UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Physics of Data

Final Dissertation

A fast classifier-based approach to credit card fraud detection

Thesis supervisor

Dr. Marco Letizia

Thesis co-supervisor

Prof. Marco Zanetti

Candidate

Alireza Molla Ali Hosseini

Academic Year 2022/2023

Abstract

This thesis aims at addressing the problem of anomaly detection in the context of credit card fraud detection with machine learning. Specifically, the goal is to apply a new approach to two-sample testing based on classifiers recently developed for new physic searches in high-energy physics. This strategy allows one to compare batches of incoming data with a control sample of standard transactions in a statistically sound way without prior knowledge of the type of fraudulent activity. The learning algorithm at the basis of this approach is a modern implementation of kernel methods that allows for fast online training and high flexibility. This work is the first attempt to export this method to a real-world use case outside the domain of particle physics.

Contents

Chapter 1

Introduction

In recent years, the realm of data-driven decision-making has been irrevocably transformed by the convergence of two groundbreaking trends: the rapid advancement of machine learning techniques and the burgeoning era of big data. The symbiotic relationship between these two phenomena has propelled industries, scientific disciplines, and everyday life into an unprecedented era of innovation and insight generation.

The Era of Machine Learning and Big Data

In the midst of this digital age, the exponential surge in data, coupled with computational prowess, has enabled groundbreaking potential for businesses, researchers, and policymakers to uncover transformative insights. Within this landscape, machine learning, a subset of artificial intelligence, stands as the linchpin, unraveling hidden value within the data deluge. By harnessing algorithmic power, machine learning delves into intricate patterns and relationships that traditional methods couldn't fathom. This synergy between machine learning and big data has spurred revolutionary applications, from healthcare advancements and financial insights to personalized marketing strategies. Examples include precision spam filtering, self-driving cars navigating complex environments, intelligent chatbots understanding and responding to human language, and recommendation systems shaping personalized content delivery. These instances underscore the innovation that's been catalyzed, reshaping industries and driving progress beyond previously perceived boundaries. The computational muscle of Graphics Processing Units (GPUs) has further fueled these strides, particularly in deep learning,

accelerating the training of complex models that underpin transformative advances across society.

The Expansion of Machine Learning in Sciences

One domain where machine learning is leaving an indelible mark is the realm of scientific exploration. In an era where the frontiers of knowledge are expanding exponentially, the availability of data from various scientific disciplines has surged. Machine learning has emerged as an indispensable tool, breathing life into these datasets and empowering scientists to glean meaningful insights from the chaos. Beyond traditional hypothesis-driven methods, machine learning's ability to unveil hidden patterns, classify intricate data, and forecast future trends has revolutionized scientific inquiry.

Machine Learning's ability to handle complex, multidimensional data has positioned it as a pivotal instrument for precision data analysis. Machine learning offers invaluable benefits to the physical and natural sciences by enabling data-driven insights, pattern recognition, and predictive modeling from vast and complex datasets. It aids in accelerating scientific discovery, from drug design and climate forecasting to medical imaging and particle physics. Simultaneously, the domain-specific challenges in these sciences contribute to the advancement of artificial intelligence. The intricacies of scientific problems inspire the creation of specialized algorithms, while the generation of synthetic data for simulations enhances artificial intelligence training. Collaborations between domain experts and artificial intelligence researchers lead to novel methodologies, benchmarks, and evaluation criteria. This reciprocal relationship enriches both fields, fostering innovation that addresses unique challenges and fuels broader technological progress.

Recent Applications to High Energy Physics

Based on experimental observations and compelling conceptual arguments, it becomes apparent that our existing understanding of fundamental physics is incomplete. The Standard Model of Particle Physics, the theory that codifies our understanding of fundamental particles and their interactions, has demonstrated remarkable accuracy in predicting and explaining almost all experimental results we have collected so far. Nonetheless, the limitations of the Standard Model are evident as it leaves a number of inquiries unanswered, spanning from the origin of the electroweak scale to the enigma of neutrino

masses, along with the intricate flavor patterns within quarks, leptons, and neutrinos. These considerations indicate the existence of underlying fundamental laws of nature that remain concealed and await revelation.

Unveiling these elusive laws necessitates a thorough examination of experimental data in pursuit of phenomena that deviate from the predictions of the Standard Model. The prevailing approach involves scrutinizing data to identify distinct signatures of models beyond the Standard Model, tackling each proposal individually. Each analysis is meticulously tailored to detect the unique characteristics associated with the specific new physics hypothesis under consideration. However, this methodology often falls short of detecting inconsistencies that stray from the predefined scenarios. Consequently, a notable endeavor is underway to establish analytical strategies that remain impartial to the potential nature of new physics, offering a valuable complement to the aforementioned model-dependent methods.

The aspiration is to develop approaches capable of discerning deviations from the expected outcomes of a given reference model, even when these deviations lack precise definition. This analytical paradigm, also known as a model-independent approach, aims to exhibit sensitivity to a wide spectrum of potential deviations, transcending the limitations posed by conventional approaches.

Concurrently, the application of machine learning techniques in scientific research heralds a paradigm shift in the formulation, testing, and validation of hypotheses. Through the automated analysis of extensive and intricate data, researchers can augment their ability to identify subtle nuances and relationships, fostering the generation of hypotheses that surpass traditional boundaries. Additionally, machine learning expedites the process of hypothesis testing by minimizing manual intervention, enabling researchers to swiftly iterate through complex hypotheses and refine their inquiries. This innovative approach accelerates the pace of discovery and facilitates the exploration of intricate interactions within natural systems, thereby empowering scientists to unlock new realms of knowledge.

Consequently, a concerted effort has been directed towards harnessing machine learning as a solution for model-independent searches in the realm of high-energy physics. In line with this pursuit, we will examine a specific recent approach known as *New Physics Learning Machine* ([1, 2]). This endeavor arises from the necessity to enhance our capability to detect indications of new physics without presupposing assumptions about their forms, presenting a fresh perspective on the exploration of the universe's fundamen-

tal fabric.

Beyond Particle Physics

The deployment of tools and techniques developed in high energy physics research in other societal domains is a fascinating prospective. The model employed for the new physics searches has indeed exhibited potential for application beyond the confines of the physics domain. Building upon this notion, we have endeavored to leverage these models in the context of fraud detection, which forms a specific subset of the broader anomaly detection field. However, it is essential to emphasize that while the prevalent approach in anomaly detection often pertains to outlier identification, our model diverges from this interpretation.Instead, we concentrate on discerning collective trends in the data that deviate from the expected norm.

In conventional anomaly detection practices, the emphasis is often on identifying data points that deviate significantly from the norm, reflecting the presence of outliers. While this paradigm has proven effective in numerous contexts, our approach ventures beyond the mere identification of outliers. We consider a different approach that involves identifying deviations from the expected statistical distributions and patterns inherent in a given dataset. By embracing this methodology, we aim to not only pinpoint outliers but also uncover intricate statistical anomalies that might otherwise remain concealed under conventional outlier-based approaches.

Outline of the Thesis

In this thesis, Chapter 2 will provide an exposition on machine learning, elucidating the particular methods that will be employed in our research. Moving forward to Chapter 3, we will delve into the concept of the physics learning machine model, which forms the bedrock of the ideas presented in this study. In the subsequent Chapter 4, we will elaborate on the formulation of our method tailored to the specific dataset within the domain of fraud detection. This will encompass the introduction of the dataset, the methodology we have developed, and a comprehensive presentation of our model's outcomes and results. Finally, in Chapter 5 we will summarize our findings and discuss future developments.

Chapter 2

Machine Learning

Machine learning, a subfield of artificial intelligence (AI), focuses on developing algorithms and models that enable computers to learn from data and make predictions or decisions without explicit programming. The learning process is data-driven, as the algorithm or model learns from a set of training data containing input examples and corresponding outputs or labels. By analyzing and processing this labeled data, the machine learning model acquires patterns and relationships, enabling it to make accurate predictions or decisions on new, unseen data. This generalization is achieved through iterative adjustments of the model's internal parameters based on the disparities between its predicted outputs and the true labels in the training data, using optimization techniques. This introductory part is mostly based on [3].

Machine learning can be categorized into three main types. The first type is *supervised learning*, which aims to learn a relationship between input and output data based on a labeled training set of input-output pairs, $D = \{(x_i, y_i)\}_{i=1}^N$. The inputs x_i can be simple numerical vectors representing attributes like height and weight, or they can be structured objects such as images, sentences, emails, time series, molecular shapes, or graphs. The outputs y_i can be categorical variables from a finite set (e.g., male or female) denoted as $y_i \in \{1, \ldots, C\}$, or they can be real-valued scalar variables (e.g., income level). *Classification* falls within the realm of supervised learning and involves tackling the task of forecasting categorical outcomes. On the other hand, *regression*, which is also a subset of supervised learning, pertains to the prediction of outputs represented as real numbers.

The second type of machine learning is *unsupervised learning*. In this approach, only input data $D = \{x_i\}_{i=1}^N$ is provided, and the objective is

to discover interesting patterns within the data. This process is a more difficult problem because there are no predefined patterns to search for, and there is no clear error metric to evaluate the performance. Common tasks in unsupervised learning are clustering and density estimation.

Reinforcement learning is the third paradigm in machine learning. It is employed to learn how to act or make decisions while interacting with an environment on the basis of reward signals, similar to how a baby learns to walk.

Machine learning finds applications in various domains, including image and speech recognition, natural language processing, recommendation systems, fraud detection, autonomous vehicles, and many others. It continues to advance with new algorithms, techniques, and tools, enabling the development of intelligent systems that can learn and adapt from data.

2.1 Supervised Learning

The scope of supervised learning is to forge a relationship between input data x and its associated output data y by leveraging a labeled training set D. This correlation aims to construct a model capable of offering predictions for new, unseen data.

To model the relationship between the input x and output y, we aim to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that maps the input space $\mathcal{X}$ to the output space $\mathcal{Y}$.

The model selects the best map among a parameterized family of functions f_θ selected in advance. The selection of the most suitable function is done by minimizing the average error between the predicted output $f_\theta(x)$ and the true output y across the training set D. The metric used to model the error is known as *loss function* $\ell(y, f(x))$.

The learning process in supervised learning can be formulated as an optimization problem, seeking to minimize the average loss over the training set (the *empirical risk* $\hat{\mathcal{E}}(f)$). Mathematically, this can be written as:

$$\hat{\theta} = \arg\min_\theta \hat{\mathcal{E}}(f) = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f_\theta(x_i)) \tag{2.1}$$

The choice of loss function is part of the design of the learning model and depends on the type of task and the nature of the problem. For classification

tasks, commonly used loss functions include the cross-entropy loss or the hinge loss. For regression tasks, popular choices of loss functions include the mean squared error or the mean absolute error.

Given a loss function, the "best" map is the function $f_* : X \to Y$ minimizing the *expected risk*

$$\mathcal{E}(f) = \mathbb{E}[\ell(y, f_\theta(x))] = \int dp(x, y)\ell(y, f_\theta(x)), \tag{2.2}$$

namely the expected value of the loss over the whole population $p(x, y)$. Since the latter is unknown, one relies on the minimization of the empirical risk in Eq. (2.1). On the other hand, being interested in prediction on unseen data, it is typically required to control how far is the solution of Eq. (2.1) to the best solution. For example, one possibility is to require an algorithm to be good in expectation, in the sense that

$$\mathbb{E}_D[\mathcal{E}(f_{\hat{\theta}}) - \mathcal{E}(f_*)]. \tag{2.3}$$

In simpler terms, a good algorithm should be able to fit the data while at the same time be robust against noise, i.e. avoid *overfitting*. Most learning models depend on a number of regularization parameters that control the trade-off between data-fitting and stability.

To find the optimal parameters $\hat{\theta}$, various optimization algorithms can be employed. One widely used approach is gradient descent, which iteratively updates the parameters in the opposite direction of the gradient of the loss function with respect to the parameters. This aspects will be addressed in more details in the following sections.

Considering these fundamental ingredients, numerous techniques and models have been developed. These encompass support vector machines, decision trees, random forests, as well as neural network models such as convolutional and recurrent neural networks. These approaches provide diverse capabilities and are well-suited for different applications.

2.2 Classification

In this chapter, we concentrate on the task of classification. As we already mentioned earlier, the output of a classification model is a member of a set of classes denoted as $y \in \{1, \ldots, C\}$, where C represents the total number of classes.

