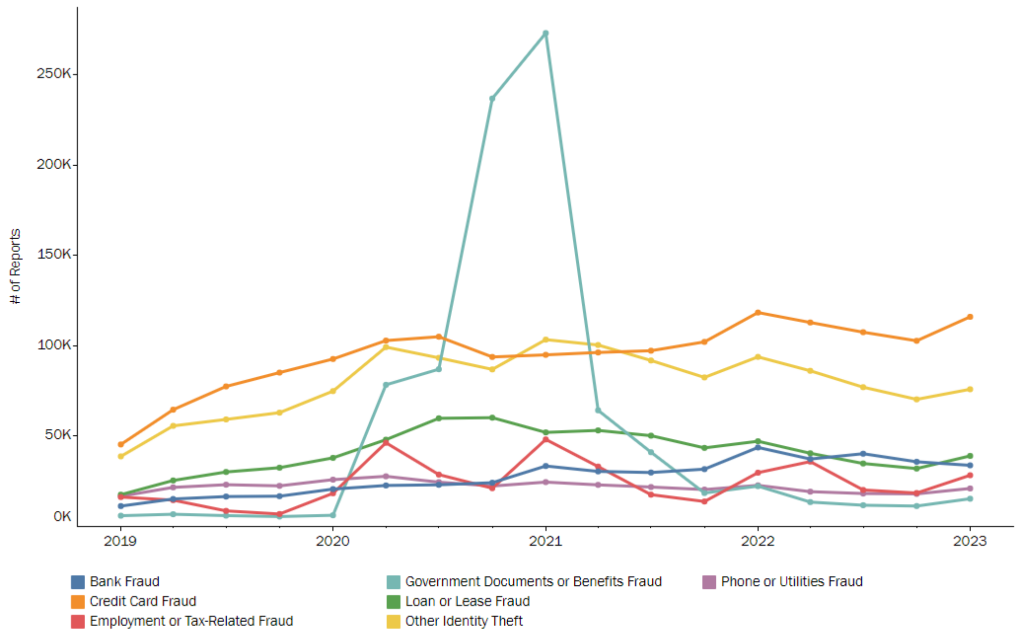

Figure 1.1 *Number of reports registered by US customers for different categories of fraud for the time period 2019-March 31,2023 [3]*

Credit card fraud, like all kinds of fraud, does not only cause monetary losses but also impacts trust, security, and the overall integrity of systems and institutions. Any effort to detect or prevent such fraudulent activity considerably impacts individual customers, organizations, and society. A few of the key benefits or impacts of credit card fraud detection systems are listed below:

- Reducing monetary losses: Credit card fraud detection systems reduce the potential financial losses, protecting individuals' and businesses' assets by promptly identifying and blocking unauthorized transactions.

- Protecting customers' assets and data: Providing a layer of protection by identifying and preventing unauthorized use of credit cards, customers can have confidence in the security of their accounts and data, which leads to increased trust and customer loyalty.

- Preserving customers' trust in organizations: Fraudulent activities can significantly damage the reputation of businesses and financial institutions, and therefore, systems mitigating such activities preserve the reputation of organizations and maintain trust among customers and stakeholders.

- Aiding law enforcement: Effective fraud detection systems provide valuable data and evidence that can assist law enforcement agencies in investigating and prosecuting fraudsters.

Credit card fraud detection, like any anomaly detection, usually has heavily imbalanced data, for example, normal (non-fraudulent) transactions happen almost every millisecond in the world, but fraudulent ones are not that common. Hence, the difference in the number of occurrences between the two classes is enormous. Furthermore, as mentioned before, new ways of fraud are being used every now and then, making it nearly impossible to incorporate every kind of fraud and all information of each of those kinds in the dataset. So, we have to develop a new set of solutions that can consider this imbalanced nature of the data while training and classifying any new instance accurately.

1.2 Fraud detection - a special kind of classification problem

The ML classification algorithms are of two types: supervised and unsupervised. In supervised learning, the dataset available is in the form of inputs and outputs; that is, we know the labels of the data for all categories, whereas in unsupervised methods, the labels are not specified, and data are usually clustered in groups and classified accordingly. In supervised methods, there comes a problem when the new data (that has to be classified) does not belong to any of the classes (for example, the fraudster came up with a new and unusual way of fraud) because the classification algorithms other than One-Class Classification (OCC) would classify the new data in one of the available classes. For example, we have two classes, cars and bicycles, and the new data is a tree. In this case, classifying the tree in either of the classes is entirely wrong and misleading.

In OCC algorithms, the data from the positive class is used for training the model, and everything that the model encounters that is not from the positive class is classified as negative. The class which is of interest and an ample amount of data is present from that class is considered positive, and the other or rest of the classes are considered negative. In the case of the cars and bicycles (binary classification) example, if the positive class is cars, then the classes would be cars and not-cars, and the tree would be classified as a negative (not-car category) object. In the case of multi-class classification problems, the one vs. all strategy is applied for predicting the class of the new data. In the same cars and bicycle example, we would then have three categories: cars, bicycles, and trees. The new tree object would be considered in all three car-vs-all, bicycle-vs-all, tree-vs-all possibilities, and assigned some probability to it being in each class. The class with the highest probability would be considered the class of the new tree object. A general overview of how an

OCC algorithm works is depicted in Figure 1.2.

The OCC algorithms are used when the data from one of the two classes is very scarce or very expensive to collect, and the model is trained based on one class (positive class) data. Another scenario where OCC algorithms are applied is when the data from one (negative) class is so diverse that it is practically impossible to statistically model the data from that class. Credit card fraud detection is one of the cases where these problems exist, and therefore, it is a special kind of classification problem where the OCC algorithms are ideal to be applied.

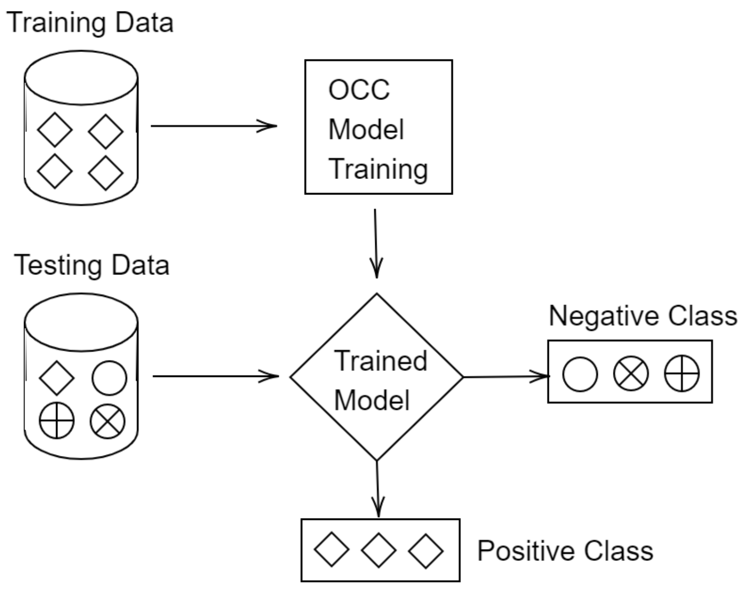

Figure 1.2 *A flowchart for an OCC algorithm showing the training and testing of the model. The training consists of only positive class objects, while the testing consists of instances from both positive and negative classes.*

1.3 Research aim

In this thesis, we are applying a few of the OCC models on financial datasets with the following research aims.

- Attempting to find a better solution for the ever-existent credit card fraud challenge.
- Addressing the imbalanced nature of the credit card transactions data in a way so that the models are unbiased towards the dense class.
- To come up with a model that can accurately distinguish the normal transactions from those which are carried out by existing as well as to-be-invented fraudulent ways.

- Implementing a novel way to reduce the curse of dimensionality by infusing the feature selection into the training phase.

1.4 Thesis structure

This document is structured as follows. Chapter 2 briefly discusses the work done on credit card fraud detection and OCC algorithms. In Chapter 3, we talk about the sources and specifications of the datasets considered and describe the methodology of the thesis, focusing on the steps taken during the research and the reasons why those steps were taken. In Chapter 4, the results are published and discussed, whereas Chapter 5 concludes the thesis with the deductions from the results and any future work that can be done in the continuity of this research.

2 Background knowledge

The background literature on credit card fraud detection and OCC algorithms is presented in the following sub-sections.

2.1 Credit card fraud

In the domain of credit card fraud, there are many different kinds of fraud based on the technique used for it. These various types are using someone's lost or stolen card to make transactions, non-receipt fraud, and using counterfeit credit cards [4]. The non-receipt fraud is when the card is somehow stolen before reaching the actual owner and used for fraudulent transactions. Counterfeit credit cards are the ones forged with the necessary information of someone's legitimate card gained by skimming, phishing, or some other way. Skimming is acquiring information from someone's credit card electronically while using it at some payment terminal [5]. The fraudsters use a device for skimming, which is usually installed on automated teller machines (ATMs) or any other payment terminal to record the information on the magnetic strip, including the card number and pin. In a phishing attack, a fraudster copies a website and lures the customer to provide personal and sensitive information containing card details which are then used for different kinds of frauds, including fraudulent transactions [6].

In general, we have two ways to deal with all kinds of fraud: either prevent them or detect and stop them. Many prevention techniques have been adopted in the last two decades. Visa uses card verification value (CVV), whereas MasterCard, along with the CVV, also uses the card validation code (CVC) embedded with the account number as a preventive measure [4]. CVC, which has different names based on the card-issuing company (for example, Visa uses CVV2 for this verification code), ensures that the initiator of the transaction has the card. It does not provide immunity against any other form of fraud [5]. Moreover, to verify that the actual owner received the card issued, card activation programs are used to activate the card only if the customer's identity is proven [7]. Apart from this, many other measures, such as address verification systems or keeping negative and positive lists (for example, to keep track of past offenders to avoid further fraud from those sources), are in use to ensure that transactions are safe and risk-free.

However, prevention alone is not the solution to this problem, as the fraudsters and or methods being used for fraud are adapting to the technology, and the losses due to such fraudulent transactions are increasing. Also, it is usually difficult to know in time that the preventive measures have failed to take other steps in order to stop the fraud that might occur. Therefore, a conjoint approach, having both preventive and detective measures, such that not only are the losses reduced but the cost of the strategy is also minimal, is the solution to this problem [5]. Since the manually detecting techniques are unfeasible, inaccurate, and inefficient, machine learning algorithms are deployed to learn the pattern and identify and stop fraudulent transactions in real-time [8]. Among the vast pool of statistical models and machine learning algorithms, neural networks are extensively used to detect fraudulent transactions. In [9], by implementing and comparing artificial neural network (ANN) to support vector machine (SVM) and k-nearest neighbor (kNN), it is found that ANN works better in terms of accuracy than SVM and kNN. Similarly, another study [10] compared ANN with logistic regression (LR) and found that ANN achieved a higher prediction rate than LR. In [11] genetic algorithm is used with a neural network (GANN) to detect and predict fraudulent transactions. A number of other models have been applied to solve the credit card fraud problem. Some of these models are decision trees [12, 13, 14], adaptive fraud detection based on knowledge discovery [15], self-organizing maps (SOM) [16, 17, 18], long short-term memory (LTSM) [19], hidden Markov model (HMM) [20, 21], fuzzy clustering [22, 23], graph- and modified butterfly optimization-based deep learning [24] and gradient boosting techniques [25, 26].

Any fraud or anomaly detection problem has two challenges, which are also the intrinsic properties of such problems. These issues are the imbalanced data and the fact that all kinds of fraud cannot be statistically modeled because the methods of fraud are ever-evolving. Once a method is exposed, measures are taken to stop further losses through that method, and fraudsters then try to find a new way to trick people or businesses. To handle the imbalanced nature of the data, previous implementations have employed data sampling methods. Some have used under-sampling [27], the method where the majority class instances are synthetically reduced to make the data from both classes equal, while some have used over-sampling [28] where the minority class data is synthetically generated to make the data from both classes equal. Some implementations have employed a mixture of under- and over-sampling techniques to overcome the imbalanced data issue [29].

Recently, there has been research on using neural networks along with OCC algorithms for anomaly or fraud detection [30, 31, 32]. These algorithms, although provide an end-to-end trained model, suffer from a drawback of crucial importance, that is, representation collapse [33]. Representation collapse is a phenomenon where a model fails to effectively capture the diversity and complexity of the input data, leading to similar or identical outputs for different inputs (possibly mapping all the input data to a single point).

To overcome the above-mentioned challenges without resorting to the synthetic alteration of data, we are using OCC algorithms, which use only the normal (non-fraud) transactions for the training. These algorithms handle the above-mentioned issues because normal transactions are available in abundance, and the model trained on such data will consider everything that is not normal as fraudulent, which includes fraud carried out by both existent and to-be-invented techniques.