The most common case of two classes is known as *binary classification*. An example of binary classification is email spam detection, where the classes are typically labeled as "spam" and "non-spam". The goal is to determine whether an incoming email is categorized as spam or not. In binary classification scenarios, it is common to assume that the output y belongs to the set $\{0, 1\}$ or $\{-1, 1\}$.

When the number of classes exceeds two, it is referred to as *multiclass classification*. For instance, we could consider a scenario where we aim to classify different types of animals based on their characteristics. In this case, the classes could include "dog", "cat", "bird", and "fish". The objective is to correctly assign the appropriate class label to each animal based on its attributes.

In some cases, the class labels are not mutually exclusive. For instance, let us take a classification task where the objective is to identify the emotions conveyed in a text message. The classes could include "joy," "sadness," "anger," and "surprise." It is possible for a message to express multiple emotions simultaneously. This type of classification is known as *multi-label classification*.

In the context of a classification task, it is more favorable to generate an output in the form of a probability distribution. This distribution, denoted as $p(y|x)$, encapsulates the probabilities associated with potential labels, given the input vector x which is drawn from the training set D. Generally, this probability distribution is represented by a vector with a length of C, where C denotes the number of classes. In scenarios involving only two classes, it suffices to provide a single probability value, namely $p(y = 1|x)$, as the relationship $p(y = 1|x) + p(y = 0|x) = 1$ holds true. By explicitly incorporating the input x as condition, we signify that the probability is contingent on this variable, indicated by the conditioning bar "|". Moreover, our probability computation inherently assumes the utilization of a specific predictive model. In instances where diverse models are being compared, this can be made explicit by denoting $p(y|x, M)$, with M signifying the model. However, if the model is evident from the context, we can omit M in the notation for the sake of brevity.

Given a probabilistic output, we can determine our "best guess" for the "true label" by computing the class with the largest probability.

2.2.1 Binary Classification

Binary classification is a form of supervised learning task in which the goal is to categorize instances into either of two classes: positive or negative. Typically, labels such as $y = \{0, 1\}$ or $y = \{-1, 1\}$ are commonly used. This type of task is frequently encountered across diverse fields including spam detection, disease diagnosis, sentiment analysis, and fraud detection.

A popular method for binary classification is *logistic regression*. It utilizes the sigmoid or logistic function to map real-valued outputs to values between 0 and 1, hence allowing for a probabilistic interpretation of the results. However, before delving into the specifics of logistic regression, it is helpful to review the concept of maximum likelihood estimation.

Maximum Likelihood Estimation (MLE) To understand MLE, let's consider a simple example where we have a dataset of independent and identically distributed (i.i.d.) observations denoted as $x_1, x_2, \ldots, x_n$, assumed to be drawn from a probability distribution $p(x, \theta)$ with unknown parameters θ. MLE is an approach to find the values of these unknown parameters by maximising the likelihood of observing the given data.

The likelihood function, denoted as $L(\theta)$, represents the probability of observing the given data for a given set of parameters θ. In other words, it measures how likely the observed data is under the assumed probability distribution. It is defined as the joint probability density function. evaluated at the observed data points:

$$L(\theta) = \prod_{i=1}^{n} p(x_i; \theta) \tag{2.4}$$

Formally, we want to solve the following optimization problem:

$$\hat{\theta} = \arg\max_{\theta} L(\theta) \tag{2.5}$$

It is more convenient to work with the logarithm of the likelihood function. Taking the logarithm of the likelihood function does not change the location of the maximum, as the logarithm is a monotonically increasing function. However, it has the advantage of transforming the product of probabilities into a sum of logarithms, making the computation more efficient.

To find the maximum likelihood estimates, we differentiate the log-likelihood function with respect to the parameters and set the derivatives equal to zero:

$$\frac{\partial \log L(\theta)}{\partial \theta} = 0 \tag{2.6}$$

Solving this equation yields the values of the parameters that maximize the log-likelihood function, which are then considered as the maximum likelihood estimates.

It is worth noting that in practice, instead of directly maximizing the log-likelihood function analytically, iterative numerical optimization algorithms such as gradient descent or Newton's method are often employed.

MLE has several desirable properties, including consistency, asymptotic normality, and efficiency under certain regularity conditions. Additionally, MLE allows for statistical inference, such as constructing confidence intervals and performing hypothesis tests, based on the estimated parameters. For more information regarding this topic please refer to [3], [4], and [5].

Logistic Regression In logistic regression, one models the probability of an instance x_i to belong to the positive class by using the sigmoid function, i.e. $\sigma(f_\theta(x_i))$, where $f_\theta(x_i)$ is the output of the model which depends on the specific class of functions that are being explored by the algorithm (typically linear functions). In this way, the output is mapped into a probability value between zero and one.

The logistic function (sigmoid function) is defined as:

$$\sigma(\eta) = \frac{1}{1 + \exp(-\eta)} \tag{2.7}$$

This function is an s-shaped curve (Figure 2.1), effectively squashing input values within the range of zero and one, enabling interpretation of the output as a probability.

The logistic regression model can be expressed as:

$$p(y = 1|x, \theta) = \sigma(\theta^T x) \tag{2.8}$$

where $p(y = 1|x, \theta)$ represents the probability of the positive class given the input variables x and the model parameters θ.

Training the logistic regression model means finding the optimal values for the model parameters θ from the training data.

Assuming that the binary labels are independent and identically distributed (i.i.d.), the likelihood function can be expressed as the product of

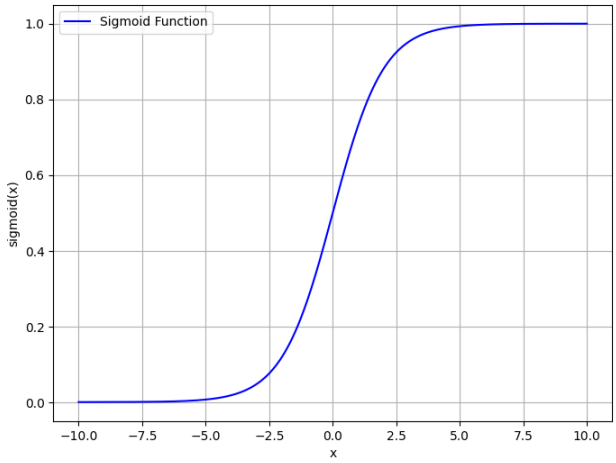


Figure 2.1: Sigmoid function

the individual probabilities:

$$L(\theta) = \prod_{i=1}^{N} [p(y_i = 1 \mid x_i, \theta)]^{y_i} \, [1 - p(y_i = 1 \mid x_i, \theta)]^{1-y_i} \tag{2.9}$$

where N represents the number of instances in the dataset. Rather than aiming to maximize the log-likelihood, we can achieve the same objective by minimizing the negative log-likelihood (NLL) function. The NLL function for logistic regression has the following form:

$$\text{NLL}(\theta) = -\sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \tag{2.10}$$

Here, y_i represents the true outcome (0 or 1), and $\hat{y}_i = p(y_i = 1|x_i, \theta)$ represents the predicted probability of y_i being 1 given the input data and the model parameters θ. This conversion allows us to utilize optimization algorithms designed for minimization problems.

The negative log-likelihood (NLL) function is a fundamental component of logistic regression which is also known as the cross-entropy error function. It measures the disagreement between the predicted probabilities $\hat{y}_i$ and the actual binary outcomes y_i.

Once the model parameters are estimated, predictions for new instances can be made by computing the probability $p(y = 1|x, \theta)$. A common decision

rule is to classify an instance as positive (1) if $p(y = 1|x, \theta)$ exceeds a certain threshold (e.g., 0.5), and as negative (0) otherwise.

Alternatively, we can consider a different encoding scheme by using a new representation $\tilde{y}_i \in \{-1, +1\}$ instead of $y_i \in \{0, 1\}$. This allows us to redefine the probabilities as $p(y = 1) = \frac{1}{1+\exp(-\theta^T x)}$ and $p(y = -1) = \frac{1}{1+\exp(\theta^T x)}$. These probabilities are derived from the logistic function, which maps the linear combination of the input data x and model parameters θ to a probability value between -1 and 1.

With this transformation, the NLL function takes the form:

$$\text{NLL}(\theta) = -\sum_{i=1}^{N} \log(1 + \exp(-\tilde{y}_i \theta^T x_i)) \tag{2.11}$$

Unlike the linear least square estimator, we cannot directly derive the solution of the logistic regression problem in closed form. Instead, we need to employ an iterative optimization algorithm to compute it. By iteratively adjusting the model parameters based on these quantities, the optimization algorithm converges to the optimal parameter values that minimize the NLL function. To delve deeper into this aspect, additional information can be found in references [3–6].

2.3 Nonlinear models

Non-linear models in machine learning are a crucial advancement that expands the capabilities of traditional linear models. These models are designed to handle intricate relationships within data that defy simple linear patterns.

Although they provide versatility and precision, non-linear models are susceptible to overfitting. To counter this challenge, regularization methods, like L1 and L2 regularization, come into play by incorporating penalty components into the loss function. An additional valuable tool for appraising a model's efficacy with unfamiliar data is cross-validation.

In the upcoming segment, we will briefly discuss two preeminent and extensively utilized non-linear models that hold paramount significance in a spectrum of machine learning endeavors. Specifically, we will shed light on *Neural Networks* and *Kernel Methods*, elucidating their inherent strengths and versatile applicability within diverse machine learning tasks.

Neural Networks Neural networks are a class of models inspired by the structure and functioning of the brain. They are composed of interconnected nodes, called neurons, organized in layers. Neural networks have gained significant popularity due to their ability to learn complex patterns and relationships in data.

A typical neural network consists of three main types of layers: input layer, hidden layers, and output layer. The input layer receives the input data, which may undergo pre-processing procedures. The hidden layers, as the name suggests, are intermediate layers between the input and output layers. Their purpose is to combine and process the input data using non-linear methods. The output layer produces the final predictions or outputs of the model.

In a basic fully connected feed-forward network, the fundamental computation happening at the level of the single neuron involves two primary steps: the weighted sum of inputs and the application of an activation function. Each neuron receives inputs from the preceding layer, multiplies them by a set weights, and calculates the weighted sum. Optionally, a bias term can be incorporated into the weighted sum. For a neuron indexed as j in layer l, the weighted sum z_j^l is then computed as:

$$z_j^l = \sum_{k=1}^{n} w_{jk}^l \cdot a_k^{l-1} + b_j^l \tag{2.12}$$

where w_{jk}^l represents the weight connecting the k-th neuron in layer $l-1$ to the j-th neuron in layer l, a_k^{l-1} is the output of the k-th neuron in layer $l-1$, and b_j^l is the bias term for the j-th neuron in layer l.

After computing the weighted sum, an activation function is applied element-wise to introduce non-linearities into the model. The activation function determines the output of the neuron based on the computed weighted sum. Popular activation functions include:

- **Sigmoid function** : $\sigma(z) = \frac{1}{1+e^{-z}}$
- **Hyperbolic tangent (tanh) function** : $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- **Rectified Linear Unit (ReLU) function** : $\mathrm{ReLU}(z) = \max(0, z)$
- **Softmax function (for multi-class classification)** : $f(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$, where K is the number of classes.

During the learning process of a neural network, the weights and biases are adjusted to minimize a loss function. *Backpropagation* is a widely utilized optimization algorithm for training neural networks based on the chain rule to compute the derivative of composition of functions. It involves a backward pass, starting from the output layer and moving towards the input layer, to update the model's parameters based on the gradient of the loss function. By iteratively adjusting the weights and biases using backpropagation, the network strives to improve its performance in making accurate predictions.

The architecture of a neural network, including the number of layers, the number of neurons in each layer, the choice of activation functions, and the optimization algorithm, can vary depending on the problem at hand. Different architectures and hyperparameters can be explored and optimized through experimentation and validation on a separate validation dataset.

Neural networks have been successfully applied to various domains, including image recognition, natural language processing, speech recognition, and many others. Their ability to learn complex representations and capture intricate patterns makes them powerful tools in machine learning and artificial intelligence. [7], [8]

Kernel Methods Kernel methods are a popular class of models used for solving non-linear learning problems. The main idea is to transform the input data into a higher-dimensional feature space, where it may become easier to separate or classify the data. This transformation is achieved by using a non-linear feature map $\phi : X \to \mathbb{R}^d$, from the space of the input data to $\mathbb{R}^d$, where the dimensionality d is higher than the one of the space X. This map can be intepreted as a sort of preprocessing of the data. With this choice, one then considers linear models of the following kind

$$f_w(x) = \sum_{i=1}^{d} w^i \phi^i(x), \tag{2.13}$$

with w the parameters to be learned. Thanks to the Representer Theorem ([9]), one can consider infinite dimensional maps as long as the following *kernel* $K = \phi^T \phi$ can be computed. Eq. (2.13) can then be rewritten as

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i), \tag{2.14}$$

where the sum now runs over the data points and $w = \sum_{i=1}^{n} \phi_i^T(x)\alpha_i$. A kernel K, in order to be admissible, should behave like an inner product. More precisely it should be symmetric and positive semi definite. The symmetry property is typically easy to check, however positive semi definiteness is more complicated to determine. Popular examples of positive definite kernels include:

- linear kernel $K(x, x') = x^T x'$,
- polynomial kernel $K(x, x') = (x^T x' + 1)^d$,
- Gaussian (RBF) kernel $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$,

where the last two kernels have a hyper-parameter, the degree d and Gaussian width σ, respectively.

The decision function can then be seen as a linear combination of kernel evaluations between the new input x and the training samples x_i. By using appropriate kernel functions, the decision function can capture complex relationships and achieve non-linear decision boundaries.

The coefficients, represented by α_i are typically learned using iterative methods given a loss function but certain problems, such as kernel ridge regression, can also be solved directly, although less efficiently.

Kernel methods provide several benefits. Firstly, they are advantageous from an optimization perspective as they are linear models. Secondly, they can be highly adaptive by selecting the appropriate kernel and with a careful hyperparamter selection. Additionally, they avoid the need for an ad-hoc selection of high-dimensional feature maps. Lastly, kernel methods have a strong theoretical foundation rooted in functional analysis, offering mathematical justification for their application and delivering robust algorithms with statistical guaranties.

Examples of kernel-based algorithms include Support Vector Machines (SVM), Gaussian Processes, and Kernel Principal Component Analysis (PCA). [10]

2.4 Optimization

Optimization is a mathematical process utilized across various fields, including mathematics, engineering, economics, and computer science, to determine

the best possible solution within a given set of constraints. The primary goal of optimization is to either maximize or minimize an objective function, which represents the quantity or criteria being optimized.

To minimize the error function in a learning problem, iterative methods that involve computing the gradient are commonly employed. One such method is known as gradient descent, which is a first-order iterative algorithm utilized to locate a local minimum of a differentiable function. [11], [7], [12]

Gradient Descent Gradient Descent (GD) is an iterative optimization algorithm extensively utilized in model training, particularly in machine learning and deep learning, to locate the minimum of a differentiable objective function. The primary concept behind GD is to iteratively adjust the model's parameters in the direction opposite to the gradient of the objective function. This process leads to descending the surface of the function and ultimately reaching a local minimum (Fig. 2.2).

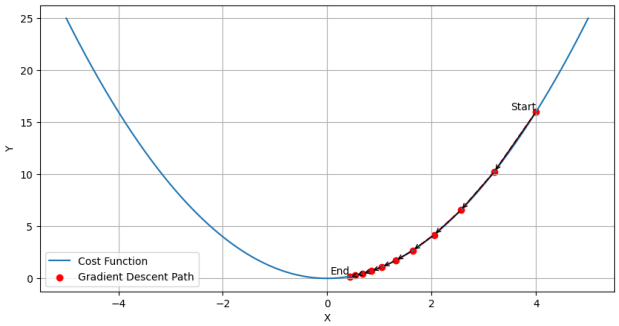


Figure 2.2: Gradient Descent

The general update equation for GD can be expressed as:

$$\theta = \theta - \alpha \nabla J(\theta) \tag{2.15}$$

In this equation: θ is the vector of model parameters. α is the learning rate, which controls the step size in each iteration. $\nabla J(\theta)$ denotes the gradient of the objective function $J(\theta)$ with respect to θ.

During each iteration of GD, the algorithm calculates the gradient of the objective function with respect to the parameters. This gradient informs the algorithm of the direction in which the parameters should be adjusted to

minimize the objective function. By multiplying the gradient by the learning rate, the algorithm controls the step size taken in each iteration. A higher learning rate results in larger parameter updates, potentially leading to faster convergence but risking overshooting the minimum. Conversely, a lower learning rate ensures smaller and more cautious updates, at the cost of slower convergence. Thus, the learning rate is a crucial hyperparameter that needs to be carefully tuned.

GD iteratively updates the parameters by subtracting the learning rate multiplied by the gradient from the current parameter values. This update process continues until a stopping criterion is met, such as reaching a maximum number of iterations or achieving a sufficiently small change in the objective function.

It is important to note that GD seeks to find a local minimum of the objective function. Depending on the function's characteristics, this local minimum may correspond to the global minimum if the objective function is convex. However, in non-convex scenarios, GD may converge to a suboptimal solution that is not the global minimum.

Gradient descent (GD) is a commonly used optimization algorithm known for its simplicity and effectiveness. However, there are variations of GD, such as accelerated or conjugate gradient methods, that can achieve faster convergence rates and address challenges associated with local minima or saddle points. These alternative methods offer improvements in terms of convergence speed and robustness when dealing with complex optimization problems.

2.5 Anomaly Detection

Anomaly detection aims to recognize patterns or occurrences that significantly deviate from the norm or expected behavior in a given dataset. These anomalies can manifest as uncommon events, errors, outliers, or suspicious activities that differ from the majority of the data. The field of anomaly detection finds utility in diverse domains, including fraud detection, network intrusion detection, system monitoring, and medical diagnosis. [13], [14]. Subsequently, we provide a brief mention of several typical anomaly detection tasks and approaches, as they exhibit certain resemblances to the analysis conducted in this thesis.

Supervised anomaly detection Supervised anomaly detection leverages labeled data, which includes both normal and anomalous instances, to train a model for classifying new instances as normal or anomalous. The main goal is to develop a mapping function that associates input features with the anomaly label, based on the provided labels in the training set.

The objective is then to train a model that can effectively classify instances as normal or anomalous based on their input features. Various models, such as support vector machines (SVMs), decision trees, random forests, or neural networks, can be employed for this purpose.

Since these types of models are in essence traditional classification models, the process proceeds as explained in the previous sections. Different evaluation metrics can be selected depending on the specific type of data or domain of application. For a more in-depth understanding of this topic, one may consult the reference [14].

Unsupervised Anomaly Detection Unsupervised anomaly detection involves identifying anomalies in data without the use of labeled examples. The goal is to detect patterns that deviate significantly from the norm, making them potential anomalies. Unlike supervised anomaly detection, there are no pre-labeled instances, and the detection is based solely on the characteristics of the data itself.

The first step in unsupervised anomaly detection is to represent the data. Each data instance is typically represented as a feature vector or a multidimensional point. Let's denote a data instance as $\mathbf{x} \in \mathbb{R}^d$, where d represents the number of features or dimensions in the data.

Unsupervised anomaly detection assumes that the majority of the data instances are representative of normal behavior, and anomalies are rare occurrences. The task is to build a model of the normal behavior that captures the underlying patterns in the data. Various techniques can be used for this purpose, such as density-based methods, distance-based methods, clustering algorithms, or probabilistic models.

Density-based techniques aim to approximate the density of data points within the feature space. Among these methods, the Gaussian Mixture Model (GMM) is widely used. The GMM represents the data by combining multiple Gaussian distributions, estimating their means, covariances, and weights. Each Gaussian component captures a distinct mode of the data distribution. Anomalies are identified as instances with a low probability of occurrence

based on the estimated density.

Distance-based methods, such as the k-nearest neighbors (k-NN) algorithm, are utilized in anomaly detection by assessing the dissimilarity between instances. The k-NN algorithm involves several steps, including distance calculation, selection of the k nearest neighbors, computation of anomaly scores, and determination of a threshold for classification. The widely adopted Euclidean distance is employed to measure dissimilarity. Once distances are calculated, the k nearest neighbors are identified, and anomaly scores are assigned based on the relationship with these neighbors. Thresholds are then established to classify instances as anomalies or normal, considering their scores. Customization of the distance metric and anomaly score calculation allows adaptation to specific requirements. Anomaly scores provide a quantitative measure of the extent to which instances deviate from the normal model, with lower scores indicating a higher likelihood of being an anomaly.

Various evaluation metrics can be used to assess the performance of unsupervised anomaly detection. Common metrics include precision, recall, F1 score, receiver operating characteristic curve (ROC curve), and area under the ROC curve (AUC-ROC). These metrics provide insights into the accuracy and effectiveness of the anomaly detection algorithm.

It is important to note that the specific equations and notions can vary depending on the chosen unsupervised anomaly detection technique. Different methods have different underlying assumptions and mathematical formulations. Therefore, the equations and algorithms mentioned above are general concepts that can be applied across different unsupervised anomaly detection approaches.

Chapter 3

The New Physics Learning Machine

3.1 Hypothesis Testing

Hypothesis test is a statistical method for testing claims about parameters in a population from data. The process typically involves four steps. Initially, we define the null and alternative hypotheses. Then, a test statistic is chosen to summarizing the strength of evidence against the null hypothesis. After that, a value is computed that characterizes the test statistic, indicating how likely it is to obtain a similar or more extreme test statistic value under the assumption that the null hypotheses is true. Lastly, the derived value is employed to make a decision regarding the initial assumption.

Hypothesis testing divides the possibilities into two scenarios: the null hypothesis $H_0 : \theta \in \Theta_0$ and the alternative hypothesis $H_1 : \theta \in \Theta_0^c$. θ denotes the population parameters, Θ_0 is some subset of the parameter space, and Θ_0^c is its complement. H_0 is the default belief about the world, while H_1 represents something different and unexpected. The treatment of H_0 and H_1 is asymmetric, as we focus on using data to reject H_0 and, in turn, provide evidence in favor of H_1. If we fail to reject H_0, our conclusions become less definitive. Meaning that we will be uncertain whether our failure to reject H_0 is due to the limited size of our sample (in such case, conducting another test on a larger or higher-quality dataset could potentially lead to rejection) or if H_0 truly remains valid.

In order to establish evidence supporting or refuting the null hypothe-

sis, we calculate a test statistic T that summarizes the degree of alignment between our data and H_0. The specific test statistic used depends on the nature of the data and the hypothesis being examined.

The sampling distribution of the test statistic under the null hypothesis, $p(T|H_0)$, describes how it behaves when we repeatedly draw samples from the population under the assumption of the null hypothesis. The shape of the distribution $p(T|H_0)$ is mainly influenced by the specific choice of the test statistic T and the sample size.

The observed test statistic T_{obs} is the actual value computed from our sample data. It is then compared to the sampling distribution $p(T|H_0)$ to determine the likelihood of observing such an extreme value if the null hypothesis were true. This comparison helps us evaluate the evidence supporting or contradicting the null hypothesis.

The p-value quantifies the probability of observing a test statistic as extreme or more extreme than the observed one, given that H_0 is true. A small p-value indicates evidence against H_0. The distribution of the test statistic under H_0 depends on the specific null hypothesis and test statistic used.

Hypothesis testing involves two types of hypotheses: a two-sided hypothesis (also known as a two-tailed hypothesis) and a one-sided hypothesis (also known as a one-tailed hypothesis). The choice between these two types determines how the p-value is interpreted.

The P-value for a one-sided right-tail and test-statistic distribution is defined as

$$\begin{aligned} \text{p-value} &= P(T \geq T_{\text{obs}} \mid H_0) \quad \text{for right-tail} \\ \text{p-value} &= P(T \leq T_{\text{obs}} \mid H_0) \quad \text{for left-tail} \end{aligned} \tag{3.1}$$

And for a two-sided test-statistic distribution as

$$\text{p-value} = 2\min\left(P(T \geq T_{\text{obs}} \mid H_0), P(T \leq T_{\text{obs}} \mid H_0)\right) \tag{3.2}$$

In a two-sided hypothesis test, the goal is to assess whether there exists a notable distinction between the sample data and the null hypothesis, encompassing various potential directions. For instance, if the null hypothesis indicates the absence of a difference, the two-sided alternative hypothesis implies the potential for differences to be observed in multiple directions. This includes the possibility of observing values that are either greater or lesser in magnitude. The two-sided p-value considers evidence in both tails

of the sampling distribution $p(T|H_0)$, measuring the probability of observing an extreme test statistic value in either direction.(Figure 3.1)

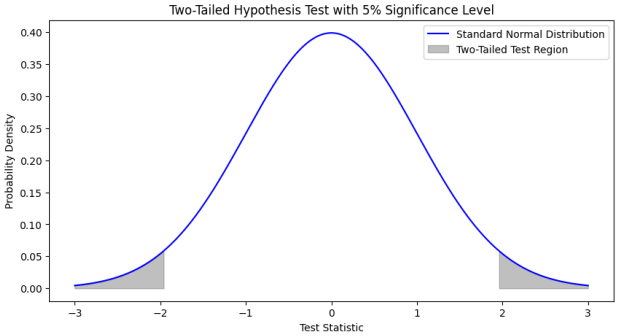


Figure 3.1: Two Tailed

In a one-sided hypothesis test, the emphasis is placed on assessing evidence for a distinction in a particular direction. It quantifies the likelihood of encountering an exceptionally extreme test statistic value in a single tail of the sampling distribution $p(T|H_0)$, see Figures 3.2 and 3.3.