2.2 One-class classification algorithms

OCC is a special type of classification problem in machine learning where the model is trained using the data from only one (positive or target) class with the goal of differentiating objects of positive class from other (negative) classes. These algorithms are employed in problems where the data from the negative class is either unavailable or expensive to gather. Over the last two decades, because of their use in peculiar problems which were not solvable through the non-OCC ML algorithms, these algorithms became the center of attention, and a lot of research has been done to improve these models and or implement them in various applications. For example, they have been employed to detect bots in Twitter [34], to detect and identify criminals using the glass pieces shattered at crime spots [35], to detect depressed patients based on the functional magnetic resonance imaging (fMRI) readings [36], for image segmentation and classification [37], for detecting abnormalities in video [32], for hyperspectral image analysis [38], to detect myocardial infection at early stages [39], to identify and classify rare insects [40, 41], to detect cyberattacks in vehicular networks [42], and detection of railway vehicle from audio data [43].

All of the OCC problems are solved by using three main approaches: density estimation, reconstruction-based, and border-based description [44]. Density-based approaches focus on estimating the underlying data density to distinguish between normal and abnormal instances. These methods assume that normal instances are generated from a high-density region, while abnormal instances lie in low-density regions [44]. The key idea is to model the density of the target class, and for this purpose, a few of the density modeling methods in practice are Parzen density [45] and Gaussian or a combination of multiple Gaussian models [46].

According to [44], in reconstruction-based approaches, a model is chosen to fit the data based on the prior information of data and the assumptions made about the data-generating process. During training, the model learns to minimize the difference between the original input and the reconstructed output. For the classification of new instances, reconstruction error is used; that is, the instances having lower reconstruction error are classified as normal, whereas the ones with higher reconstruction error are said to be outliers or belong to the negative class. A few of the methods used for the reconstruction are self-organizing maps [47] and k-means clustering [46].

According to [44], these two approaches work well when the target data can be accurately modeled using a specific distribution and the assumptions about the data generation process are reliable. However, a challenge in these approaches lies in determining an appropriate distribution that represents the given data accurately. On the contrary, a third category of techniques, namely the border-based approaches, exists, which does not rely on data distribution. These methods define the decision boundary that separates normal instances from anomalies by identifying the instances in the training data that are located at or lie close to the boundary between the classes. Such instances are called support vectors; therefore, these methods are also known as support vector-based approaches. All of the methods used in this thesis belong to the border-based category.

The OCC algorithms which are implemented and compared in this thesis are discussed one by one in the subsequent part.

2.2.1 One-Class Support Vector Machine

Support Vector Machine (SVM) is a powerful algorithm that classifies the data in the groups (or predicts a continuous variable in case of regression problems) by finding the optimal hyperplane which maximizes the margin or distance between the classes in data. In case the data is not linearly separable, that is, no hyperplane exists which can divide the data into the respective classes, then a kernel function is used, which projects the data to a higher dimensional space where it is linearly separable, find the hyperplane in the new high dimensional space which linearly classifies the data, and projects the data back to original dimensions.

The One-Class Support Vector Machine (OCSVM), which is our benchmark model for this study, is an OCC algorithm based on SVM; that is, it is trained by the data from only positive class to find a hyperplane such that the margin between the hyperplane and the positive class data points is maximum. In OCSVM a parameter C is used, which is the percentage of the positive class data to be considered as outliers. It is a trade-off between the number of support vectors and the percentage of data points that will be outside the boundary. There are two approaches for making the decision boundary between the normal data and the outliers: one is, using a mapping function using the density distribution, and another is, making the boundary between the origin and the data points while keeping the C amount of data points outside the boundary [48]. The latter is often used in the OCSVM algorithm because of its more straightforward implementation.

The mathematical form of the OCSVM problem, along with the constraints, is given by

$$\begin{aligned} & min \quad \frac{1}{2}\mathbf{W}^T\mathbf{W} + C\sum_i \xi_i - b, \\ & s.t. \quad \mathbf{W}\cdot\phi\left(\mathbf{x}_i\right) + b \geq b - \xi_i, \quad \xi_i \geq 0; \quad i = 1, 2, ..., N, \end{aligned} \tag{2.1}$$

where $\mathbf{W}$ is the normal vector to the hyperplane, $\mathbf{x}_i$ is the i^{th} observation, N is the number of observations, ϕ is the function which maps $\mathbf{x}_i$ to the kernel space, b is the threshold variable that defines the position of the decision boundary with respect to the origin, and ξ_i is the slack variable [49].

To solve the OCSVM problem, variables $\mathbf{W}$ and b have to be found such that they minimize (2.1). To do this, the optimization problem in (2.1) is re-formulated using Lagrange multipliers as

$$L = \frac{1}{2}\mathbf{W}^T\mathbf{W} + C\sum_i \xi_i - \sum_i \gamma_i\xi_i - \sum_i \alpha_i\left[\mathbf{W}\cdot\phi\left(\mathbf{x}_i\right) - b + \xi_i\right]. \tag{2.2}$$

By minimizing (2.2) with respect to $\mathbf{W}$, ξ, and b, and maximizing with respect to α, and γ, and back-substituting those equations into (2.2), we get the final form of the problem, that is,

$$\begin{aligned} & min \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right), \\ & s.t \quad 0 \leq \alpha_i \geq C, \quad \sum_i \alpha_i = 1, \end{aligned} \tag{2.3}$$

where $k\left(\mathbf{x}_i, \mathbf{x}_j\right)$ is $\phi\left(\mathbf{x}_i\right)\cdot\phi\left(\mathbf{x}_j\right)$. The problem in (2.3) is solved for the α_i values corresponding to the data points, where the data points with the positive α value are the support vectors that determine the boundary of the positive class. The α_i are used to find the threshold b, using

$$b = \sum_j \alpha_j k\left(\mathbf{x}_j, \mathbf{x}_i\right), \tag{2.4}$$

where $\mathbf{x}_i$ corresponds to any one of the support vectors. Any test data point $\mathbf{x}^*$ can then be classified into positive or negative class using the sign function (denoted by '*sgn*') as

$$f(\mathbf{x}^*) = sgn\left[\sum_i \alpha_i k\left(\mathbf{x}_i, \mathbf{x}*\right) - b\right]. \tag{2.5}$$

The test data points having $f(\mathbf{x}^*) < 0$ will be considered from positive class, and $f(\mathbf{x}^*) \geq 0$ will be considered from negative class.

2.2.2 Support Vector Data Description

An approach, other than OCSVM, to detect fraud is forming a data description enclosing the normal transaction data. One such model is the kernel-based model, known as Support Vector Data Description (SVDD) [50], which, unlike OCSVM, forms a hyper-sphere around the positive class data points. The goal is to form an optimal hyper-sphere of radius R and center $\mathbf{a}$ by minimizing the radius and hence the volume of the hyper-sphere containing positive class data points while keeping the outliers outside the boundary. To accommodate the outliers in our problem, we add slack variables $\xi_i \geq 0$. Mathematically, this is given as

$$\begin{gathered} min \quad R^2 + C\,\Sigma_i\,\xi_i, \\ s.t \quad \|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0; \qquad i = 1, 2, ..., N, \end{gathered} \tag{2.6}$$

where the hyper-parameter C is the trade-off between the volume of the sphere and the fraction of data points outside the boundary. The Lagrangian equation of (2.6) is formed by incorporating the constraints using the Lagrange multipliers α_i, γ_i as

$$L = R^2 + C\,\Sigma_i\,\xi_i - \sum_i \alpha_i \left[R^2 + \xi_i - \left(\|\mathbf{x}_i\|^2 - 2\mathbf{a}\cdot\mathbf{x}_i + \|\mathbf{a}\|^2\right)\right] - \sum_i \gamma_i\,\xi_i. \tag{2.7}$$

Equation (2.7) is minimized and maximized with respect to R, $\mathbf{a}$, ξ_i and α_i, γ_i, respectively. The resulting equations from minimization and maximization of the Lagrangian are back-substituted into (2.7), which re-formulate the problem as

$$\begin{gathered} L = \sum_i \alpha_i \left(\mathbf{x}_i \cdot \mathbf{x}_i\right) - \sum_{i,j} \alpha_i \alpha_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right), \\ s.t. \quad 0 \leq \alpha_i \geq C. \end{gathered} \tag{2.8}$$

The α_i, according to the constraint (2.8), can only be equal to or between 0 and C. Figure 2.1 shows the mapping of the data points according to their respective α_i values (and the data description for SVDD), which is described as follows:

- If α_i = 0, the data points will lie inside the hyper-sphere.
- If α_i = C, the data points will lie outside the hyper-sphere.
- If $0 < \alpha_i > C$, the data points will lie on the surface of the hyper-sphere. These are the points that are the support vectors for the data description.

The support vectors, that is, data points with α values between 0 and C, are used to calculate radius R of the data description as

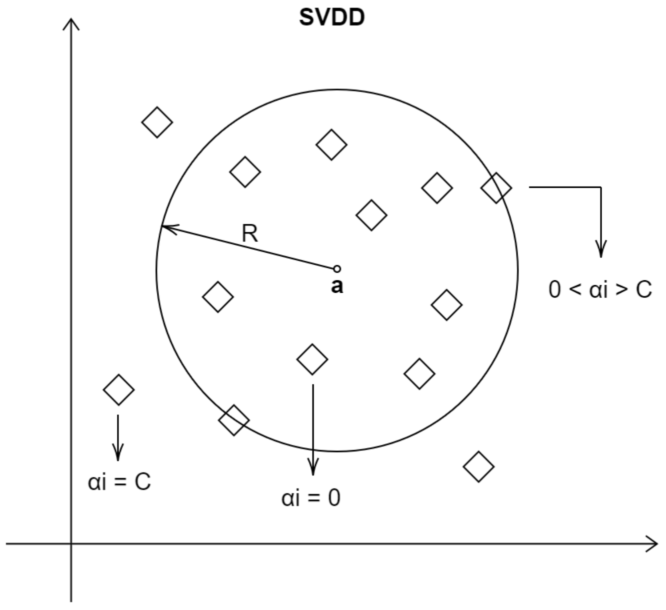

Figure 2.1 *The mapping of the data points with their alpha values in the case of SVDD in 2D. The center of the SVDD boundary is denoted by a, and R represents the circle's radius.*

$$R^2 = (\mathbf{x}_k \cdot \mathbf{x}_k) - 2\sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_k) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{2.9}$$

where $\mathbf{x}_k$ are the support vectors. Any test object, $\mathbf{x}^*$, can be checked if it is a normal data point or an outlier by calculating its distance from the center $\mathbf{a}$ of the hyper-sphere, using

$$\|\mathbf{x}^* - \mathbf{a}\|^2 = (\mathbf{x}^* \cdot \mathbf{x}^*) - 2\sum_i \alpha_i (\mathbf{x}^* \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \leq R^2. \tag{2.10}$$

If the distance is greater than R^2, the data point is classified as an outlier, otherwise normal.

2.2.3 Subspace Support Vector Data Description

To optimize the SVDD model and to reduce the impact of high dimensional feature space of the dataset, another model is proposed by [51] known as Subspace Support Vector Data Description (SSVDD), which finds an optimized subspace to map the original high-dimensional data to lower dimension for improved OCC. The low-dimensional data is then used to form a hyper-sphere boundary of radius R and center $\mathbf{a}$ around the positive class. Let D be the original feature-dimension of the data and d (d being smaller than or equal to D) be the subspace to which data is to be projected. The aim is to find a mapping function that projects the data from D to d feature space and find a hyper-sphere that encloses the positive class such that the radius R of the hyper-sphere is kept at a minimum. The projection matrix $\mathbf{Q}$ is used to transform the data to d-dimensional subspace, represented as

$$\mathbf{y}_i = \mathbf{Q}\mathbf{x}_i; \quad i = 1, 2, ..., N. \tag{2.11}$$

The problem formulation for SSVDD is the same as SVDD, except that the projected data is being used instead of the original one. Thus, the mathematical formulation of the SSVDD problem is

$$\begin{aligned} &min \quad R^2, \\ s.t \quad &\|\mathbf{Q}\mathbf{x}_i - \mathbf{a}\|_2^2 \leq R^2; \quad i = 1, 2, ..., N. \end{aligned} \tag{2.12}$$

To incorporate the slack variables into the model to have certain relaxation in the description, a set of ξ_i are added to the model in (2.12), as follows:

$$\begin{aligned} min \quad & R^2 + C\sum_i \xi_i, \\ s.t \quad & \|\mathbf{Q}\mathbf{x}_i - \mathbf{a}\|_2^2 \leq R^2 + \xi_i, \\ & \xi_i \geq 0; \quad i = 1, 2, , ..., N, \end{aligned} \tag{2.13}$$

where the term $C > 0$ is the trade-off hyper-parameter same as in SVDD. It regulates the volume of the hyper-sphere and the fraction of data points to be outside the description. The higher the value of C, the greater the fraction of data points outside the hyper-sphere will be.