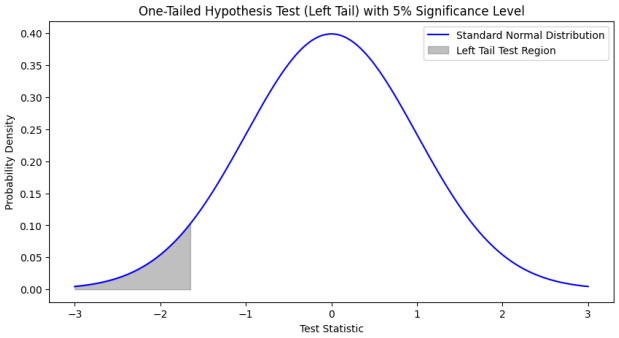


Figure 3.2: Left Tailed

Once we calculate the p-value corresponding to H_0, we need to decide whether to reject H_0 or not. As already mentioned, a small p-value indicates that the observed test statistic is unlikely to occur under H_0, providing evidence against H_0. The significance level (often denoted as α) determines when to reject H_0; if the p-value is lower than this threshold, we reject H_0 and claim a "discovery" as a strong evidence against H_0 is present. The precise value of the threshold depends on the problem at hand (a typical value is $\alpha = 0.05$).

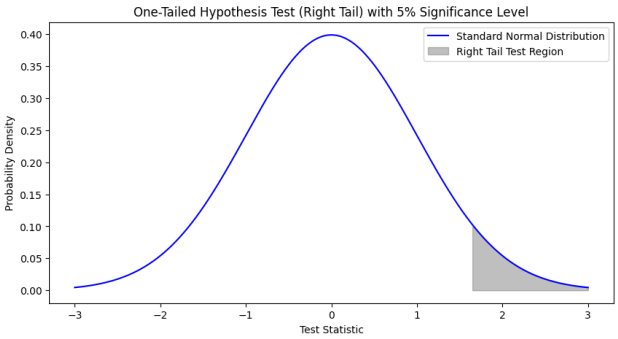


Figure 3.3: Right Tailed

It is important to note that the p-value is often misunderstood. It does not represent the probability that H_0 is true; instead, it signifies the frequency of seeing such an extreme test statistic if we were to repeat the experiment multiple times assuming H_0 holds. Reporting a two-sided p-value is generally recommended, except when there is a compelling reason to use a one-sided p-value.

Hypothesis testing involves the consideration of two types of errors. Type I errors, or *false positives*, occur when the null hypothesis is wrongly rejected despite being true in the population. Type II errors, or *false negatives*, arise when the null hypothesis is not rejected even though it is false in the population. The significance level and the *power* of the test influence the likelihood of committing these errors.

The power of a test is the probability that the test rejects the null hypothesis when a specific alternative hypothesis is true. It is typically denoted by $1-\beta$, with β the probability of type II errors, and represents the chances of a true positive.

The critical region corresponds to the set of values of the test statistic for which the null hypothesis is rejected. It is determined based on the significance level and the distribution of the test statistic. Values of the test statistic falling within the critical region lead to the rejection of the null hypothesis.

There are several hypothesis tests that can be used in different situations. For instance, the Z-Test is used to test the mean of a population when the population standard deviation is known or the sample size is large. On the other hand, the t-Test is employed when testing the mean of a population assuming the population standard deviation is unknown or the sample size is

small. The Chi-Square Test is specifically used for hypothesis testing involving categorical data and examining the independence between variables. The ANOVA (Analysis of Variance) is applied when testing the means of multiple groups or treatments. For more details on hypothesis testing, please refer to Ref.s [5,6,15].

3.2 A classifier for hypothesis testing

We utilize the *New Physics Learning Machine* (NPLM) method for hypothesis testing, developed in [1,2] in the context of model-independent searches of new physics in collider experiments. In that scenario, the method compares measured data $\mathcal{D} = \{x_i\}_{i=1}^{N_1}$ with a *reference dataset* $\mathcal{R} = \{x_i\}_{i=1}^{N_0}$, which follows the statistical distribution $p(x|R)$ predicted by a reference model R which is typically not known in closed form (i.e., the Standard Model of Particle Physics). The primary goal is to evaluate the accuracy of the distribution prediction and determine whether the standard laws adequately represent the experimental data or if new physical laws are required.

Adapting the NPLM approach to fraud detection problems is straightforward. In our fraud detection scenario, we initially analyze the distribution of genuine data to understand its characteristics. With this understanding, our next step involves detecting anomalies within the dataset that could potentially signify instances of fraudulent behavior.

In the discussion that follows, one set of independent and identically distributed random variables (i.i.d.) of $p(x|R)$ is represented by

$$\mathcal{R} = \{x_i\}_{i=1}^{N_0}, \text{ where } x_i \overset{\text{i.i.d.}}{\sim} p(x|R), \tag{3.3}$$

and the actual measured data by,

$$\mathcal{D} = \{x_i\}_{i=1}^{N_1}, \text{ where } x_i \overset{\text{i.i.d.}}{\sim} p_{true}(x), \tag{3.4}$$

where $p_{true}(x)$ is the true and unknown distribution of the data. It is crucial to emphasize that in real-world scenarios, it becomes imperative to account for the statistical uncertainties that impact the understanding of the reference model. Similarly to Refs. [1,16], we will adopt the assumption that $N_0 \gg N_1$ to minimize the impact of the statistical uncertainties linked with the reference sample. Although it is possible to integrate systematic uncertainties as auxiliary variables, as exemplified in Reference [17] for neural network

applications, for the context of this discourse, we will consider systematic uncertainties as insignificant.

The concept outlined in Reference [1] involves converting the process of maximizing the log-likelihood-ratio test into a machine learning challenge. In this approach, the null hypothesis, which defines one of the likelihood terms, corresponds to the reference hypothesis. Simultaneously, the alternative hypothesis linked to the other likelihood term remains undetermined initially and is acquired from the data itself during the training phase. Consequently, the resulting test statistic achieved through this method serves as a commendable approximation of the optimal test statistic, as outlined by the Neyman-Pearson lemma.

The likelihood of the data set $\mathcal{D}$ under a generic hypothesis H is defined as

$$\begin{aligned} L(\mathcal{D}, H) &= \frac{e^{-N(H)} N(H)^{N_1}}{N_1!} \prod_{i=1}^{N_1} p(x_i|H) \\ &= \frac{e^{-N(H)}}{N_1!} \prod_{i=1}^{N_1} n(x_i|H) \end{aligned} \tag{3.5}$$

where

$$n(x|H) = N(H)p(x|H) \tag{3.6}$$

represents the data distribution normalized by the expected number of events

$$N(H) = \int n(x|H)\, dx \tag{3.7}$$

While $p(x|R)$ is well-known and accurately represented by the reference sample, $p_{true}(x)$ is unknown and must be approximated by a family of distributions $p(x|H_w)$, parameterized by a trainable set of variables w. The likelihood ratio test statistic is given by

$$
\begin{aligned}
t_w(\mathcal{D}) &= -2\log\frac{L(\mathcal{D},R)}{L(\mathcal{D},H_w)} \\
&= -2\log[e^{N_w(1)-N(0)}\prod_{i=1}^{N_1}\frac{n(x_i|R)}{n(x_i|H_w)}] \\
&= -2[N_w(1)-N(0)-\sum_{i=1}^{N_1}\log\frac{n(x_i|H_w)}{n(x_i|R)}].
\end{aligned}
\tag{3.8}
$$

which is then optimized by maximizing over the parameter set w. [1] The initial suggestion in Reference [1] proposed utilizing the capacity of neural networks as universal approximators to establish a collection of functions that delineate the logarithmic ratio of the density distributions mentioned in Equation 3.8.

$$
f_w(x) = \log\frac{n(x|H_w)}{n(x|R)} \tag{3.9}
$$

While it could be a neural network, in this case, we will opt for kernel methods for specific reasons that will be clarified later. The model is trained by adjusting its parameters to best fit the observed data, effectively accommodating the best-fit hypothesis $H_{\hat{w}}$ with the trained parameters $\hat{w}$.

Following the above reasoning, the maximum of the test statistic can be recast as the minimum of a loss function $L(f_w)$

$$
\begin{aligned}
t_{\hat{w}}(\mathcal{D}) &= \max_w\ t_w(\mathcal{D}) \\
&= -2\min_w\ L(f_w) \\
&= -2\min_w\ [\sum_{x\in\mathcal{R}}\frac{N(0)}{N_0}(e^{f_w(x)}-1)-\sum_{x\in\mathcal{D}}f_w(x)]
\end{aligned}
\tag{3.10}
$$

with the parameter set $\hat{w}$ that maximizes $t_w(\mathcal{D})$

$$
n(x|H_{\hat{w}}) = n(x|R)e^{f_{\hat{w}}(x)} \approx n_{true}(x), \tag{3.11}
$$

provides also the best approximation of the true underlying data distribution and with it a first insight on the source and shape of the discrepancy,

[1] Note that, for simplicity, we are identifying the null hypothesis with the symbol used for the reference model, i.e. $H_0 = R$.

if present. Deriving Equation 3.10 from Equation 3.8 requires an estimation of the anticipated event count under the alternative hypothesis. This estimation can be achieved through the utilization of Equation 3.11 in conjunction with the Monte Carlo method, namely

$$\begin{aligned} N_w(1) &= \int n(x|H_w)\,dx = \int n(x|R)e^{f_w(x)}\,dx \\ &\approx \sum_{x\in R} \frac{N(0)}{N_0} e^{f_w(x)} \end{aligned} \tag{3.12}$$

In this study, we further elaborate on the aforementioned concepts by incorporating an alternative loss function, specifically a weighted cross-entropy (logistic) loss function. This alternative was previously suggested as a viable option in Reference [1], and we demonstrate its numerous advantages. During the tests of the NPLM method conducted so far, using the logistic loss in place of the maximum likelihood loss has not shown any significant performance degradation. [2].

To compute the ratio in Equation 3.11, we train a binary classifier utilizing the weighted cross-entropy loss.

$$\ell(y, f(x)) = a_0(1-y)\log(1+e^{f(x)}) + a_1 y \log(1+e^{-f(x)}) \tag{3.13}$$

where y represents the class label, taking on a value of zero for R and one for D. The classifier is obtained by minimizing an empirical criterion.

$$\hat{\ell}(f_w) = \frac{1}{N}\sum_{i=1}^{N} \ell(y, f_w(x)) \tag{3.14}$$

over a suitable class of machine learning models f_w . If the class of such models is adequately diverse, as the sample size becomes large, we would converge towards a solution that minimizes the expected risk.

$$\ell(f) = \int \ell(y, f(x))\,dp(x,y) \tag{3.15}$$

where $p(x, y)$ is the joint data distribution.

By establishing $a_0 = \frac{N(0)}{N_0}$ and $a_1 = 1$ through the utilization of Equation 3.13, the test statistics described in Equation 3.8 can be expressed as follows:

$$t_{\hat{w}}(\mathcal{D}) = -2[\frac{N(0)}{N_0}\sum_{x\in\mathcal{R}}(1 - e^{f_{\hat{w}}(x)}) + \sum_{x\in\mathcal{D}} f_{\hat{w}(x)}] \tag{3.16}$$

recovering the original result from Ref. [1].

In conclusion, it is important to highlight that the test statistic value $t_{\hat{w}}(\mathcal{D})$ is inherently a random variable, following a distribution denoted as $p(t|H)$. The significance level attributed to a particular test statistic value is determined by calculating its p-value in relation to the distribution it follows under the null hypothesis.

$$p_{\mathcal{D}} = \int_{t(\mathcal{D})}^{\infty} p(t|R)\, dt \tag{3.17}$$

This can be further rewritten as a Z-score

.

$$Z_{\text{obs}}(\mathcal{D}) = \Phi^{-1}(1 - p_{\mathcal{D}}) \tag{3.18}$$

where Φ^{-1} is the quantile of a Normal distribution. In this way Z_{obs} is expressed in units of standard deviations. In accordance with the approach detailed in Reference [1], we exploit the opportunity to draw samples from the reference distribution. As a result, we opt to reconstruct $p(t|R)$ by evaluating the likelihood ratio test statistics across a set of N_{toy} simulated experiments conducted on synthetic datasets derived from the reference data. These synthetic datasets share the statistical properties of the real data but lack any anomalies.

The comprehensive analytical approach employed by the NPLM technique for fraud detection can be succinctly encapsulated within three distinct stages:

- The empirical construction of the test statistic distribution under the null hypothesis involves training through approximately $N_{\text{toy}} = \mathcal{O}(300)$ reference-distributed toy experiments against the reference sample $\mathcal{R}$.

- A conclusive round of training is executed utilizing the dataset of interest, denoted as $\mathcal{D}$. In this context, the actual underlying hypothesis is unknown, and the test statistic value $t(D)$ is computed.

- The p-value associated with $t(D)$ is calculated relative to the test statistic distribution established during the first step under the null hypothesis.

Should a statistically significant deviation from the reference data be detected, further insight into the nature of the discrepancy can be gained by analyzing the acquired density ratio as described in Equation 3.11. This quantity is anticipated to approach zero when no inconsistencies are present, and its behavior can be explored in relation to input features or their combinations. Since we are working in a controlled scenario, we compute the observed test statistics for multiple i.i.d. copies of $\mathcal{D}$ to characterize the distribution under the alternative hypothesis.

Asymptotic Formula Typically, achieving an accurate estimate of $p(t|R)$ requires reconstructing the empirical distribution of the test statistic under the reference hypothesis through a large number of toy experiments, which can often be practically unfeasible. If the value of $t(\mathcal{D})$ falls outside of the range of the empirical distribution the p-value cannot be computed and only a lower bound can be set. Drawing inspiration from the findings of Wald and Wilks [18]- [19], which characterizing the asymptotic behavior of log-likelihood test statistics, we approximate the null distribution using a χ^2 distribution. This approximation involves determining the degrees of freedom of the χ^2 distribution based on an empirical estimate derived from toy experiments. We then employ a Kolmogorov-Smirnov test to evaluate the consistency of the empirical test statistic distribution with the χ^2 hypothesis. This approximative approach holds up well in nearly all instances of our model. This same approximation technique is also applied in the neural network model presented in references [1] and [16]. It is important to emphasize that in real-world scenarios, if the computed p-value using this approach suggests a discovery, additional toy experiments would be executed to achieve a precise empirical estimation by exhaustively utilizing extensive computing resources.

This estimation method allows the computation of very small p-values, which correspond to highly discrepant data with very large $t(\mathcal{D})$. However, the agreement between $p(t|R)$ and the χ^2 distribution cannot be verified in the high $t(\mathcal{D})$ region that the toys do not populate. Thus, the quantification of the p-value is accurate only in the region where the toys are statistically represented. For example, if 300 Toys are generated, only p-values larger than approximately 1/300 can be accurately computed.

Moreover, the possibility of reconstructing the data distribution using $f_{\hat{w}}$ serves as a useful debugging tool. It enables verification of whether the

learning model correctly recognizes deviations from the reference distribution and how it handles such discrepancies. However, we did not explore this possibility in this work and we leave it for future developments.

3.2.1 Efficient kernel methods

The performance of the kernel-based version of NPLM relies on the capabilities of the Falkon library, which forms the foundation of our implementation. The fundamental theoretical and algorithmic concepts implemented in Falkon, as outlined in Ref. [20]- [21], are summarized below.

In kernel methods, the functions are learned in the form of a weighted sum of kernel functions applied to the training dataset. Specifically, the function $f_w(x)$ is expressed as:

$$f_w(x) = \sum_{i=1}^{N} w_i k_\sigma(x, x_i) \tag{3.19}$$

where $N = N_0 + N(1)$ where $N(1) = N(0) + N(S)$ represents the total size of the training dataset, $k_\sigma(x, x_i)$ is the kernel function with a hyperparameter σ, and w are the model parameters to be optimized. In our analysis, we employ the Gaussian kernel

$$k_\sigma(x, x') = e^{-||x-x'||/2\sigma^2} \tag{3.20}$$

where f_w is a linear combination of Gaussians with a constant width σ centered at the training data points. The optimization involves minimizing the empirical risk $\hat{L}(f_w)$ augmented with a regularization term:

$$\hat{L}_\lambda(f_w) = \hat{L}(f_w) + \lambda R(f_w) \tag{3.21}$$

The empirical risk $\hat{L}(f_w)$ is determined by the logistic loss function (3.13) and can be expressed as:

$$\hat{L}(f_w) = \sum_{i=1}^{N} \ell(y_i, f_w(x_i)) \tag{3.22}$$

The regularization term $R(f_w)$ is given by:

$$R(f_w) = \sum_{i,j} w_i w_j k_\sigma(x_i, x_j) \tag{3.23}$$

The hyperparameter λ controls the relative importance of the regularization term in the optimization objective (3.21).

Kernel methods are categorized as non-parametric approaches because the number of parameters w in Eq. (3.19) automatically increases with the total number of data points. In the limit of a large sample, they have the ability to recover any continuous function (Ref. [22], [23]). However, optimizing the function in Eq. (3.19) with the target in Eq. (3.21) involves dealing with an $N \times N$ matrix, known as the kernel matrix, with entries $k_\sigma(x_i, x_j)$. This optimization process exhibits a cubic time complexity and quadratic space complexity with respect to the number of training points N (Ref. [24], [20]), making it computationally expensive for large-scale settings. Therefore, some approximation methods are needed to handle such cases efficiently.

The Falkon library approaches the minimization problem of Eq. (3.21) using an approximate Newton method, detailed in Algorithm 2 of [24]. This algorithm relies on the Nyström approximation, which is employed twice.

Firstly, to reduce the problem's size, solutions are considered in the form of:

$$f_w(x) = \sum_{i=1}^{M} w_i k_\sigma(x, \tilde{x}_i) \tag{3.24}$$

where $\{\tilde{x}_1, ..., \tilde{x}_M\} \subset \{x_1, ..., x_N\}$ are known as Nyström centres and are sampled uniformly at random from the input data. The value of $M < N$ is a hyperparameter that needs to be determined.

Secondly, the Nyström approximation is used once again to obtain an approximate Hessian matrix:

$$\tilde{\boldsymbol{H}} = \frac{1}{M} T \tilde{D} T^T + \lambda I \tag{3.25}$$

Here, T is structured in such a way that $T^T T = \tilde{K}$ (Cholesky decomposition), where $\tilde{K} \in \mathbb{R}^{M \times M}$ represents the kernel matrix subsampled with respect to both rows and columns. Additionally, $\tilde{K}$ is a diagonal matrix where the i-th element corresponds to the second derivative of the loss $\ell''(y_i, f_w(x_i))$ concerning its first variable.

To perform conjugate gradient descent, Eq. (3.25) is used as a preconditioner. By employing this strategy, the overall computational cost to achieve optimal statistical bounds is $O(N)$ in memory and, of particular importance

for our scope, $O(N\sqrt{N}\log N)$ in time. For further elaboration, the reader can refer to Ref. [24].

Hyperparameter tuning The process of selecting the three Falkon hyperparameters, namely M, σ, and λ, adheres to the guidelines outlined in Ref. [2]. The hyperparameter selection is conducted using data gathered under the reference working condition, and it follows the subsequent steps.

The parameter M, representing the number of centers, plays a vital role in determining the model's expressive power. To maintain sensitivity to anomalous distributions with intricate shapes, M should be as large as possible. Furthermore, based on studies in Ref. [25], [26], it is necessary for M to be at least as large as $\sqrt{N}$ to achieve statistically optimal bounds for training convergence. However, it is worth noting that smaller values of M lead to faster training times. The experiments conducted in Ref. [2] reveal that any value of M above approximately the data batch size (N_1) does not compromise sensitivity to anomalous distributions.

The Gaussian width σ is determined as the 90th percentile of the pairwise distance between reference-distributed data points. This σ is related to the resolution of the model and its ability to fit statistical fluctuations in the data. To estimate the relevant scales of the problem and strike a suitable trade-off between complexity and smoothness, we examine the distribution of pairwise (Euclidean) distances in the reference data. Subsequently, we fix σ approximately as the 90th percentile to achieve a balance between model resolution and data characteristics. It is important to note that the model (3.24) operates on an input vector x with input features that are standardized to have zero mean and unit variance on reference-distributed data, and the same standardization is applied before computing the distances.

The regularization parameter λ is carefully chosen to be as small as possible, ensuring stable training and avoiding long training times or non-numerical outputs. Several reference-distributed toy data batches are used in this study, with each batch trained against the reference sample $\mathcal{R}$. We found that increasing this parameter can typically raise the level of compatibility between $p(t|H_0)$ with a χ^2 distribution.

Chapter 4

Beyond particle physics: credit card fraud detection

Recognizing fraudulent credit card transactions is of utmost importance for credit card companies. This capability serves as a crucial safeguard to ensure that customers are shielded from unwarranted charges linked to purchases they have not authorized. By swiftly and accurately identifying such unauthorized activities, credit card companies not only protect their customers' financial well-being but also fortify the trust that customers place in their services. This proactive approach not only minimizes potential financial losses for both the customers and the companies themselves but also fosters an environment where secure and worry-free transactions can take place.

The primary goal of this endeavor is to leverage the NPLM method in a realm that extends beyond the confines of particle physics, delving into the uncharted territory of fraud detection, and uncovering the potentials it holds within this new domain.

It is important to note that this particular approach is not explicitly tailored for the task of identifying outliers. Instead, its central focus is directed towards discerning whether the distribution of transactions subjected to analysis conforms to the established norms of standard, or reference, behavior.

4.1 Data

The dataset employed in this thesis has been obtained from the Kaggle website[1]. It comprises credit card transactions conducted by European cardholders during September 2013 and presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, with the positive class (frauds) accounting for 0.172% of all transactions.

For privacy considerations, the original characteristics have not been disclosed. Consequently, attributes V1, V2, ..., V28 emerge as outcomes of a PCA transformation process. The only features which have not been transformed with PCA are "Time" and "Amount". Feature "Time" contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature "Amount" is the transaction amount.

In this section, our objective is to deepen our comprehension of the dataset by conducting a thorough examination of its weighted distributions across diverse input features.

The subsequent visual depictions (Figures 4.1 and 4.2) showcase the distributions of the input features of the dataset. The plots have been constructed with distributions displayed on a logarithmic scale. This specific choice was made to strategically enhance the visual contrast between the two classes, thereby magnifying any discernible differences between them. Each histograms is normalized to have unit area to facilitate the comparison (see A for technical details).