Similar to the SVDD problem, Lagrange multipliers are used to express the problem in (2.13) in the Lagrangian form as

$$\begin{aligned} L(R, \mathbf{a}, \alpha_i, \xi_i, \gamma_i, \mathbf{Q}) = R^2 &+ C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i \\ &- \sum_{i=1}^{N} \alpha_i (R^2 + \xi_i - \mathbf{x}_i^T \mathbf{Q}^T \mathbf{Q}\mathbf{x}_i + 2\mathbf{a}^T \mathbf{Q}\mathbf{x}_i - \mathbf{a}^T \mathbf{a}). \end{aligned} \tag{2.14}$$

Equation (2.14) is to be minimized with respect to the radius R, center $\mathbf{a}$, slack variables ξ_i, and projection matrix $\mathbf{Q}$, and maximized with respect to the Lagrange multipliers α_i, and γ_i. The equation which results from back-substituting (2.11) and the end equations of minimizing and maximizing (2.14) into the (2.14) is given as

$$L = \sum_{i=1}^{N} \alpha_i \mathbf{y}_i^T \mathbf{y}_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \mathbf{y}_i^T \mathbf{y}_j \alpha_j. \tag{2.15}$$

In SSVDD, the alpha values and the projection matrix are interdependent components that work together to define the decision boundary and the representation of data points. The projection matrix affects the arrangement of data points in the subspace, while the alpha values define the decision boundary within that subspace, and therefore, they cannot be optimized simultaneously. For this reason, the equation (2.15) is maximized to get the optimized α_i values first. The next step is to find the optimal $\mathbf{Q}$ by optimizing the modified Lagrangian equation, given by

$$L = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^T \mathbf{Q}^T \mathbf{Q} \mathbf{x}_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \mathbf{x}_i^T \mathbf{Q}^T \mathbf{Q} \mathbf{x}_i \alpha_j - \beta \Psi, \tag{2.16}$$

where Ψ is the regularization term defining the class variance, and β is the weight assigned to the Ψ. The mathematical form of the Ψ is

$$\Psi = tr(\mathbf{Q}\mathbf{X}\lambda\lambda^T \mathbf{X}^T \mathbf{Q}^T), \tag{2.17}$$

where the trace operator $tr(\cdot)$ gives the sum of the elements on the main diagonal of the matrix given in the argument, and the vector λ, belonging to the vector space $\mathbf{R}^N$, is used to regulate or control the impact of each training sample on the regularization term Ψ.

The gradient of (2.16), denoted by ΔL, is used to update $\mathbf{Q}$, using the following update rule:

$$\mathbf{Q} \leftarrow \mathbf{Q} - \eta \Delta L, \tag{2.18}$$

where η is a hyper-parameter that specifies the step of the ΔL.

The parameter λ, which directly affects the regularization term Ψ, can take many different values where each value results in utilizing a different set of data points in updating $\mathbf{Q}$. The four different values for λ considered in this work are:

- If $\lambda_i = 0$, Ψ becomes irrelevant, and the optimization of $\mathbf{Q}$ is performed using equation (2.15).
- If $\lambda_i = 1$, all training samples contribute equally to the Ψ.
- In the case where $\lambda_i = \alpha_i$, both the outliers and the samples belonging to the class boundary are used in Ψ and in the update of $\mathbf{Q}$.
- In the fourth case, $\lambda_i = \alpha_i^C$, where α_i^C takes on the values α_i if $\mathbf{Q}\mathbf{x}_i$ is a support vector, and $\alpha_i^C = 0$ in other cases. That is, only the support vectors

are considered for the class variance, and hence, in the update of $\mathbf{Q}$.

Any test object $\mathbf{x}^*$ can be categorized into its respective class based on its distance from the center of the hyper-sphere. Since the model is trained on data projected into lower dimensional space d, the test object $\mathbf{x}^*$ must be projected into the same subspace d using the optimized projection matrix $\mathbf{Q}$ in (2.11). The distance of the d-dimensional test sample, denoted by $\mathbf{y}^*$, from the center of the hyper-sphere is calculated using

$$\|\mathbf{y}^* - \mathbf{a}\|_2^2 = \left(\mathbf{y}^{*T}\mathbf{y}^*\right) - 2\sum_{i=1}^{N}\alpha_i\mathbf{y}^{*T}\mathbf{y}_i + \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\mathbf{y}_i^T\mathbf{y}_j. \tag{2.19}$$

If the distance is less than R^2, the test sample $\mathbf{x}^*$ is categorized into positive class. Otherwise, it is considered from the negative class.

2.2.4 Ellipsoidal Support Vector Data Description

The training data is not always in a spherical shape, and therefore, to have a more flexible shape around the target class, a model known as Ellipsoidal Support Vector Data Description (ESVDD), proposed by [52], uses hyper-ellipsoid instead of a hyper-sphere. This is achieved by using the inverse of covariance matrix $\mathbf{E}$ in the optimization problem, given as

$$\begin{aligned} min \quad & R^2 + C\sum_i \xi_i, \\ s.t \quad & (\mathbf{x}_i - \mathbf{a})^T \mathbf{E}^{-1} (\mathbf{x}_i - \mathbf{a}) \leq R^2 + \xi_i, \\ & \xi_i \geq 0; \quad i = 1, 2, , ..., N, \end{aligned} \tag{2.20}$$

where C is the trade-off variable, $\mathbf{a}$ is the center of the hyper-ellipsoid, and ξ_i are the slack variables. The inverse of the covariance matrix $\mathbf{E}$ is given by

$$\mathbf{E}^{-1} = (\mathbf{X}\mathbf{X}^T)^{-1}. \tag{2.21}$$

Let us denote the projected center of the hyper-ellipsoid in the new feature space by $\mathbf{u} = \mathbf{E}^{-\frac{1}{2}}\mathbf{a}$. The problem formulated in (2.20) is re-expressed in terms of $\mathbf{u}$ as follows:

$$\begin{aligned} min \quad & R^2 + C\sum_i \xi_i, \\ s.t \quad & \left\|\mathbf{E}^{-\frac{1}{2}}\mathbf{x}_i - \mathbf{u}\right\|_2^2 \leq R^2 + \xi_i, \\ & \xi_i \geq 0; \quad i = 1, 2, , ..., N. \end{aligned} \tag{2.22}$$

The corresponding Lagrangian for the problem (2.22) is

$$\begin{aligned} L = R^2 + C\sum_i \xi_i - \sum_i \gamma_i \xi_i \\ - \sum_i \alpha_i (R^2 + \xi_i - (\mathbf{E}^{-\frac{1}{2}}\mathbf{x}_i)^T \mathbf{E}^{-\frac{1}{2}}\mathbf{x}_i + 2\mathbf{u}^T \mathbf{E}^{-\frac{1}{2}}\mathbf{x}_i - \mathbf{u}^T\mathbf{u}). \end{aligned} \tag{2.23}$$

Solving the Lagrange dual problem in (2.23) gives the equations for $\mathbf{u}$ and R as

$$\mathbf{u} = \sum_i \alpha_i \mathbf{E}^{-\frac{1}{2}}\mathbf{x}_i, \tag{2.24}$$

and

$$R = \sqrt{(\mathbf{E}^{-\frac{1}{2}}\mathbf{s})^T \mathbf{E}^{-\frac{1}{2}}\mathbf{s} - 2(\mathbf{E}^{-\frac{1}{2}}\mathbf{s})^T\mathbf{u} + \mathbf{u}^T\mathbf{u})}, \tag{2.25}$$

where $\mathbf{s}$ is any support vector, that is, a data point for which the α value is between *0* and C.

To classify any test instance $\mathbf{x}^*$ into either class, the distance between the test instance and the center, denoted by R^* is calculated using

$$R^* = \sqrt{(\mathbf{E}^{-\frac{1}{2}}\mathbf{x}^*)^T \mathbf{E}^{-\frac{1}{2}}\mathbf{x}^* - 2(\mathbf{E}^{-\frac{1}{2}}\mathbf{x}^*)^T\mathbf{u} + \mathbf{u}^T\mathbf{u})}. \tag{2.26}$$

The test instance is classified as positive if $R^* \leq R$ and vice versa.

2.2.5 Graph-Embedded One-Class Support Vector Machine

Graph-Embedded One-Class Support Vector Machine (GEOCSVM), proposed by [53], uses geometric information in data to find an optimal hyperplane that distinguishes the target class from the other class. The mathematical form of the model GEOCSVM is as follows:

$$\begin{aligned} min \quad & \frac{1}{2}\mathbf{W}^T\mathbf{S}\mathbf{W} + C\sum_{i=1}^{N}\xi_i - b, \\ & s.t. : \mathbf{W}^T\phi(\mathbf{x}_i) \geq b - \xi_i, \\ & \xi_i \geq 0; \quad i = 1, 2, ..., N, \end{aligned} \tag{2.27}$$

where $\mathbf{S}$ is the matrix containing geometric information of the data, $\phi(\mathbf{x}_i$ is the function that transforms the data from input feature space to the kernel space, $\mathbf{W}$ is the vector normal to the hyperplane, C is the trade-off variable, b is the threshold, and ξ_i are the slack variables. The geometric information is embedded into matrix $\mathbf{S}$ by the Graph Laplacian matrix $\mathbf{L}$, as

$$\mathbf{S} = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T, \tag{2.28}$$

where $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), ..., \phi(\mathbf{x}_N)]$. Among the various options for the choice of the graph, the within-class scatter matrix, denoted by $\mathbf{S}_w$, is used to exploit the local geometric relationships in the data. Mathematically, $\mathbf{S}_w$ is given by

$$\mathbf{S}_w = \mathbf{X}\mathbf{L}_w\mathbf{X}^T = \mathbf{X}\left(\mathbf{I} - \sum_{c=1}^{\mathcal{C}}\frac{1}{N_c}\mathbf{1}_c\mathbf{1}_c^T\right)\mathbf{X}^T, \tag{2.29}$$

where $\mathcal{C}$ is the total number of classes, N_c is the number of instances belonging to class c, and vector $\mathbf{1}_c$ contains elements that are equal to one for instances belonging to class c, and all other elements are set to zero. A few other graphs, employed in other models, are described in section (2.2.7).

The problem in (2.27) is redefined using a regularized $\mathbf{S}$ (denoted by $\bar{\mathbf{S}}$), and training data. The $\bar{\mathbf{S}}$ is given by

$$\bar{\mathbf{S}} = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T + r\mathbf{I}, \tag{2.30}$$

whereas, the decision boundary $\mathbf{W}$ is expressed in terms of the training data [54] as

$$\mathbf{W} = \mathbf{\Phi}\mathcal{B}, \tag{2.31}$$

where r is the regularization parameter, $\mathbf{I}$ is the identity matrix, and $\mathcal{B}$ is the reconstruction vector of dimension N. The redefined version of the problem is as follows:

$$\begin{aligned} min \quad & \frac{1}{2}\mathcal{B}^T(\mathbf{K}\mathbf{L}_w\mathbf{K}+r\mathbf{K})\mathcal{B}+C\sum_{i=1}^{N}\xi_i-b, \\ & s.t.: \mathcal{B}^T\mathbf{k}_i \geq b-\xi_i, \\ & \xi_i \geq 0; \quad i=1,2,...,N, \end{aligned} \tag{2.32}$$

where $\mathbf{K}=\mathbf{\Phi}^T\mathbf{\Phi}$ is the kernel matrix, and $\mathbf{k}_i$ is a vector having dot products of the i^{th} column with all remaining columns of $\mathbf{K}$. In order to solve the optimization problem (2.32), we have to transform it into the Lagrangian form using the Lagrange multipliers α_i, and γ_i as

$$\begin{aligned} L(\mathcal{B},\xi_i,b,\alpha_i,\gamma_i) = & \frac{1}{2}\mathcal{B}^T(\mathbf{K}\mathbf{L}_w\mathbf{K}+r\mathbf{K})\mathcal{B}+C\sum_{i=1}^{N}\xi_i-b \\ & -\sum_{i=1}^{N}\alpha_i(\mathcal{B}^T\mathbf{k}_i-b+\xi_i)-\sum_{i=1}^{N}\gamma_i\xi_i. \end{aligned} \tag{2.33}$$

The solution of (2.33) gives the reconstruction vector $\mathcal{B}$ as

$$\mathcal{B}=(\mathbf{K}\mathbf{L}_w\mathbf{K}+r\mathbf{K})^{-1}\mathbf{K}\boldsymbol{\alpha}, \tag{2.34}$$

where $\boldsymbol{\alpha}$ is the vector having the Lagrange multipliers α_i. Equation (2.34) is used in (2.31) to find the optimal decision boundary $\mathbf{W}$. Any test object $\mathbf{x}^*$ can be classified into the respective class using

$$f(\mathbf{x}^*)=\mathbf{W}^T\phi(\mathbf{x}^*)=\mathcal{B}^T\mathbf{k}^*-b, \tag{2.35}$$

where $\mathbf{k}^*=\mathbf{\Phi}^T\phi(\mathbf{x}^*)=[k(\mathbf{x}^*,\mathbf{x}_1),k(\mathbf{x}^*,\mathbf{x}_2),...,k(\mathbf{x}^*,\mathbf{x}_N)]$ is the vector having the dot products of the test object with all instances in the training data in feature space. The test object is said to be from the target class if $f(\mathbf{x}^*)\geq 0$.

2.2.6 Graph-Embedded Support Vector Data Description

The goal here is to implement SVDD by using the geometric information present in the data, that is, to use the geometric relationships in the data to form the smallest possible hyper-sphere of radius R and center $\mathbf{a}$ around the target class such that it contains most of the training data in the feature space. The problem formulated for the Graph-Embedded Support Vector Data Description (GESVDD) model by [53] is as follows:

$$\begin{aligned} min \quad & R^2 + C\sum_{i=1}^{N} \xi_i, \\ s.t.: (\phi(\mathbf{x}_i) - \mathbf{a})^T \bar{\mathbf{S}}^{-1} (\phi(\mathbf{x}_i) - \mathbf{a}) & \leq R^2 + \xi_i, \\ \xi_i \geq 0; \quad i & = 1, 2, ..., N, \end{aligned} \tag{2.36}$$

where $\bar{\mathbf{S}}$ is the regularized matrix containing the geometric relationships, ξ_i are the slack variables, and C is the trade-off parameter. The problem in (2.36) can be reformulated by defining a vector $\mathbf{u} = \bar{\mathbf{S}}^{\frac{1}{2}}\mathbf{a}$, as

$$\begin{aligned} min \quad & R^2 + C\sum_{i=1}^{N} \xi_i, \\ s.t.: \left\| \bar{\mathbf{S}}^{-\frac{1}{2}} \phi(\mathbf{x}_i) - \mathbf{u} \right\|_2^2 & \leq R^2 + \xi_i, \\ \xi_i \geq 0; \quad i & = 1, 2, ..., N. \end{aligned} \tag{2.37}$$

The Lagrangian form of (2.37), using Lagrange multipliers α_i and γ_i, is given as

$$L(R, \xi_i, \mathbf{u}, \alpha_i, \gamma_i) = R^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i - \sum_{i=1}^{N} \alpha_i \left(R^2 + \xi_i - \left\| \bar{\mathbf{S}}^{-\frac{1}{2}} \phi(\mathbf{x}_i) - \mathbf{u} \right\|_2^2 \right). \tag{2.38}$$

By solving the Lagrangian equation, we get the equations for the vector $\mathbf{u}$, $\mathbf{a}$, and an optimal radius R, given by

$$\mathbf{u} = \sum_{i=1}^{N} \alpha_i \bar{\mathbf{S}}^{-\frac{1}{2}} \phi(\mathbf{x}_i), \tag{2.39}$$

$$\mathbf{a} = \bar{\mathbf{S}}^{-1} \boldsymbol{\Phi} \boldsymbol{\alpha}, \tag{2.40}$$

and

$$R^2 = \left\{ min \left\| \bar{\mathbf{S}}^{-\frac{1}{2}} \phi(\mathbf{x}_i) - \mathbf{u} \right\|_2^2, \mathbf{x}_i \text{ is a support vector} \right\}. \tag{2.41}$$

Any test object $\mathbf{x}^*$ is classified into the target class if $f(\mathbf{x}^*) \geq 0$, where $f(\mathbf{x}^*)$ is given by

$$f(\mathbf{x}^*) = R - \left\| \bar{\mathbf{S}}^{-\frac{1}{2}} \phi(\mathbf{x}^*) - \mathbf{u} \right\|_2. \tag{2.42}$$

2.2.7 Graph-Embedded Subspace Support Vector Data Description

To incorporate the geometric relationships in data with the projection matrix used in (2.11) to transform the data into a lower dimensional subspace and train the model using the graph information to form a boundary around the target class with radius R, and center $\mathbf{a}$, a model known as Graph-Embedded Subspace Support Vector Data Description (GESSVDD), is proposed by [55]. The data in the D feature space is projected to the lower d-dimensional subspace. The data is assumed to be normalized by subtracting the mean of the training data. The problem for the model is formulated as follows:

$$\begin{aligned}
&min \quad R^2 + C\sum_{i=1}^{N}\xi_i, \\
&s.t. : (\mathbf{Q}\mathbf{x}_i - \mathbf{a})^T {\mathbf{S}_\mathbf{Q}}^{-1}(\mathbf{Q}\mathbf{x}_i - \mathbf{a}) \leq R^2 + \xi_i, \\
&\xi_i \geq 0; \quad i = 1, 2, ..., N,
\end{aligned} \tag{2.43}$$

where $\mathbf{S}_\mathbf{Q}$ is the matrix with the geometric data relationships in the projected subspace, $\mathbf{Q}$ is the projection matrix from (2.11), C is the trade-off, and ξ_i are the slack variables. The geometric information of the data in the d-dimensional subspace is embedded into the $\mathbf{S}$ matrix by using $\mathbf{L}$ and $\mathbf{Q}$, as given by

$$\mathbf{S}_\mathbf{Q} = \mathbf{Q}\mathbf{X}\mathbf{L}_x\mathbf{X}^T\mathbf{Q}^T = \mathbf{Q}\mathbf{S}_x\mathbf{Q}^T, \tag{2.44}$$

where $\mathbf{X}$ is the matrix having all of the training instances, and $\mathbf{L}_x$ denotes the availability of different $\mathbf{L}$ in the literature.

The optimization problem (2.43) is reformulated using a vector $\mathbf{u} = {\mathbf{S}_\mathbf{Q}}^{-\frac{1}{2}}\mathbf{a}$ as

$$\begin{aligned}
&min \quad R^2 + C\sum_{i=1}^{N}\xi_i, \\
&s.t. : \left\| {\mathbf{S}_\mathbf{Q}}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}_i - \mathbf{u} \right\|_2^2 \leq R^2 + \xi_i, \\
&\xi_i \geq 0; \quad i = 1, 2, ..., N.
\end{aligned} \tag{2.45}$$

where $\mathbf{u}$ is the center of the description in the projected subspace. The projected data vectors in this new feature space are represented as $\mathbf{z}_i = {\mathbf{S}_\mathbf{Q}}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}_i$. Similar to the previous methods, the problem will be solved using the Lagrange multipliers α_i, and γ_i. The Lagrangian form of the problem is

$$L = R^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\,(\,R^2 + \xi_i -$$
$$(\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}_i)^T\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}_i + 2\mathbf{u}^T\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}_i - \mathbf{u}^T\mathbf{u}\,) - \sum_{i=1}^{N}\gamma_i\xi_i. \tag{2.46}$$

The solution of (2.46) gives us the α_i values for each instance, and the equations for the projected center of the hyper-sphere $\mathbf{u}$ and the radius R as given by

$$\mathbf{u} = \sum_{i=1}^{N}\alpha_i\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}_i, \tag{2.47}$$

and

$$R = \sqrt{(\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{s})^T\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{s} - 2(\mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{s})^T\mathbf{u} + \mathbf{u}^T\mathbf{u}}. \tag{2.48}$$

The α value depicts the location of the data point in the projected subspace. The data point lies inside the boundary of the hyper-sphere if the α value is zero, lies on the boundary if $0 \geq \alpha \leq C$, or outside the boundary if $\alpha > C$. The matrix $\mathbf{s}$ contains the support vectors which are the data points corresponding to the α values between 0 and C.

Any test object $\mathbf{x}^*$ that has to be classified in either class must be transformed to the d-dimensional subspace using

$$\mathbf{z}^* = \mathbf{S_Q}^{-\frac{1}{2}}\mathbf{Q}\mathbf{x}^*, \tag{2.49}$$

and then the distance of the projected test data point from the center in the projected subspace is calculated. The test object is classified into the positive class based on the decision rule given by

$$\|\mathbf{z}^* - \mathbf{u}\|_2^2 \leq R^2, \tag{2.50}$$

that is, if the distance of the test object from the center is less than or equal to the radius R.

As from equation (2.44), $\mathbf{S}_x = \mathbf{X}\mathbf{L}_x\mathbf{X}^T$, we can use different graphs, $\mathbf{L}_x$, which will result in different variants of the model. Some of the graphs that we have used in our study are:

- Using the identity matrix, $\mathbf{I}$, instead of $\mathbf{L}_x$. This variant is denoted by GESSVDD-I in our experiments and results.
- Replacing $\mathbf{S}_x$ with $\frac{1}{N}\mathbf{S}_t$, which is named as PCA graph, and therefore, the variant is denoted by GESSVDD-PCA in results. The total scatter matrix $\mathbf{S}_t$, which defines the scatter of data points in the feature space, is expressed as

$$\mathbf{S}_t = \mathbf{X}\mathbf{L}_t\mathbf{X}^T = \mathbf{X}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\mathbf{X}^T, \tag{2.51}$$

where $\mathbf{I}$ is an identity matrix, $\mathbf{1}$ is a vector of ones, and N is the number of data points.

- The geometric information in the data is also exploited using k-nearest neighbor (kNN). In this case, the corresponding $\mathbf{S}_x$ becomes

$$\mathbf{S}_x = \mathbf{S}_{kNN} = \mathbf{X}\mathbf{L}_{kNN}\mathbf{X}^T, \tag{2.52}$$

where $\mathbf{L}_{kNN} = \mathbf{D}_{kNN} - \mathbf{A}_{kNN}$ and $\mathbf{D}$ and $\mathbf{A}$ are the diagonal and adjacency matrices, respectively. The $\mathbf{A}$ matrix has 1 if the data point is in the "neighborhood" and 0 otherwise. Mathematically,

$$[\mathbf{A}_{ij}] = \left\{ \begin{array}{ll} 1, & if \quad \mathbf{x}_i \in \mathbf{N}_j \quad or \quad \mathbf{x}_j \in \mathbf{N}_i \\ 0, & \text{otherwise} \end{array} \right\}, \tag{2.53}$$

where $\mathbf{N}_i$ denotes the neighborhood of the ith data point. This graph is denoted by GESSVDD-kNN in our results.

Moreover, each of these variants has been solved using three approaches: gradient-based where we use the gradient of $\mathbf{L}$ and update $\mathbf{Q}$ as in (2.18), spectral, where we solve eigenvalue problem $\mathbf{S}_\alpha\mathbf{q} = v\mathbf{S}_x\mathbf{q}$ and update $\mathbf{Q}$, or spectral regression-based method, where we solve eigenvalue problem $\mathbf{L}_\alpha\mathbf{t} = v\mathbf{L}_x\mathbf{t}$ and update $\mathbf{Q}$. The $\mathbf{q}$ and $\mathbf{t}$ are the eigenvectors for the respective problems, and the v is the eigenvalue in both cases. The equations for $\mathbf{S}_\alpha$ and $\mathbf{L}_\alpha$ are given as

$$\mathbf{S}_\alpha = \mathbf{X}\mathbf{L}_\alpha\mathbf{X}^{\mathbf{T}}, \tag{2.54}$$

and

$$\mathbf{L}_\alpha = \bar{\mathbf{A}} - \boldsymbol{\alpha}\boldsymbol{\alpha}^T, \tag{2.55}$$

where $\boldsymbol{\alpha}$ is a vector containing α_i values, and $\bar{\mathbf{A}}$ is a diagonal matrix with α_i values.

As there is no universally recommended approach to exclusively maximize or minimize each solution, we conducted experiments employing both strategies, referred to as "max" and "min", within the context of subspace-based model. In the gradient-based technique, ascending ($\mathbf{Q} \leftarrow \mathbf{Q} + \eta\Delta L$) and descending (as in (2.18)) steps in the update rule are applied for the purpose of maximizing and minimizing, respectively. Conversely, for the remaining two methods, the selection of the highest and lowest sets of positive eigenvalues and their corresponding eigenvectors is used for maximizing and minimizing, respectively.

2.3 Non-linear Projection Trick

A technique known as Non-Linear Projection Trick (NPT), is applied to use all of the models mentioned above in a non-linear setting. Unlike other non-linear transformation methods, NPT explicitly maps the training data into a kernel space using eigenvalue decomposition of the kernel matrix [56], which is formulated using the kernel function. One of the many such kernel functions is the radial basis function (RBF), given by

$$\mathbf{K}_{ij} = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right), \tag{2.56}$$

where σ is the hyper-parameter that defines the width of the kernel. The kernel matrix $\mathbf{K}$ in (2.56) is centralized as

$$\hat{\mathbf{K}} = (\mathbf{I} - \mathbf{J})\mathbf{K}(\mathbf{I} - \mathbf{J}), \tag{2.57}$$

where $\mathbf{I}$ is an identity matrix of dimensions NxN and $\mathbf{J}$ is calculated using

$$\mathbf{J} = \frac{1}{N}\mathbf{1}\mathbf{1}^T, \tag{2.58}$$

where $\mathbf{1}$ is an N-dimensional vector having 1 as its elements. The eigenvalue decomposition is then performed on $\hat{\mathbf{K}}$, given by

$$\hat{\mathbf{K}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \tag{2.59}$$

which gives a diagonal matrix $\mathbf{\Lambda}$ containing non-negative eigenvalues and a matrix $\mathbf{U}$ containing the corresponding eigenvectors. Finally, the data in the reduced dimensional kernel space is obtained using

$$\mathbf{\Phi} = \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^T, \tag{2.60}$$

where $\mathbf{\Phi}$ represents the transformed data. This $\mathbf{\Phi}$ can be used instead of the original data matrix $\mathbf{X}$ in any linear method to achieve a non-linear transformation. The transformation due to any kernel method like NPT is shown in Figure 2.2.