[1] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

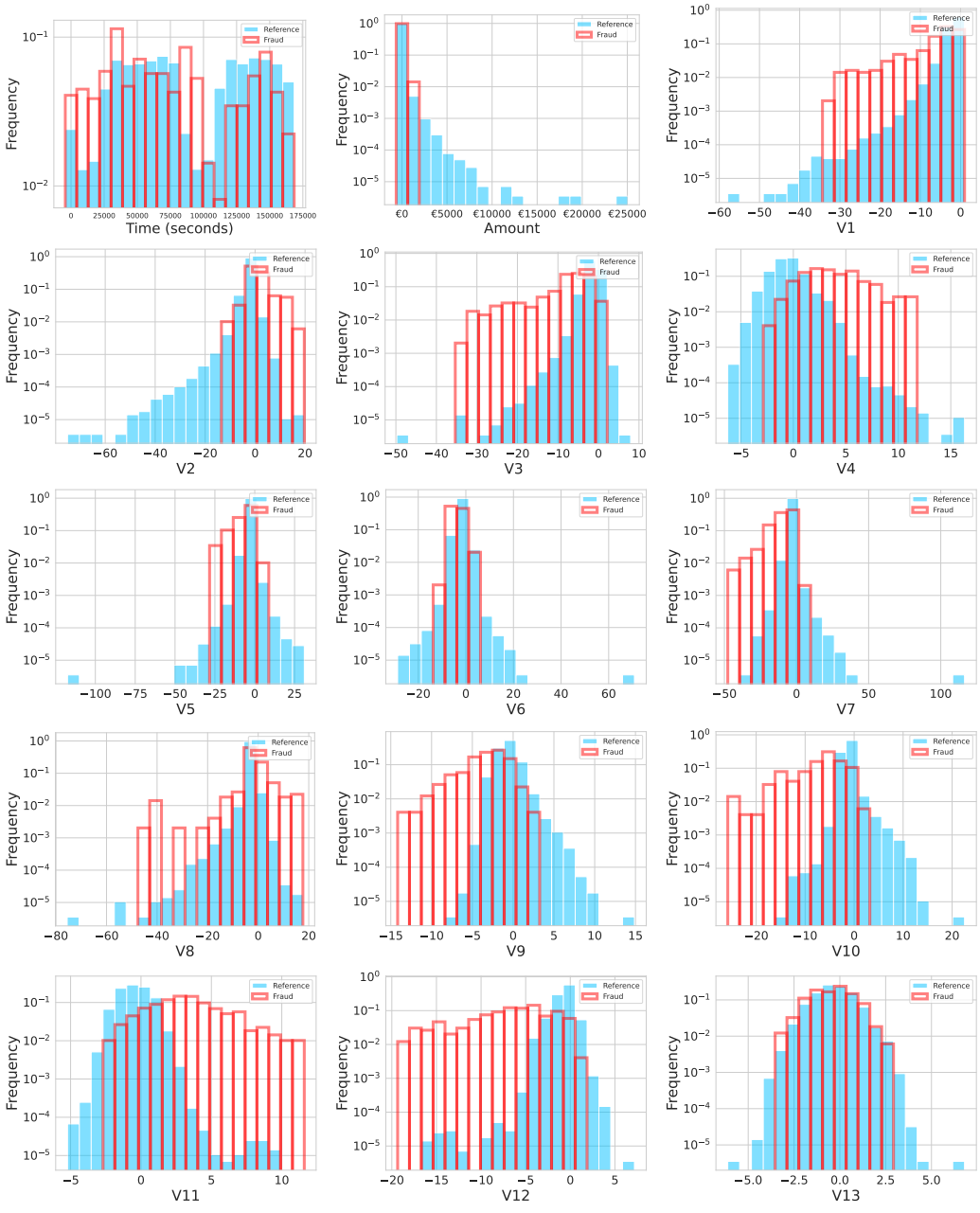


Figure 4.1: Distribution of the input features.

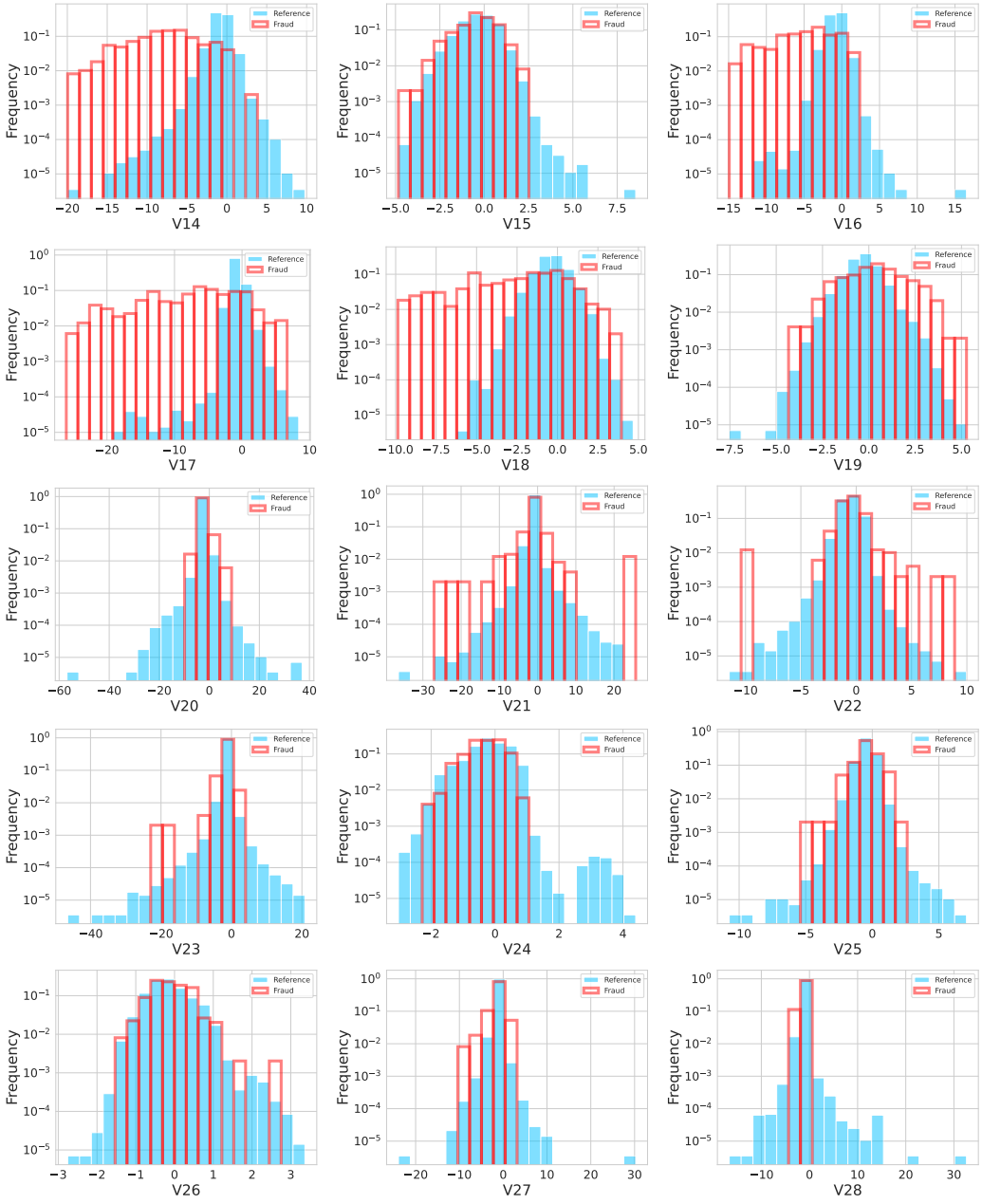


Figure 4.2: Distribution of the input features.

4.2 Results

This chapter presents the results obtained from our dataset and model. We will explore the hyperparameters of the model and assess its performance while varying the sample size and other parameters characterizing the data. We begin by discussing the preprocessing phase applied to the dataset. This step holds significant importance as it serves as a vital and advantageous stage within every machine learning approach.

Throughout this thesis, we employ simulation experiments, which referred to as "toys," to approximate the distribution of test statistics under the null hypothesis. For the null hypothesis, we generate 300 simulated experiments, and for the alternative hypothesis, we generate 100. However, given the benchmark nature of this scenario, we adopt a similar approach to estimate the distribution under the alternative hypothesis. This approach enables us to comprehensively evaluate the model's effectiveness. In a real-world context, one would typically have a single dataset associated with just one specific test statistic value.

Prepossessing The dataset consists of PCA-transformed values, which are typically standardized in advanced. However, the "Time" and "Amount" features differ from this pattern, displaying distinct averages and standard deviations. Consequently, we undertook preprocessing steps for these two features to ensure consistent inputs for our model. To achieve this, we standardize these features by subtracting the mean and dividing by the standard deviation.[2] For additional details on the preprocessing procedures, please consult Appendix B.

Tuning of the hyperparameters In the initial phase of our analysis, we concentrated on tuning the hyperparameters, specifically referred to as σ, λ, and M.

We begin by considering the kernel width σ, which we determine by fixing it at approximately the 90th percentile of the pairwise eucildean distance distribution (as thoroughly explained in the preceding chapter). This percentile value corresponds to 9.6 for the dataset considered in this work and we keep this value for our entire analysis. In this way, we estimate the typical

[2]We utilized the StandardScaler class from the scikit-learn library available at [27].

spread and variability of the data effectively. The method for choosing the candidate sigma value for this analysis is detailed in Appendix C.

We then considered the hyperparameter M which dictates the number of centers or Gaussians utilized, significantly influencing both the model's accuracy and the computational resources it demands. Previous research [20] suggests that attaining optimal statistical bounds can be achieved by setting $M \sim O(\sqrt{N})$, where N corresponds to the total number of data points.

This suggests that as the size of the dataset increases, it becomes more advantageous to increase the number of centers in order to capture the underlying patterns and variations in the data. However, it is important to consider the trade-off involved. Larger values of M can make the model more sensitive to subtle changes in the data, potentially leading to improved detection of patterns or anomalies. However, this comes at the expense of increased training times and memory usage, which may limit the scalability and efficiency of the model.

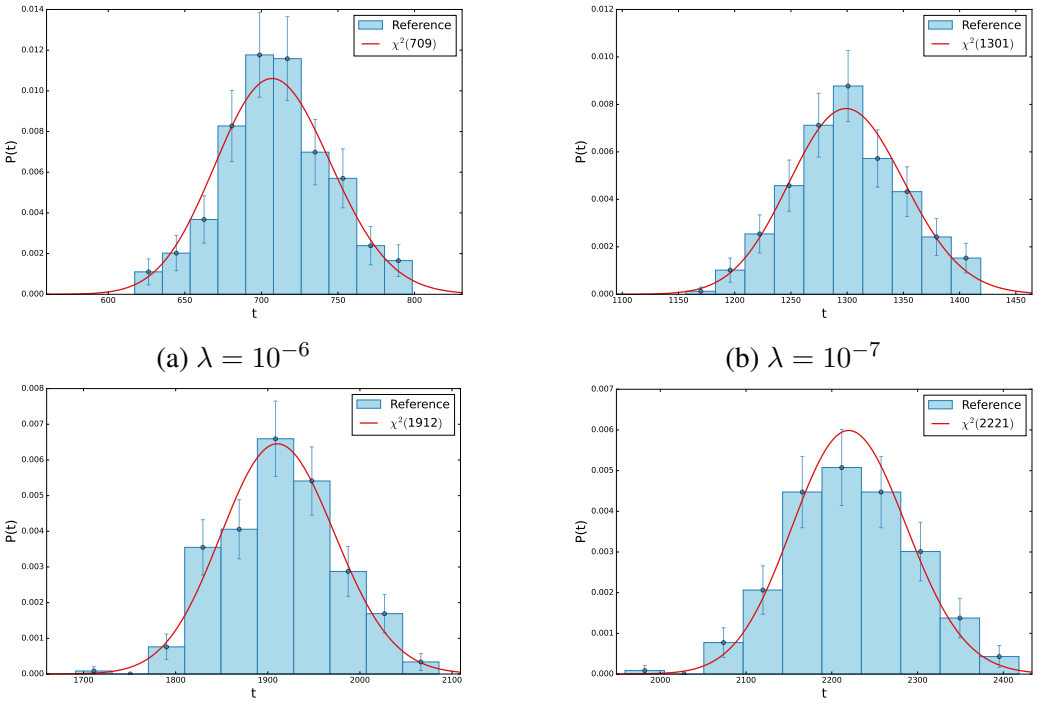
In the context of our particular study, we have maintained a constant value for M, setting it at 1500, while concurrently considering training sets of size $N = \mathcal{O}(1000)$. This choice is based on careful considerations and balancing the need for accuracy with practical constraints. By setting M to a reasonably large value, we aim to achieve a good balance between model sensitivity and computational efficiency.

As outlined in Section 3.2.1, the regularization parameter λ is deliberately chosen to be as small as possible. This choice is aimed at ensuring stable training and preventing prolonged training times. During this specific stage, we made deliberate decisions regarding certain parameter values. Notably, we set $N_0 = 3000$ and $N(0) = 1000$ (refer to Fig. 4.4). The specification of $N(0)$ and N_0 takes into account the constraints posed by the limited availability of data while guaranteeing an adequate quantity of reference data points.

We then proceeded to explore the compatibility between the chi-square (χ^2) and $p(t)$ by examining different values of the parameter λ. This investigation is aimed at identifying the value that yields the best fit between these distributions, indicating a favorable alignment between them (Figures 4.3).

We observed that increasing the value of λ helps in obtaining a better compatibility between $p(t|R)$ and a χ^2 distribution with a number of degrees of freedom that is fitted from the data themselves. However, there exists a trade-off between the utilization of this value, the resources at hand, and the time involved. These limitations are dynamic and contingent upon the distinct attributes of the dataset, available resources, and the efficiency of

time in reaching a solution. Hence, in this thesis, we have selected a λ value that strikes a balance, ensuring a strong concordance with the χ^2 distribution while maintaining a reasonable average training duration.



(a) $\lambda = 10^{-6}$

(b) $\lambda = 10^{-7}$

(c) $\lambda = 10^{-8}$

(d) $\lambda = 10^{-9}$

Figure 4.3: compatibility with the χ^2 distribution for various λ ($N_0 = 3 \times 10^3$, $N(0) = 10^3$)

The table (4.1) presents a comprehensive summary of the statistical outcomes associated with different values of λ, offering valuable insights into the model's performance under various regularization strengths. This facilitates a thorough evaluation and comparison of the results, allowing for a deeper understanding of how different regularization settings impact essential aspects, such as time, χ^2 degrees of freedom (DoF), and p-value of the Kolmogorov–Smirvon test.

The column on the average training time confirms that decreasing the value of λ will result in a longer time for the model to train, indicating a trade-off between time and model regularization, as expected. Additionally, all the p-values show a robust agreement between $p(t)$ and the χ^2 distribu-

Table 4.1: Statistical parameters for $p(t)$ with different values of λ ($N_0 = 3 \times 10^3$, $N(0) = 10^3$)

Avg training time (s)	DoF	p-value	λ
1.19	709	0.65	10^{-6}
4.10	1301	0.75	10^{-7}
16.29	1912	0.97	10^{-8}
48.93	2221	0.49	10^{-9}

tion, indicating a favorable fit. Having a variety of λ values offers important observations regarding how this parameter impacts the statistical characteristics of $p(t)$, particularly when considering a less regularized model where the DoF tend to increase. Therefore, the value $\lambda = 10^{-8}$ appears to be a suitable choice with a reasonable average training time. The compatibility is visually validated in Figure 4.4.

While maintaining compatibility between $p(t)$ and χ^2 as the data parameters undergo changes during the analyses is a crucial consideration, our investigations have consistently revealed that this compatibility perseveres across our analytical processes. Specifically, upon observing that the compatibility remains intact even when adjusting various data parameters, including N_0 and $N(0)$, while holding λ fixed at $\lambda = 10^{-8}$, we have made the decision to maintain this constant value for λ.

The last hyper-parameter in our configuration concerns the upper limit on the number of iterations for the gradient descent process. For our specific scenario, we have set this hyperparameter to a value of one million iterations, which is quite large. This choice influences the convergence and refinement of the model's parameters over the course of the optimization routine, ensuring that the algorithm performs a comprehensive exploration of the optimization landscape to attain optimal performance.

Exploring the model performance

After determining the most suitable values for the hyperparameters of the model, we studied the performance of the model while changing the parameters characterizing the dataset, such as for instance the sample size or the amount of fraudelent transactions.

Our initial focus was directed towards the parameter represented as $N(S)$.

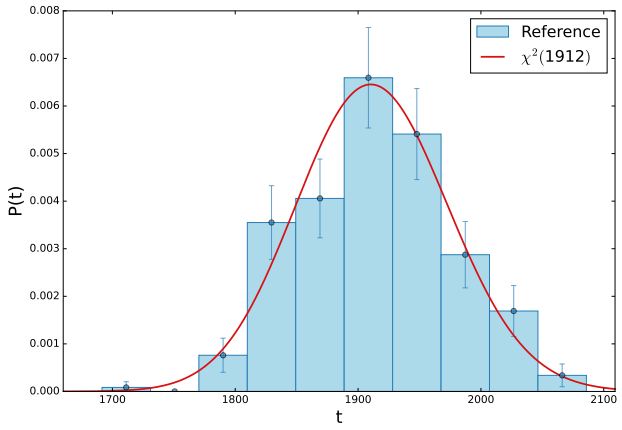


Figure 4.4: $p(t)$ of Reference Samples
($N_0 = 3 \times 10^3$, $N(0) = 10^3$)

This parameter quantifies the presence of fraudulent transactions within the data.

Before delving into the outcomes corresponding to different values of $N(S)$, it is important to note that there are different effects associated with anomaly detection using the NPLM method.

Fraudulent transactions can occur either in conjunction with regular transactions or replacing them partially. In the former scenario, the distribution of features will deviate from the baseline distribution $p(x|R)$. Additionally, the total count of transactions within a given timeframe will be impacted due to the introduction of abnormal data. Consequently, the average count of measured transactions can be expressed as $< N_1 >= N(1) = N(0) + N(S)$.This scenario is denoted as a *shape and normalization effect* and it represent the most general case.

In the second scenario, referred to as a *shape-only effect*, the expected number of transactions remains unaffected by the presence of anomalous data points. In this case, a subset of reference events within the data sample $\mathcal{D}$ (specifically, an expected $N(S)$) will be substituted with fraudulent transactions, maintaining an unchanged overall expected transaction count of $N(0)$.

The fundamental concept involves enhancing the amount of signal present in the dataset $\mathcal{D}$ to evaluate the model's ability to detect anomalies. Consequently, the model might react by recognizing and pinpointing these anoma-

lies. Nonetheless, this outcome doesn't inherently confirm that the model has truly acquired meaningful insights about the data distribution. In the first scenario described earlier, the response of the model could potentially solely stem from the increased presence of signal compared to the reference-only situation. We intend to tackle this matter through a targeted analysis.

The subsequent type of detection relates to the configuration of the distribution of references and data, while keeping the average dataset size constant (e.g., $N_0 = 3000$ and $N(0) = 1000$), as explained earlier. This concept emphasizes the idea that diverse data batches could display different distribution shapes for references and data. Although this fact marks an anomaly on its own due to the dissimilar distribution shape, it does not necessarily signify fraudulent behavior. This is due to the anticipated fluctuations in the distributions caused by finite size effects. This possibility is taken into account for when estimating the null distribution of the test statistic using reference-distributed toys, although this does not guarantee the absence of false positives.

In the rest of this chapter, we will go through the analyses outlined above. In the full case of shape and normalization effects we will explore different values for all the parameters characterizing the dataset. In the other scenarios, we will only focus on the amount if injected signal $N(S)$. Consequently, this chapter is divided into three parts: NPLM with shape and normalization effects, a test to evaluate the importance of the increase in data points due to the presence of signal, and NPLM with shape-only effects.

NPLM for shape and normalization By adjusting the value of $N(S)$, we aimed to evaluate how sensitive our model is to the injection of anomalous data in the dataset. Varying $N(S)$ allowed us to explore the level of complexity or granularity at which the model examines the data.

In our study, we observed a linear relationship between $N(S)$ and the model's effectiveness in detecting fraudulent usage as measure by the Z-score, as shown in Fig. 4.5. The bars used in all figures about the sensitivity of the model with Z-scores represent the 68 percent confidence intervals.

Before proceeding to the exploration of the next data parameter, we would like to present Figure 4.6. This figure showcases an example of the distribution of the test statistic for the null and alternative hypotheses. These distributions are depicted for specific fixed values: $N_0 = 3 \times 10^3$, $N(0) = 10^3$, and $N(S) = 50$.

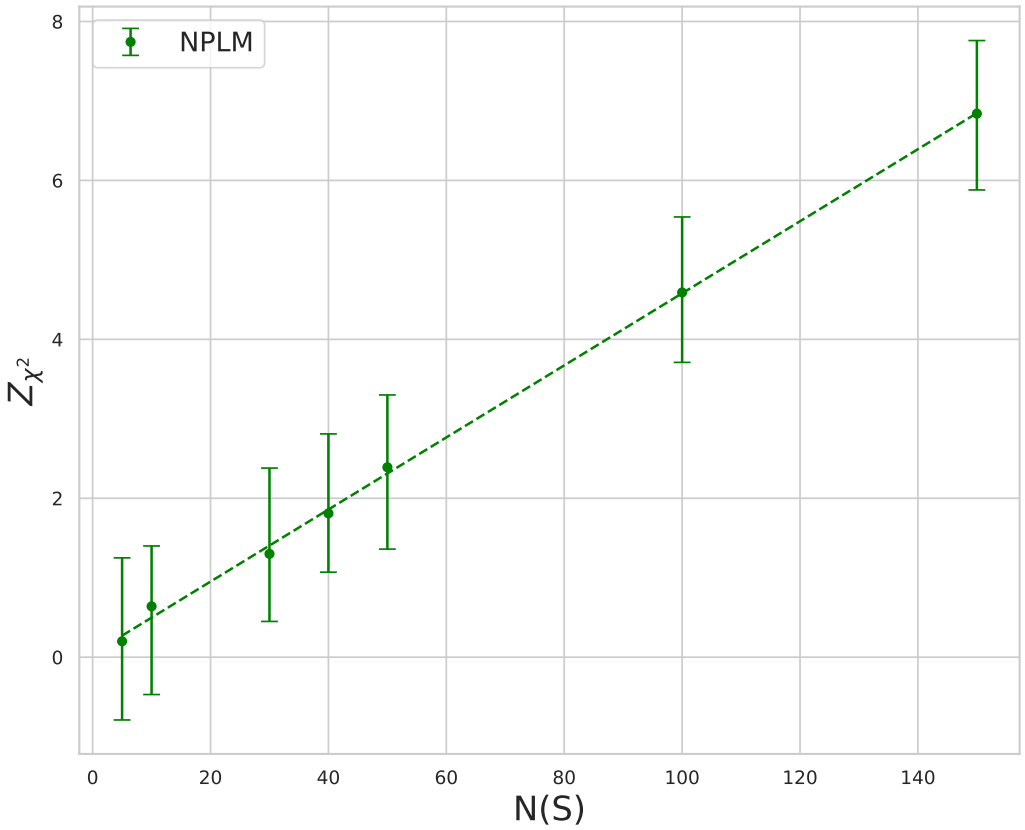


Figure 4.5: Comparing Z values for various $N(S)$
($N_0 = 3 \times 10^3$, $N(0) = 10^3$)

We now move on to the next data parameter, denoted as N_0.We monitor the model sensitivity by exploring three different ratios $N(S)/N(0) \in \{0.1, 0.3, 0.5\}$) while keeping the ratio of $N(S)$ to $N(0)$ fixed.

For a ratio of 0.1 percent, we observed (Fig. 4.7) that N_0 had minimal effect on the model's behavior. However, for the other ratios, increasing the value of N_0 resulted in a higher level of sensitivity for the model.

As already mentioned earlier, we sought to examine the compatibility between $p(t)$ and the χ^2 distribution. Specifically, our investigation focused on assessing how well these two distributions align with each other for different values of N_0, determining the degree of compatibility between them.

During our experiments, we found that the compatibility between the χ^2

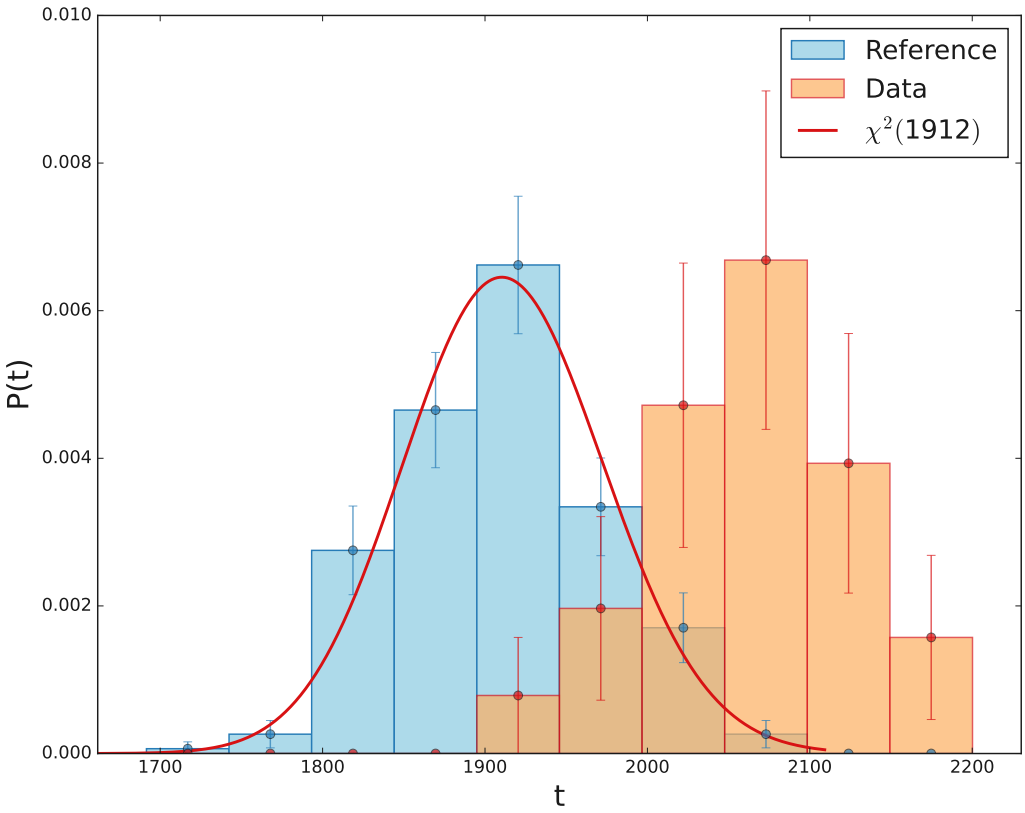


Figure 4.6: $p(t|R)$ and $p(t|H_{\hat{w}})$

distribution and $p(t)$ remained intact (see Fig. 4.8), regardless of the changes made to the reference sample size, N_0. This indicates that the relationship between these distributions was preserved throughout our analysis.