2.4 Performance measures

Performance measures are essential in evaluating the effectiveness and accuracy of machine learning models. A broad spectrum of performance measures is available, and depending on the nature of the data and the task at hand, different measures are used. For OCC or any binary classification task, all measures, in one way or another, count the number of correctly classified objects from a positive and negative

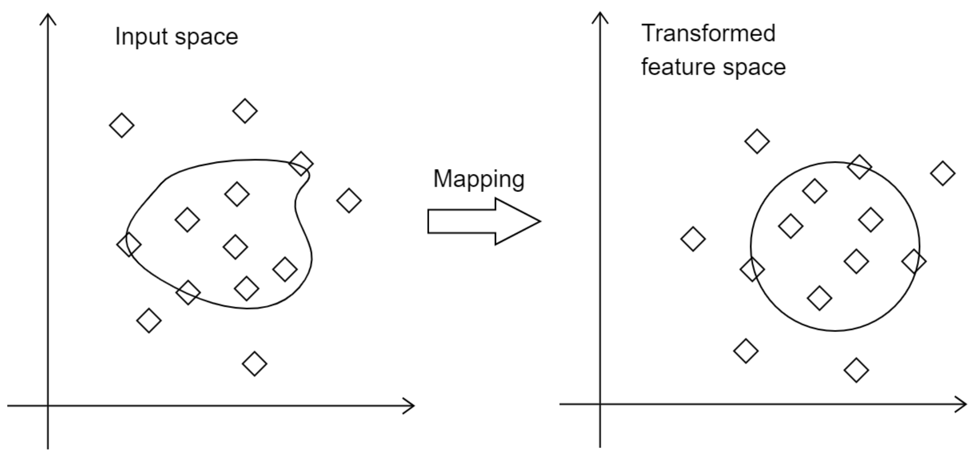

Figure 2.2 *Kernel transformation*

class. The counts used for these measures are:

- **True positives**: True positives, denoted by *tp*, is the count of correctly classified positive class objects.
- **True negatives**: True negatives, denoted by *tn*, is the count of correctly classified negative class objects.
- **False positives**: It represents the number of instances from the negative class that are classified into the positive class. It is denoted by *fp*.
- **False negatives**: It represents the number of instances from the positive class that are classified into the negative class. It is denoted by *fn*.

Here are some of the common performance measures for machine learning shown in the context of binary class classification problems:

2.4.1 Accuracy

Accuracy is a widely used performance measure that calculates the proportion of correctly predicted instances, both positive and negative, over the total number of instances. It provides a general overview of how well the model is performing. Mathematically, it is given by

$$Accuracy = \frac{tp + tn}{N}, \tag{2.61}$$

where $N = tn + tp + fn + fp$.

2.4.2 Precision

Precision measures the proportion of true positives out of all predicted positives. It indicates the model's ability to correctly identify positive instances and avoid false positives. It is expressed as

$$Precision = \frac{tp}{tp + fp}. \tag{2.62}$$

2.4.3 True positive rate

The true positive rate, denoted by *tpr*, calculates the proportion of true positives out of all actual positive instances. It quantifies the model's ability to identify all relevant positive instances. True positive rate, also known as sensitivity or recall, is given by

$$tpr = \frac{tp}{tp + fn}. \tag{2.63}$$

2.4.4 True negative rate

The true negative rate, denoted by *tnr*, measures the proportion of true negatives out of all actual negative instances. It is useful in binary classification tasks where identifying true negatives is of primary importance. True negative rate, also known as specificity, is expressed as

$$tnr = \frac{tn}{fp + tn}. \tag{2.64}$$

2.4.5 F-measure

The F-measure, also known as F1-measure or F-score, is the harmonic mean of precision and recall.It is often used while working with imbalanced datasets as it provides a balance between precision and recall and quantifies the model's ability to identify positive instances correctly (precision) and to capture all actual positive instances (recall). Mathematically, the F-measure is written as

$$F - measure = 2\,\mathrm{x}\,\frac{precision\ \times\ tpr}{precision + tpr}. \tag{2.65}$$

2.4.6 G-mean

The G-mean, or geometric mean, is a performance measure that combines the evaluation of both sensitivity and specificity in binary classification tasks. It provides a balanced assessment of a model's ability to correctly identify both positive and negative instances. Like F-measure, it is often used in classification problems having highly imbalanced datasets. It can be expressed as

$$G - mean = \sqrt{tpr \times tnr}. \tag{2.66}$$

3 Datasets and methodology

3.1 Datasets

In this thesis, four different datasets are used for detecting and predicting fraudulent credit card transactions. All of these datasets have been taken from Kaggle[1] open-source datasets. The details of each of these datasets are given below, which are summarized in Table 3.1.

The dataset-1 [57] was collected for research on data mining and fraud detection by the Worldline and Machine Learning Group at Université Libre de Bruxelles (ULB). The dataset consists of credit card transactions made by European cardholders over the span of two days in September 2013. It contains a total of 284,807 transactions with 29 features. Out of the 29 features, 28 are the result of PCA (principal component analysis) transformation, and the other feature is the amount of transaction. The background information about the original features from which these 28 have been extracted is confidential and, therefore, not publicly available. Moreover, the dataset is highly imbalanced, with only 492 fraudulent transactions, which make up about 0.172% of all transactions.

The dataset-2 used contains the transactions of digital payments. It has 7 features and 1 million instances, of which only 87,403 are fraudulent, and the rest are normal (non-fraudulent) transactions. So the dataset is highly imbalanced with a ratio of 0.087.

The dataset-3 is generated synthetically using a simulator called PaySim [58]. A sample of real mobile transactions for a period of one month is used to generate this synthetic dataset. It has 5 features and 1,048,575 transactions, out of which only 1142 are fraudulent, which makes the skewness ratio 0.001.

A dataset from some banks for fraud detection, available on Kaggle, is used as dataset-4 in this study. It has 112 features and 20,468 transactions, out of which only 5437 are fraudulent, making up 26.6% of the dataset.

Dataset	**Total Instances**	**Resampled Training Instances**	**Testing Instances**	**Features**	**Imbalance-ratio**
1	284,807	3144	85,443	29	0.12
2	1,000,000	3000	300,000	7	0.2
3	1,048,575	3000	314573	5	0.2
4	20,468	3000	6141	112	0.2

Table 3.1 *Descriptive statistics of datasets used. The imbalance-ratio is the ratio of fraudulent to normal transactions in the resampled training data.*

[1]https://www.kaggle.com/

3.2 Methodology

A general overview of the process as a whole is described in Figure 3.1.

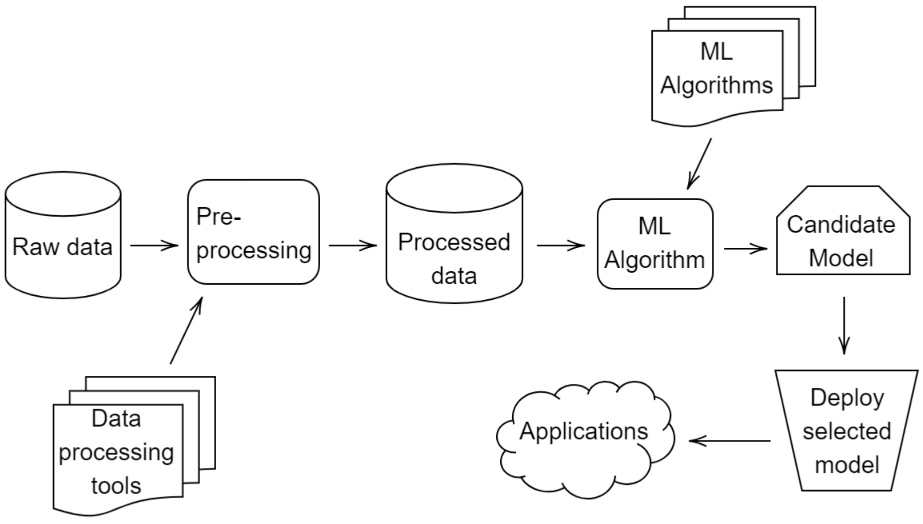

Figure 3.1 *Flowchart of the general methodology*

3.2.1 Data pre-processing

All of the datasets used are divided into train and test data based on the 70-30 split rule. The target class in the 70% train data is first used to calculate the mean and standard deviation, which are used for normalization. Since most models have the complexity of $O(N^3)$ [59], it is not possible to train the model with the positive class of 70% train dataset because of the high number of instances. So, the training dataset is resampled in a way to maintain the skewed nature of the data. For this reason, the random resampling is done to extract 500 fraudulent (or a maximum number of fraudulent instances, if the maximum is less than 500) and 2500 (except for dataset-1 where we have extracted 3144) normal transactions, which gives the ratio of fraudulent to normal transactions of 0.2, which is shown in Table 3.1. This reduced train dataset is then normalized using the mean and standard deviation calculated in the beginning.

3.2.2 Different variants of models selected

The datasets after prepossessing in 3.2.1 are used to train the models described in 2.2. Each of the algorithms used has different variants. On a general level, each has a linear and non-linear version (except GEOCSVM and GESVDD, which have only a non-linear version), and a few algorithms have many other variants depending on the choice of regularization term or some other criterion. The OCSVM, SVDD, and ESVDD have only linear and non-linear while the SSVDD and graph-based SSVDD, in addition to the linear and non-linear, also have other variants. The NPT, discussed in 2.3, is used to implement the non-linear version of these models.

Based on the Ψ term, which depends on the choice of value for λ, the four different variants of SSVDD are mentioned in section 2.2.3. Based on the update rule used for updating the projection matrix $\mathbf{Q}$, SSVDD has two kinds of variants: one, referred to as minimization, in which the update rule is the same as in (2.18), and second, referred to as maximization, in which the update rule is as $\mathbf{Q} \leftarrow \mathbf{Q} + \eta \Delta L$. So in total, 16 different variants of the SSVDD model have been implemented.

Like the SSVDD, GESSVDD has linear, non-linear, and minimization, maximization variants. Apart from that, as discussed in 2.2.7, GESSVDD has been implemented using three different graphs, and each of these three versions has been solved using all three solving techniques. So, altogether, 36 different variants for GESSVDD are implemented and compared during this study.

3.2.3 Models training

In order to train the models, the hyper-parameters are fine-tuned by using 5-fold cross validation. A range is decided for every hyper-parameter depending upon its scope, and the performance of the model using each value in the range is evaluated and compared with each other to get the best-performing value in the range for that specific parameter. All of the performance measures described in 2.4 are reported for every model. Since the g-mean provides a balanced assessment of both positive and negative instances, it is used as a metric of evaluation in cross-validation and in the comparison of models after testing. The hyper-parameters and their range of values are given below:

- $C \rightarrow$ [0.1 0.2 0.3 0.4 0.5].
- $d \rightarrow$ [1 2 3 4 5 10 20].
- $\beta \rightarrow$ [0.01, 0.1, 1, 10, 100].
- $\eta \rightarrow$ [0.1, 1, 10, 100, 1000].
- $\sigma \rightarrow$ [0.1, 1, 10, 100, 1000].

The number of iterations for all iterative methods and the number of neighbors for the kNN graph are both set to 5.

4 Results and discussion

The performance based on G-mean for linear and non-linear versions of all models (and their variants) for each of the four datasets are given in Tables 4.1 and 4.2 whereas the results for the rest of the measures described in 2.4 are in Appendix. From the analysis of Tables 4.1 and 4.2, it is found that for dataset-1, the GESSVDD model performed better. The linear version and the minimization-update rule for the GESSVDD rank higher than the respective counter-versions. It is also noticed that the kNN graph and the gradient-based solution technique outperform all other counter-variants for this dataset. For all other datasets, a graph-based non-linear model GEOCSVM performed better than all other models.

The different variants of SSVDD, based on the regularization term, Ψ, were analyzed, and it is found that Ψ_0 gives better results for dataset 1. In contrast, for all other datasets, all variants of Ψ work the same. Also, an overall comparison was made on the basis of the average of g-mean, and it is found that Ψ_0 ranks higher than the other versions. This shows that for the datasets in consideration, the regularization term adds no extra information, and the problem could be optimized by solving the Lagrange equation without the added regularization term (2.15) formulated for the respective model.

In the analysis of graph-based vs. non-graph-based models, it is found that for our considered datasets, the geometric knowledge present in the data added extra information, and hence, graph-embedded models, in general, perform better than the non-graph-based models. Furthermore, it is seen that the kNN graph works significantly better than the other considered graphs. The eigenvalue decomposition and gradient-based solution both work almost the same for all datasets.

In each dataset, few models have very high tnr and very low tpr or vice versa. One example is the SSVDD with Ψ_1 for dataset 1, which shows a tpr of 1.00 and tnr of 0.00. Such models overfit the data on the positive class; therefore, in testing, it fails to predict the objects from the negative class (fraudulent transactions). In the opposite case, models like ESVDD for dataset 1, where tpr is 0.0, and tnr is 1.0, form an extremely small boundary during training, which practically excludes all of the positive instances along with the negative ones.

For almost all datasets, SSVDD shows behavior that for the linear versions of respective variants, the tpr is quite high, and the tnr is quite low. In contrast, the non-linear versions of the same variants exhibit an opposite behavior (high tnr and low tpr). This shows that the linear models struggle to capture the complexities of the data distribution and are sensitive to outliers that are close to the decision boundary, leading to a high tpr and a low tnr. On the other hand, the NPT, employed to capture the non-linear relationships in the data, makes a flexible decision boundary which makes the model sensitive to the positive class, and therefore, misclassifies the normal transactions as fraudulent ones.

Overall, based on the experiments and results, a model that performed reliably (on average) for every dataset is the linear version of GESSVDD with the kNN graph, gradient-based solution, and minimization-update rule. Moreover, the tnr and tpr values are also reasonable across all datasets, which shows that this model can accurately detect (in most cases) both fraudulent and normal transactions. Therefore, we recommend this model for the given datasets for credit card fraud detection.

Model	**G-mean**				**Average of**
	Dataset-1	**Dataset-2**	**Dataset-3**	**Dataset-4**	**G-means**
GESSVDD-kNN-G-min	**0.906**	0.551	0.603	0.644	**0.676**
GESSVDD-kNN-G-max	0.849	0.541	0.603	**0.691**	0.671
GESSVDD-kNN-E-min	0.640	0.326	0.697	0.598	0.565
GESSVDD-kNN-E-max	0.686	**0.692**	0.603	0.501	0.620
GESSVDD-kNN-S-min	0.296	0.472	**0.728**	0.410	0.477
GESSVDD-kNN-S-max	0.296	0.214	0.638	0.472	0.405
GESSVDD-PCA-G-min	0.432	0.282	0.595	0.074	0.346
GESSVDD-PCA-G-max	0.644	0.192	0.605	0.070	0.378
GESSVDD-PCA-E-min	0.777	0.177	0.624	0.055	0.408
GESSVDD-PCA-E-max	0.720	0.186	0.595	0.082	0.396
GESSVDD-PCA-S-min	0.164	0.178	0.595	0.241	0.294
GESSVDD-PCA-S-max	0.082	0.193	0.595	0.243	0.279
GESSVDD-I-G-min	0.116	0.209	0.595	0.401	0.330
GESSVDD-I-G-max	0.000	0.123	0.595	0.527	0.311
GESSVDD-I-E-min	0.725	0.170	0.624	0.434	0.488
GESSVDD-I-E-max	0.493	0.186	0.595	0.429	0.426
GESSVDD-I-S-min	0.164	0.086	0.595	0.085	0.233
GESSVDD-I-S-max	0.082	0.205	0.610	0.113	0.252
SSVDD-Ψ_0-min	0.438	0.204	0.407	0.150	0.300
SSVDD-Ψ_0-max	0.391	0.162	0.404	0.123	0.270
SSVDD-Ψ_1-min	0.000	0.204	0.407	0.150	0.190
SSVDD-Ψ_1-max	0.000	0.162	0.404	0.123	0.172
SSVDD-Ψ_2-min	0.183	0.204	0.407	0.150	0.236
SSVDD-Ψ_2-max	0.183	0.162	0.404	0.123	0.218
SSVDD-Ψ_3-min	0.182	0.204	0.407	0.150	0.236
SSVDD-Ψ_3-max	0.182	0.162	0.404	0.123	0.218
OCSVM	0.446	0.559	0.227	0.355	0.397
SVDD	0.198	0.185	0.404	0.216	0.251
ESVDD	0.000	0.187	0.595	0.035	0.204

Table 4.1 *Performance of linear models based on G-mean across all datasets.*

Model	G-mean				Average of G-means
	Dataset-1	Dataset-2	Dataset-3	Dataset-4	
GESSVDD-kNN-G-min	0.329	0.264	0.570	0.283	0.362
GESSVDD-kNN-G-max	0.070	0.314	0.559	0.345	0.322
GESSVDD-kNN-E-min	0.007	0.346	0.576	0.321	0.313
GESSVDD-kNN-E-max	0.000	0.346	0.576	0.321	0.311
GESSVDD-kNN-S-min	**0.522**	0.365	0.580	0.052	0.380
GESSVDD-kNN-S-max	0.550	0.365	0.580	0.066	0.390
GESSVDD-PCA-G-min	0.003	0.110	0.592	0.347	0.263
GESSVDD-PCA-G-max	0.446	0.019	0.592	0.335	0.348
GESSVDD-PCA-E-min	0.181	0.591	0.547	0.114	0.358
GESSVDD-PCA-E-max	0.000	0.301	0.533	0.113	0.237
GESSVDD-PCA-S-min	0.417	0.627	0.227	0.121	0.348
GESSVDD-PCA-S-max	0.341	0.627	0.579	0.121	0.417
GESSVDD-I-G-min	0.003	0.060	0.555	0.192	0.203
GESSVDD-I-G-max	0.494	0.043	0.601	0.323	0.365
GESSVDD-I-E-min	0.007	0.360	0.557	0.068	0.248
GESSVDD-I-E-max	0.000	0.360	0.497	0.133	0.247
GESSVDD-I-S-min	0.151	0.268	0.557	0.469	0.361
GESSVDD-I-S-max	0.000	0.268	0.557	0.129	0.238
SSVDD-Ψ_0-min	0.215	0.384	0.089	0.398	0.272
SSVDD-Ψ_0-max	0.208	0.380	0.244	0.398	0.308
SSVDD-Ψ_1-min	0.194	0.384	0.089	0.398	0.266
SSVDD-Ψ_1-max	0.173	0.380	0.244	0.398	0.299
SSVDD-Ψ_2-min	0.193	0.384	0.089	0.398	0.266
SSVDD-Ψ_2-max	0.386	0.380	0.244	0.398	0.352
SSVDD-Ψ_3-min	0.197	0.384	0.089	0.398	0.267
SSVDD-Ψ_3-max	0.208	0.380	0.244	0.398	0.308
OCSVM	0.131	0.574	0.102	0.662	0.367
SVDD	0.198	0.073	0.039	0.179	0.122
ESVDD	0.214	0.048	0.592	0.108	0.241
GEOCSVM	0.183	**0.937**	**0.791**	**0.714**	**0.656**
GESVDD	0.215	0.913	0.593	0.694	0.604

Table 4.2 *Performance of non-linear models based on G-mean across all datasets.*

5 Conclusion

Credit card fraud detection is still a challenge, even in this era of advanced and AI-driven technology. This is owed to the imbalanced nature of data and the ever-evolving techniques of fraud. Furthermore, the curse of dimensionality is also a problem that has to be dealt with in a way to let the model itself extract meaningful features. To address these inherent properties of the problem in a way to not synthetically alter the proportion of data classes, we use OCC algorithms, and specifically subspace-based models which also cope with the high-dimensionality problem, to efficiently and effectively learn the patterns in the data and predict the fraudulent transactions and eventually mitigate the losses caused by such fraudulent activities.

For this thesis, we used four datasets taken from Kaggle. These datasets are highly imbalanced, with the percentage of minority class instances as low as 0.001%. All of the datasets have a high number of total instances which required a very high computational power, given that the models implemented in this thesis have the complexity of $O(N^3)$. Therefore, the datasets were resampled while keeping the imbalanced nature of the data. We used these resampled datasets to train 60 (in total) different variants of SVM, SVDD, SSVDD, graph-embedded versions of these models, and ESVDD.

From the results, we found that the linear version of GESSVDD with kNN graph, gradient-based solution, and minimization-update rule works better than all other models and variants for all datasets. The cost of miss-classifying a fraudulent transaction as normal is high in terms of monetary value, and the cost of miss-classifying a normal one as a fraudulent transaction is high in terms of customer trust and satisfaction. Therefore, the miss-classification of both classes' instances is crucial and must be avoided. For this reason, we chose g-mean as an evaluation metric because it provides a balanced assessment of both positive and negative instances.

However, one of the shortcomings we faced during this thesis is the high computational power needed to train the models. These models have to be improved in terms of complexity and efficiency to be able to get trained with fewer resources in terms of time and computing power. Moreover, another limitation of this work is the unavailability of real-world datasets. The banks and financial institutions, because of confidentiality issues, do not publish real data publicly. In some cases, like in the first dataset we used, the original features of the dataset have been transformed using some feature-extraction process (in our case PCA) and then anonymized using random names, which makes it difficult to interpret and extract meaningful features from the data using hand-crafted methods. This transformation and anonymization also affect the interpretation of the results and to infer meaningful insights from the results.

For future work, the existing models can be modified in terms of complexity to work efficiently in real-time. Moreover, in the quest to get better results, these models can also be modified by using different kernels (or a combination of kernels) and or different graphs (or a combination of different graphs). Many other versions of SSVDD can be derived and implemented based on introducing new regularization terms. Also, the existing and the derived versions of SSVDD can be infused with the geometric information in data to get many different graph-based solutions to the problem at hand. Furthermore, all subspace-based models (SSVDD, GESSVDD, and their variants) can be implemented using a newly proposed solution based on Newton's method [60]. Moreover, the credit card fraud detection system can be adapted to the Multi-modal Subspace Support Vector Data Description [61] settings for future reference.

References

[1] Masoumeh Zareapoor, KR Seeja, and M Afshar Alam. “Analysis on credit card fraud detection techniques: based on certain design criteria”. In: *International journal of computer applications* 52.3 (2012).

[2] Sam Maes et al. “Credit card fraud detection using Bayesian and neural networks”. In: *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*. Vol. 261. 2002, p. 270.

[3] Federal Trade Commission. *Identity Theft Reports*. Available online: https://public.tableau.com/app/profile/federal.trade.commission/viz/Identity TheftReports/TheftTypesOverTime (accessed on 10 August 2023). 2023.

[4] Katherine J Barker, Jackie D’amato, and Paul Sheridon. “Credit card fraud: awareness and prevention”. In: *Journal of financial crime* (2008).

[5] Tej Paul Bhatla, Vikram Prabhu, and Amit Dua. “Understanding credit card frauds”. In: *Cards business review* 1.6 (2003), pp. 1–15.

[6] Akarshita Shankar, Ramesh Shetty, and B Nath. “A review on phishing attacks”. In: *International Journal of Applied Engineering Research* 14.9 (2019), pp. 2171–2175.

[7] Mark Arend. “New card fraud weapons emerge”. In: *American Bankers Association. ABA Banking Journal* 85.9 (1993), p. 91.

[8] John O. Awoyemi, Adebayo O. Adetunmbi, and Samuel A. Oluwadare. “Credit card fraud detection using machine learning techniques: A comparative analysis”. In: *2017 International Conference on Computing Networking and Informatics*. 2017, pp. 1–9.

[9] Asha RB and Suresh Kumar KR. “Credit card fraud detection using artificial neural network”. In: *Global Transitions Proceedings* 2.1 (2021). 1st International Conference on Advances in Information, Computing and Trends in Data Engineering (AICDE - 2020), pp. 35–41. ISSN: 2666-285X.

[10] Sumit Kumar Ojha and Padmapriya Govindhan. “Analysis of credit card fraud detection using novel ANN algorithm to improve the rate of accuracy based on assorted transactional behaviour in comparison with logistic regression”. In: *American Institute of Physics Conference Series*. Vol. 2655. 1. 2023, p. 020081.

[11] Raghavendra Patidar, Lokesh Sharma, et al. “Credit card fraud detection using neural network”. In: *International Journal of Soft Computing and Engineering* 1.32-38 (2011).

[12] Yusuf G Şahin and Ekrem Duman. “Detecting credit card fraud by decision trees and support vector machines”. In: (2011).

[13] Prajal Save et al. "A novel idea for credit card fraud detection using decision tree". In: *International Journal of Computer Applications* 161.13 (2017).

[14] MR Dileep, AV Navaneeth, and M Abhishek. "A novel approach for credit card fraud detection using decision tree and random forest algorithms". In: *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*. IEEE. 2021, pp. 1025–1028.

[15] Tom Fawcett and Foster Provost. "Adaptive fraud detection". In: *Data mining and knowledge discovery* 1.3 (1997), pp. 291–316.

[16] Vladimir Zaslavsky and Anna Strizhak. "Credit card fraud detection using self-organizing maps". In: *Information and Security* 18 (2006), p. 48.

[17] Harsh Harwani et al. "Credit card fraud detection technique using hybrid approach: An amalgamation of self organizing maps and neural networks". In: *International Research Journal of Engineering and Technology* 7.2020 (2020).

[18] E Saraswathi et al. "Credit card fraud prediction and detection using artificial neural network and self-organizing maps". In: *2019 3rd International Conference on Computing Methodologies and Communication*. IEEE. 2019, pp. 1124–1128.

[19] Johannes Jurgovsky et al. "Sequence classification for credit-card fraud detection". In: *Expert Systems with Applications* 100 (2018), pp. 234–245.

[20] Ankit Vartak, Chinmay D Patil, and Chinmay K Patil. "Hidden Markov Model for credit card fraud detection". In: *International Journal of Computer Science and Information Technologies* 5.6 (2014), pp. 7446–7451.

[21] Abhinav Srivastava et al. "Credit card fraud detection using hidden Markov model". In: *IEEE Transactions on dependable and secure computing* 5.1 (2008), pp. 37–48.

[22] Tanmay Kumar Behera and Suvasini 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*. IEEE. 2015, pp. 494–499.

[23] Arti Jain, Archana Purwar, and Divakar Yadav. "Credit Card Fraud Detection Using K-Means and Fuzzy C-Means". In: *Handbook of Research on Innovations and Applications of AI, IoT, and Cognitive Technologies*. IGI Global, 2021, pp. 216–240.

[24] N Geetha and G Dheepa. "A Hybrid Deep Learning And Modified Butterfly Optimization Based Feature Selection For Transaction Credit Card Fraud Detection". In: *Journal of Positive School Psychology* 6.7 (2022), pp. 5328–5345.

[25] Kasarapu Ramani et al. “Gradient boosting techniques for credit card fraud detection”. In: *Journal Of Algebraic Statistics* 13.3 (2022), pp. 553–558.

[26] Altyeb Altaher Taha and Sharaf Jameel Malebary. “An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine”. In: *IEEE Access* 8 (2020), pp. 25579–25587.

[27] Fengjun Zhang et al. “GMM-based undersampling and its application for credit card fraud detection”. In: *2019 International Joint Conference on Neural Networks*. IEEE. 2019, pp. 1–8.

[28] Dilip Singh Sisodia, Nerella Keerthana Reddy, and Shivangi Bhandari. “Performance evaluation of class balancing techniques for credit card fraud detection”. In: *2017 IEEE International Conference on power, control, signals and instrumentation engineering*. IEEE. 2017, pp. 2747–2752.

[29] Jalal Ahammad, Nazia Hossain, and Mohammad Shafiul Alam. “Credit card fraud detection using data pre-processing on imbalanced data-Both oversampling and undersampling”. In: *Proceedings of the International Conference on Computing Advancements*. 2020, pp. 1–4.

[30] Mohammad Sabokrou et al. “Deep End-to-End One-Class Classifier”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.2 (2021), pp. 675–684.

[31] Poojan Oza and Vishal M. Patel. “One-Class Convolutional Neural Network”. In: *IEEE Signal Processing Letters* 26.2 (2019), pp. 277–281.

[32] Mohammad Sabokrou et al. “Adversarially learned one-class classifier for novelty detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3379–3388.

[33] Sachin Goyal et al. “DROCC: Deep robust one-class classification”. In: *International conference on machine learning*. PMLR. 2020, pp. 3711–3721.

[34] Jorge Rodríguez-Ruiz et al. “A one-class classification approach for bot detection on Twitter”. In: *Computers & Security* 91 (2020), p. 101715. ISSN: 0167-4048.

[35] Francesca Fortunato, Laura Anderlucci, and Angela Montanari. “One-Class Classification with Application to Forensic Analysis”. In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 69.5 (Aug. 2020), pp. 1227–1249. ISSN: 0035-9254.

[36] Janaina Mourão-Miranda et al. “Patient classification as an outlier detection problem: An application of the One-Class Support Vector Machine”. In: *NeuroImage* 58.3 (2011), pp. 793–804. ISSN: 1053-8119.

[37] Bogusław Cyganek. "One-class support vector ensembles for image segmentation and classification". In: *Journal of Mathematical Imaging and Vision* 42 (2012), pp. 103–117.

[38] Sertac Kilickaya et al. "Hyperspectral Image Analysis with Subspace Learning-based One-Class Classification". In: *2023 Photonics & Electromagnetics Research Symposium*. 2023, pp. 953–959.

[39] Aysen Degerli et al. "Early Myocardial Infarction Detection with One-Class Classification over Multi-view Echocardiography". In: *2022 Computing in Cardiology*. Vol. 498. IEEE. 2022, pp. 1–4.

[40] Fahad Sohrab and Jenni Raitoharju. "Boosting rare benthic macroinvertebrates taxa identification with one-class classification". In: *2020 IEEE Symposium Series on Computational Intelligence*. IEEE. 2020, pp. 928–933.

[41] Firas Laakom et al. "Convolutional autoencoder-based multimodal one-class classification". In: *arXiv preprint arXiv:2309.14090* (2023).

[42] Jake Guidry et al. "One-Class Classification for Intrusion Detection on Vehicular Networks". In: *arXiv preprint arXiv:2309.14134* (2023).

[43] Fahad Sohrab. *Railway vehicle detection from audio recordings using one-class classification*. Master's Thesis. 2016.

[44] David Martinus Johannes Tax. "One-class classification: Concept learning in the absence of counter-examples." PhD thesis. Technische Universiteit Delft (The Netherlands), 2001.

[45] Emanuel Parzen. "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[46] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[47] Teuvo Kohonen. *Self-organizing maps*. Vol. 30. Springer Science & Business Media, 2012.

[48] Alexander Senf, Xue-wen Chen, and Anne Zhang. "Comparison of one-class SVM and two-class SVM for fold recognition". In: *Neural Information Processing: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006. Proceedings, Part II 13*. Springer. 2006, pp. 140–149.

[49] J. Zhou et al. "Extraction of Brain Tumor from MR Images Using One-Class Support Vector Machine". In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. 2005, pp. 6411–6414.

[50] David M.J. Tax and Robert P.W. Duin. "Support Vector Data Description". In: *Machine Learning* 54.1 (Jan. 2004), pp. 45–66. ISSN: 1573-0565.

[51] Fahad Sohrab et al. “Subspace support vector data description”. In: *2018 24th International Conference on Pattern Recognition*. IEEE. 2018, pp. 722–727.

[52] Yahya Forghani et al. “Support Vector Data Description by using hyper-ellipse instead of hyper-sphere”. In: *2011 1st International eConference on Computer and Knowledge Engineering*. 2011, pp. 22–27.

[53] Vasileios Mygdalis et al. “Graph embedded one-class classifiers for media data classification”. In: *Pattern Recognition* 60 (2016), pp. 585–595.

[54] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. “A Generalized Representer Theorem”. In: *Computational Learning Theory*. Ed. by David Helmbold and Bob Williamson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 416–426.

[55] Fahad Sohrab et al. “Graph-embedded subspace support vector data description”. In: *Pattern Recognition* 133 (2023), p. 108999. ISSN: 0031-3203.

[56] Nojun Kwak. “Nonlinear projection trick in kernel methods: An alternative to the kernel trick”. In: *IEEE transactions on neural networks and learning systems* 24.12 (2013), pp. 2113–2119.

[57] Andrea Dal Pozzolo et al. “Learned lessons in credit card fraud detection from a practitioner perspective”. In: *Expert systems with applications* 41.10 (2014), pp. 4915–4928.

[58] Edgar Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. “PaySim: A financial mobile money simulator for fraud detection”. In: *28th European Modeling and Simulation Symposium, EMSS, Larnaca*. Dime University of Genoa. 2016, pp. 249–255.

[59] Fahad Sohrab. “Subspace Support Vector Data Description and Extensions”. PhD thesis. 2022.

[60] Fahad Sohrab, Firas Laakom, and Moncef Gabbouj. “Newton Method-based Subspace Support Vector Data Description”. In: *arXiv preprint arXiv:2309.13960* (2023).

[61] Fahad Sohrab et al. “Multimodal subspace support vector data description”. In: *Pattern Recognition* 110 (2021), p. 107648. ISSN: 0031-3203.

A Appendix

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.855	0.855	0.959	1.000	0.922
GESSVDD-kNN-G-max	0.987	0.988	0.730	1.000	0.994
GESSVDD-kNN-E-min	0.993	0.994	0.412	0.999	0.996
GESSVDD-kNN-E-max	0.995	0.996	0.473	0.999	0.997
GESSVDD-kNN-S-min	0.998	1.000	0.088	0.998	0.999
GESSVDD-kNN-S-max	0.998	1.000	0.088	0.998	0.999
GESSVDD-PCA-G-min	0.196	0.195	0.959	1.000	0.326
GESSVDD-PCA-G-max	0.450	0.449	0.926	1.000	0.619
GESSVDD-PCA-E-min	0.993	0.993	0.608	0.999	0.996
GESSVDD-PCA-E-max	0.996	0.996	0.520	0.999	0.998
GESSVDD-PCA-S-min	0.998	1.000	0.027	0.998	0.999
GESSVDD-PCA-S-max	0.998	1.000	0.007	0.998	0.999
GESSVDD-I-G-min	0.996	0.998	0.014	0.998	0.998
GESSVDD-I-G-max	0.998	1.000	0.000	0.998	0.999
GESSVDD-I-E-min	0.997	0.998	0.527	0.999	0.998
GESSVDD-I-E-max	0.997	0.999	0.243	0.999	0.999
GESSVDD-I-S-min	0.998	1.000	0.027	0.998	0.999
GESSVDD-I-S-max	0.998	1.000	0.007	0.998	0.999
SSVDD-Ψ_0-min	0.216	0.215	0.892	0.999	0.354
SSVDD-Ψ_0-max	0.172	0.170	0.899	0.999	0.291
SSVDD-Ψ_1-min	0.998	1.000	0.000	0.998	0.999
SSVDD-Ψ_1-max	0.998	1.000	0.000	0.998	0.999
SSVDD-Ψ_2-min	0.049	0.048	0.689	0.989	0.092
SSVDD-Ψ_2-max	0.049	0.048	0.689	0.989	0.092
SSVDD-Ψ_3-min	0.049	0.048	0.689	0.989	0.092
SSVDD-Ψ_3-max	0.049	0.048	0.689	0.989	0.092
OCSVM	0.919	0.920	0.216	0.999	0.958
SVDD	0.050	0.048	0.811	0.993	0.092
ESVDD	0.002	0.000	1.000	0.000	0.000

Table A.1 *Results for the linear models using dataset-1.*

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.998	1.000	0.108	0.998	0.999
GESSVDD-kNN-G-max	0.007	0.005	0.905	0.970	0.011
GESSVDD-kNN-E-min	0.002	0.000	0.919	0.294	0.000
GESSVDD-kNN-E-max	0.002	0.000	1.000	0.000	0.000
GESSVDD-kNN-S-min	0.984	0.985	0.277	0.999	0.992
GESSVDD-kNN-S-max	0.951	0.952	0.318	0.999	0.975
GESSVDD-PCA-G-min	0.002	0.000	1.000	1.000	0.000
GESSVDD-PCA-G-max	0.950	0.952	0.209	0.999	0.975
GESSVDD-PCA-E-min	0.049	0.048	0.676	0.988	0.092
GESSVDD-PCA-E-max	0.998	1.000	0.000	0.998	0.999
GESSVDD-PCA-S-min	0.951	0.953	0.182	0.999	0.975
GESSVDD-PCA-S-max	0.952	0.954	0.122	0.998	0.976
GESSVDD-I-G-min	0.002	0.000	0.980	0.250	0.000
GESSVDD-I-G-max	0.950	0.952	0.257	0.999	0.975
GESSVDD-I-E-min	0.002	0.000	0.905	0.222	0.000
GESSVDD-I-E-max	0.998	1.000	0.000	0.998	0.999
GESSVDD-I-S-min	0.025	0.024	0.966	0.998	0.046
GESSVDD-I-S-max	0.998	1.000	0.000	0.998	0.999
SSVDD-Ψ_0-min	0.049	0.048	0.966	0.999	0.091
SSVDD-Ψ_0-max	0.050	0.048	0.899	0.996	0.092
SSVDD-Ψ_1-min	0.050	0.048	0.777	0.992	0.092
SSVDD-Ψ_1-max	0.032	0.031	0.980	0.999	0.059
SSVDD-Ψ_2-min	0.050	0.048	0.770	0.992	0.092
SSVDD-Ψ_2-max	0.998	1.000	0.149	0.999	0.999
SSVDD-Ψ_3-min	0.050	0.048	0.804	0.993	0.092
SSVDD-Ψ_3-max	0.050	0.048	0.899	0.996	0.092
OCSVM	0.019	0.017	1.000	1.000	0.034
SVDD	0.050	0.048	0.811	0.993	0.092
ESVDD	0.048	0.047	0.980	0.999	0.089
GEOCSVM	0.036	0.034	0.980	0.999	0.066
GESVDD	0.048	0.047	0.993	1.000	0.089

Table A.2 *Results for the non-linear models using dataset-1.*

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.402	0.359	0.846	0.961	0.523
GESSVDD-kNN-G-max	0.636	0.654	0.448	0.925	0.766
GESSVDD-kNN-E-min	0.259	0.242	0.440	0.819	0.373
GESSVDD-kNN-E-max	0.820	0.844	0.567	0.953	0.896
GESSVDD-kNN-S-min	0.527	0.538	0.414	0.906	0.675
GESSVDD-kNN-S-max	0.128	0.047	0.974	0.949	0.089
GESSVDD-PCA-G-min	0.915	0.995	0.080	0.919	0.955
GESSVDD-PCA-G-max	0.911	0.995	0.037	0.915	0.953
GESSVDD-PCA-E-min	0.910	0.994	0.032	0.915	0.953
GESSVDD-PCA-E-max	0.913	0.997	0.035	0.915	0.954
GESSVDD-PCA-S-min	0.912	0.997	0.032	0.915	0.954
GESSVDD-PCA-S-max	0.912	0.996	0.037	0.915	0.954
GESSVDD-I-G-min	0.757	0.824	0.053	0.901	0.861
GESSVDD-I-G-max	0.911	0.997	0.015	0.914	0.953
GESSVDD-I-E-min	0.910	0.995	0.029	0.914	0.953
GESSVDD-I-E-max	0.913	0.997	0.035	0.915	0.954
GESSVDD-I-S-min	0.907	0.994	0.008	0.913	0.951
GESSVDD-I-S-max	0.913	0.996	0.042	0.916	0.954
SSVDD-Ψ_0-min	0.913	0.996	0.042	0.916	0.954
SSVDD-Ψ_0-max	0.912	0.997	0.026	0.914	0.954
SSVDD-Ψ_1-min	0.913	0.996	0.042	0.916	0.954
SSVDD-Ψ_1-max	0.912	0.997	0.026	0.914	0.954
SSVDD-Ψ_2-min	0.913	0.996	0.042	0.916	0.954
SSVDD-Ψ_2-max	0.912	0.997	0.026	0.914	0.954
SSVDD-Ψ_3-min	0.913	0.996	0.042	0.916	0.954
SSVDD-Ψ_3-max	0.912	0.997	0.026	0.914	0.954
OCSVM	0.463	0.440	0.712	0.941	0.599
SVDD	0.912	0.997	0.034	0.915	0.954
ESVDD	0.912	0.996	0.035	0.915	0.954

Table A.3 *Results for the linear models using dataset-2.*

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.914	0.994	0.070	0.918	0.955
GESSVDD-kNN-G-max	0.179	0.111	0.885	0.910	0.199
GESSVDD-kNN-E-min	0.226	0.185	0.647	0.846	0.304
GESSVDD-kNN-E-max	0.226	0.185	0.647	0.846	0.304
GESSVDD-kNN-S-min	0.311	0.297	0.447	0.849	0.441
GESSVDD-kNN-S-max	0.311	0.297	0.447	0.849	0.441
GESSVDD-PCA-G-min	0.912	0.999	0.012	0.913	0.954
GESSVDD-PCA-G-max	0.087	0.000	0.997	0.542	0.001
GESSVDD-PCA-E-min	0.430	0.384	0.908	0.978	0.552
GESSVDD-PCA-E-max	0.233	0.214	0.422	0.795	0.338
GESSVDD-PCA-S-min	0.508	0.478	0.822	0.966	0.640
GESSVDD-PCA-S-max	0.508	0.478	0.822	0.966	0.640
GESSVDD-I-G-min	0.087	0.004	0.956	0.474	0.008
GESSVDD-I-G-max	0.088	0.002	0.984	0.538	0.004
GESSVDD-I-E-min	0.250	0.217	0.596	0.849	0.346
GESSVDD-I-E-max	0.250	0.217	0.596	0.849	0.346
GESSVDD-I-S-min	0.151	0.081	0.883	0.879	0.149
GESSVDD-I-S-max	0.151	0.081	0.883	0.879	0.149
SSVDD-Ψ_0-min	0.228	0.162	0.913	0.951	0.277
SSVDD-Ψ_0-max	0.241	0.192	0.753	0.890	0.315
SSVDD-Ψ_1-min	0.228	0.162	0.913	0.951	0.277
SSVDD-Ψ_1-max	0.241	0.192	0.753	0.890	0.315
SSVDD-Ψ_2-min	0.228	0.162	0.913	0.951	0.277
SSVDD-Ψ_2-max	0.241	0.192	0.753	0.890	0.315
SSVDD-Ψ_3-min	0.228	0.162	0.913	0.951	0.277
SSVDD-Ψ_3-max	0.241	0.192	0.753	0.890	0.315
OCSVM	0.447	0.413	0.797	0.955	0.577
SVDD	0.069	0.007	0.713	0.214	0.014
ESVDD	0.085	0.002	0.952	0.348	0.005
GEOCSVM	0.891	0.881	0.997	1.000	0.936
GESVDD	0.870	0.860	0.969	0.997	0.923

Table A.4 *Results for the non-linear models using dataset-2.*

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.996	0.997	0.364	0.999	0.998
GESSVDD-kNN-G-max	0.996	0.997	0.364	0.999	0.998
GESSVDD-kNN-E-min	0.722	0.722	0.673	1.000	0.838
GESSVDD-kNN-E-max	0.996	0.997	0.364	0.999	0.998
GESSVDD-kNN-S-min	0.547	0.546	0.971	1.000	0.707
GESSVDD-kNN-S-max	0.420	0.419	0.971	1.000	0.591
GESSVDD-PCA-G-min	0.996	0.996	0.356	0.999	0.998
GESSVDD-PCA-G-max	0.996	0.997	0.367	0.999	0.998
GESSVDD-PCA-E-min	0.996	0.996	0.391	0.999	0.998
GESSVDD-PCA-E-max	0.996	0.996	0.356	0.999	0.998
GESSVDD-PCA-S-min	0.996	0.996	0.356	0.999	0.998
GESSVDD-PCA-S-max	0.996	0.996	0.356	0.999	0.998
GESSVDD-I-G-min	0.996	0.996	0.356	0.999	0.998
GESSVDD-I-G-max	0.996	0.996	0.356	0.999	0.998
GESSVDD-I-E-min	0.996	0.996	0.391	0.999	0.998
GESSVDD-I-E-max	0.996	0.996	0.356	0.999	0.998
GESSVDD-I-S-min	0.996	0.996	0.356	0.999	0.998
GESSVDD-I-S-max	0.997	0.997	0.373	0.999	0.998
SSVDD-Ψ_0-min	0.996	0.997	0.166	0.999	0.998
SSVDD-Ψ_0-max	0.996	0.997	0.163	0.999	0.998
SSVDD-Ψ_1-min	0.996	0.997	0.166	0.999	0.998
SSVDD-Ψ_1-max	0.996	0.997	0.163	0.999	0.998
SSVDD-Ψ_2-min	0.996	0.997	0.166	0.999	0.998
SSVDD-Ψ_2-max	0.996	0.997	0.163	0.999	0.998
SSVDD-Ψ_3-min	0.996	0.997	0.166	0.999	0.998
SSVDD-Ψ_3-max	0.996	0.997	0.163	0.999	0.998
OCSVM	0.052	0.051	1.000	1.000	0.098
SVDD	0.996	0.997	0.163	0.999	0.998
ESVDD	0.996	0.996	0.356	0.999	0.998

Table A.5 *Results for the linear models using dataset-3.*

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.545	0.545	0.598	0.999	0.705
GESSVDD-kNN-G-max	0.622	0.623	0.501	0.999	0.767
GESSVDD-kNN-E-min	0.850	0.850	0.391	0.999	0.919
GESSVDD-kNN-E-max	0.850	0.850	0.391	0.999	0.919
GESSVDD-kNN-S-min	0.841	0.841	0.399	0.999	0.913
GESSVDD-kNN-S-max	0.841	0.841	0.399	0.999	0.913
GESSVDD-PCA-G-min	0.896	0.897	0.391	0.999	0.945
GESSVDD-PCA-G-max	0.896	0.897	0.391	0.999	0.945
GESSVDD-PCA-E-min	0.942	0.942	0.318	0.999	0.970
GESSVDD-PCA-E-max	0.946	0.946	0.300	0.999	0.972
GESSVDD-PCA-S-min	0.070	0.069	0.746	0.996	0.129
GESSVDD-PCA-S-max	0.919	0.920	0.364	0.999	0.958
GESSVDD-I-G-min	0.911	0.911	0.338	0.999	0.953
GESSVDD-I-G-max	0.961	0.961	0.376	0.999	0.980
GESSVDD-I-E-min	0.940	0.941	0.329	0.999	0.969
GESSVDD-I-E-max	0.952	0.952	0.259	0.999	0.975
GESSVDD-I-S-min	0.940	0.941	0.329	0.999	0.969
GESSVDD-I-S-max	0.940	0.941	0.329	0.999	0.969
SSVDD-Ψ_0-min	0.009	0.008	0.997	1.000	0.016
SSVDD-Ψ_0-max	0.094	0.094	0.638	0.996	0.171
SSVDD-Ψ_1-min	0.009	0.008	0.997	1.000	0.016
SSVDD-Ψ_1-max	0.094	0.094	0.638	0.996	0.171
SSVDD-Ψ_2-min	0.009	0.008	0.997	1.000	0.016
SSVDD-Ψ_2-max	0.094	0.094	0.638	0.996	0.171
SSVDD-Ψ_3-min	0.009	0.008	0.997	1.000	0.016
SSVDD-Ψ_3-max	0.094	0.094	0.638	0.996	0.171
OCSVM	0.011	0.010	1.000	1.000	0.021
SVDD	0.003	0.002	0.886	0.934	0.003
ESVDD	0.896	0.897	0.391	0.999	0.945
GEOCSVM	0.862	0.862	0.726	1.000	0.926
GESVDD	0.789	0.789	0.446	0.999	0.882

Table A.6 *Results for the non-linear models using dataset-3.*

Model	**Accuracy**	**tpr**	**tnr**	**Precision**	**F-measure**
GESSVDD-kNN-G-min	0.600	0.538	0.770	0.866	0.664
GESSVDD-kNN-G-max	0.725	0.761	0.628	0.850	0.803
GESSVDD-kNN-E-min	0.647	0.694	0.515	0.798	0.743
GESSVDD-kNN-E-max	0.573	0.637	0.394	0.744	0.687
GESSVDD-kNN-S-min	0.496	0.568	0.297	0.690	0.623
GESSVDD-kNN-S-max	0.653	0.786	0.284	0.752	0.769
GESSVDD-PCA-G-min	0.730	0.992	0.006	0.734	0.844
GESSVDD-PCA-G-max	0.731	0.994	0.005	0.734	0.844
GESSVDD-PCA-E-min	0.732	0.996	0.003	0.734	0.845
GESSVDD-PCA-E-max	0.732	0.995	0.007	0.735	0.845
GESSVDD-PCA-S-min	0.745	0.994	0.058	0.745	0.852
GESSVDD-PCA-S-max	0.741	0.987	0.060	0.744	0.848
GESSVDD-I-G-min	0.584	0.714	0.225	0.718	0.716
GESSVDD-I-G-max	0.638	0.731	0.380	0.765	0.748
GESSVDD-I-E-min	0.671	0.831	0.227	0.748	0.787
GESSVDD-I-E-max	0.540	0.630	0.292	0.711	0.668
GESSVDD-I-S-min	0.731	0.993	0.007	0.734	0.844
GESSVDD-I-S-max	0.730	0.989	0.013	0.735	0.843
SSVDD-Ψ_0-min	0.736	0.994	0.023	0.738	0.847
SSVDD-Ψ_0-max	0.734	0.994	0.015	0.736	0.846
SSVDD-Ψ_1-min	0.736	0.994	0.023	0.738	0.847
SSVDD-Ψ_1-max	0.734	0.994	0.015	0.736	0.846
SSVDD-Ψ_2-min	0.736	0.994	0.023	0.738	0.847
SSVDD-Ψ_2-max	0.734	0.994	0.015	0.736	0.846
SSVDD-Ψ_3-min	0.736	0.994	0.023	0.738	0.847
SSVDD-Ψ_3-max	0.734	0.994	0.015	0.736	0.846
OCSVM	0.319	0.257	0.491	0.582	0.356
SVDD	0.740	0.991	0.047	0.742	0.848
ESVDD	0.734	1.000	0.001	0.734	0.847

Table A.7 *Results for the linear models using dataset-4.*

Model	Accuracy	tpr	tnr	Precision	F-measure
GESSVDD-kNN-G-min	0.252	0.196	0.407	0.478	0.278
GESSVDD-kNN-G-max	0.344	0.344	0.346	0.592	0.435
GESSVDD-kNN-E-min	0.308	0.291	0.355	0.555	0.381
GESSVDD-kNN-E-max	0.308	0.291	0.355	0.555	0.381
GESSVDD-kNN-S-min	0.267	0.003	0.998	0.800	0.005
GESSVDD-kNN-S-max	0.267	0.004	0.991	0.571	0.009
GESSVDD-PCA-G-min	0.640	0.819	0.147	0.726	0.770
GESSVDD-PCA-G-max	0.326	0.316	0.355	0.575	0.407
GESSVDD-PCA-E-min	0.270	0.013	0.979	0.632	0.026
GESSVDD-PCA-E-max	0.270	0.013	0.979	0.634	0.026
GESSVDD-PCA-S-min	0.254	0.016	0.910	0.332	0.031
GESSVDD-PCA-S-max	0.254	0.016	0.910	0.332	0.031
GESSVDD-I-G-min	0.287	0.038	0.974	0.803	0.072
GESSVDD-I-G-max	0.299	0.264	0.395	0.547	0.356
GESSVDD-I-E-min	0.266	0.005	0.990	0.553	0.009
GESSVDD-I-E-max	0.262	0.019	0.936	0.447	0.036
GESSVDD-I-S-min	0.492	0.516	0.426	0.713	0.599
GESSVDD-I-S-max	0.255	0.018	0.909	0.358	0.035
SSVDD-Ψ_0-min	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_0-max	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_1-min	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_1-max	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_2-min	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_2-max	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_3-min	0.602	0.743	0.213	0.723	0.733
SSVDD-Ψ_3-max	0.602	0.743	0.213	0.723	0.733
OCSVM	0.741	0.814	0.538	0.830	0.822
SVDD	0.212	0.049	0.662	0.284	0.083
ESVDD	0.230	0.014	0.827	0.185	0.026
GEOCSVM	0.757	0.800	0.638	0.859	0.828
GESVDD	0.740	0.787	0.612	0.849	0.817

Table A.8 *Results for the non-linear models using dataset-4.*