The data presented in Table 4.2 shows explicitly our findings regarding the χ^2 compatibility of $p(t)$ under the null hypothesis. Moreover, it shows that the the number of DoF *decreases* with larger and larger reference samples. This could signify that less complex models emerge from training with larger values of N_0 as the reference distribution is better represented.

Lastly, we conducted an examination of the data parameter denoted as $N(0)$. This parameter also required compatibility checks to ensure its effectiveness within the model. To carry out this evaluation, we kept the previously established ratios unchanged and proceeded to analyze the impact of varying $N(0)$ on the model's behavior. Similar to our observations

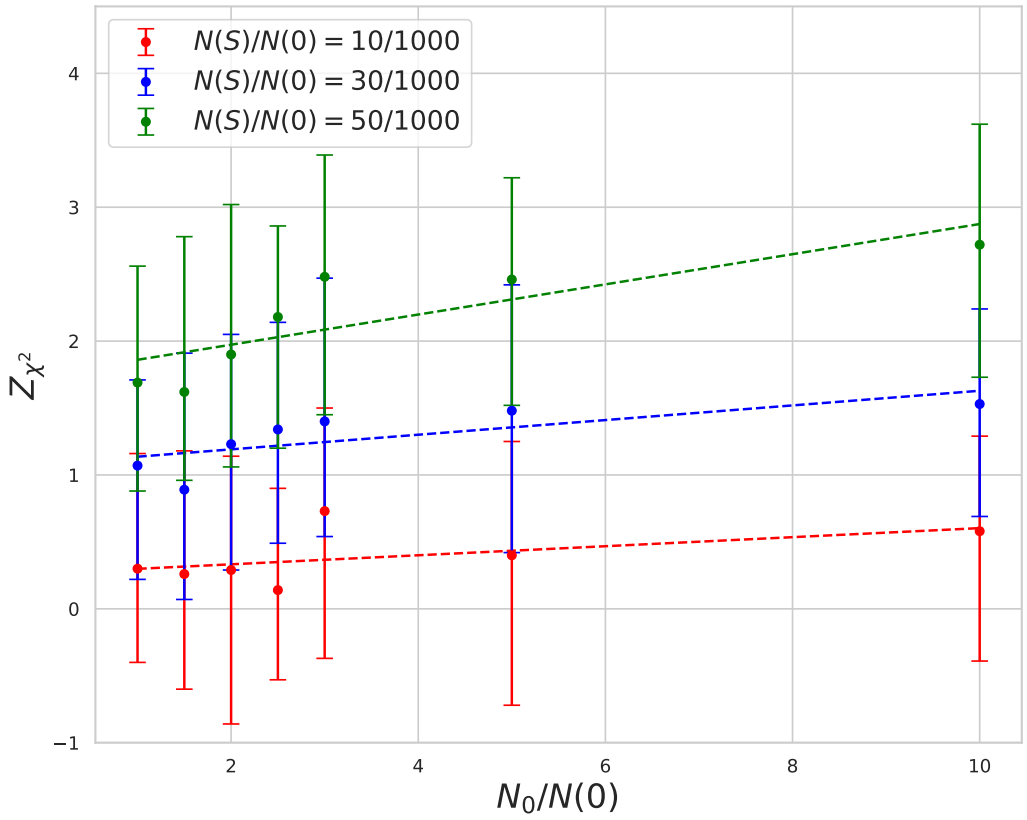
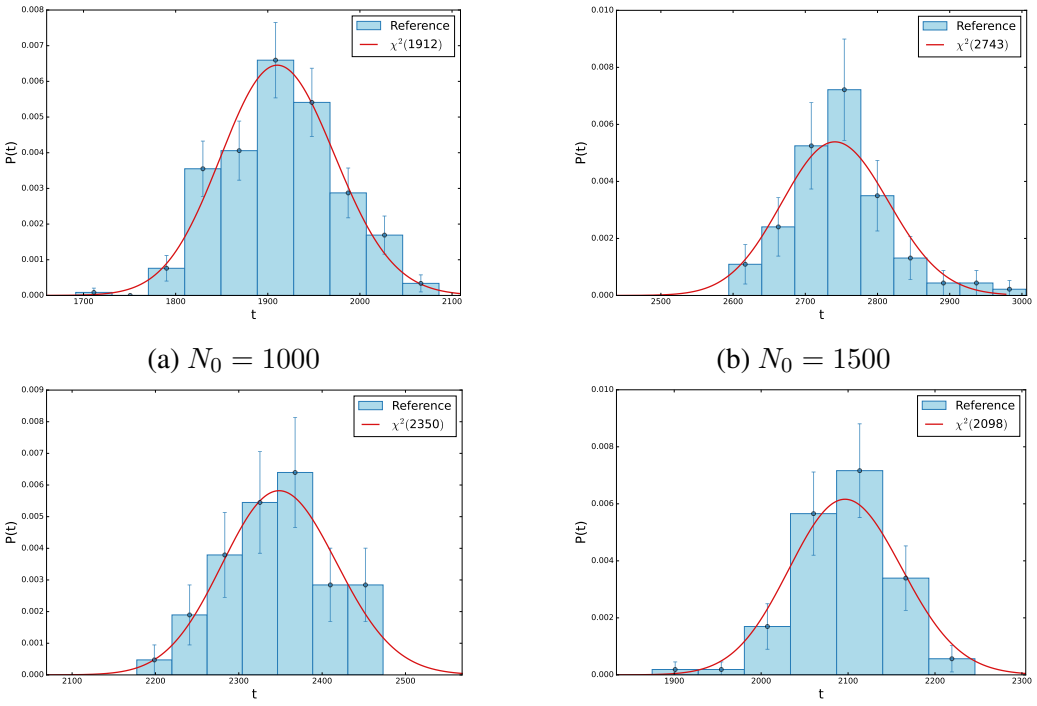


Figure 4.7: Comparing Z with different values of N_0 ($N(0) = 10^3$)

with the N_0 parameter, we found that increasing the value of $N(0)$ resulted in improved sensitivity of the model, particularly for ratios larger than 0.1 percent (see Fig. 4.9).

As in the previous analyses, we verified the alignment between the test statistics and the χ^2 distribution. The information is provided in Table 4.3.

Counting the number of events We ask here the question of whether the model is detecting more than the excess of events in the case of the full analysis. We address this problem with a comparison between the NPLM method and an specific metric that focuses on the impacts of additional signal. To facilitate this comparative analysis, we introduce a new test statistic value that enables the investigation of this particular influence



(a) $N_0 = 1000$ (b) $N_0 = 1500$

(c) $N_0 = 2000$ (d) $N_0 = 2500$

Figure 4.8: Compatibility of $p(t)$ with χ^2 distribution for various N_0 ($N(0) = 10^3$)

$$t_N = \Delta N / \sqrt{N(0)} \tag{4.1}$$

It is also necessary to define an alternative metric to evaluate performance, since the χ^2 distribution does not characterize t_N. Therefore, we utilized the empirical Z score as a measure that quantifies the extent to which an observed value differs from the mean of a distribution, expressed in terms of standard deviations.

Figure 4.11 illustrates the empirical Z values of both methods across various values of N(S). This figure strongly suggests that the NPLM method is acquiring knowledge beyond the impact of additional signal as the NPLM results are consistently above the t_N results. The confidence intervals are however quite large and overlapping, hence a more detailed analysis with potentially more data would be interesting.

Table 4.2: Statistical parameters for $p(t)$ with different values of N_0 ($N(0) = 10^3$)

Avg training time (s)	DoF	p-value	N_0
9.18	3401	0.93	1000
9.82	2743	0.46	1500
11.08	2350	0.83	2000
11.35	2098	0.66	2500
16.29	1912	0.97	3000

Table 4.3: Statistical parameters for $p(t)$ with different values of $N(0)$ ($N_0 = 3 \times 10^3$)

Avg training time (s)	DoF	p-value	$N(0)$
15.94	1414	0.52	600
13.28	1683	0.98	800
16.29	1912	0.97	1000
11.80	2119	0.99	1200

NPLM for shape-only effects In this part, we consider the NPLM method with "shape-only" effects, where the method identifies changes in the shape of the distributions, and we compare it with the full case, where it learns from the inclusion of additional signal as well, with same same amount of expected signal $N(S)$.

Figure 4.12 displays the Z values of both methods across various values of N(S). This illustration effectively underscores that the NPLM method, specifically in the context of shape and normalization, acquires insights that extend beyond the sole shapes of the distributions, resulting in enhanced performance with respect to anomaly detection within the dataset. As in the previous analysis, the confidence intervals are large and, therefore more studies would be needed to investigate this aspect in more details.

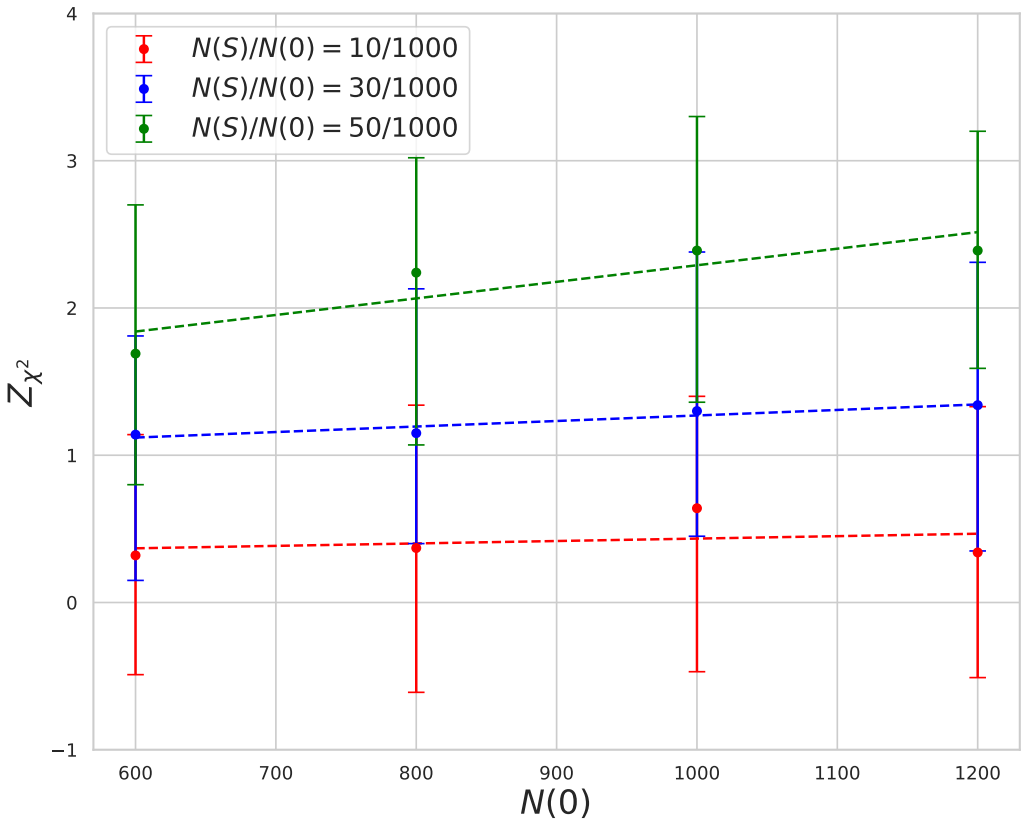
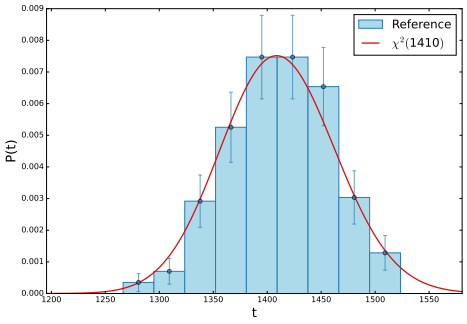
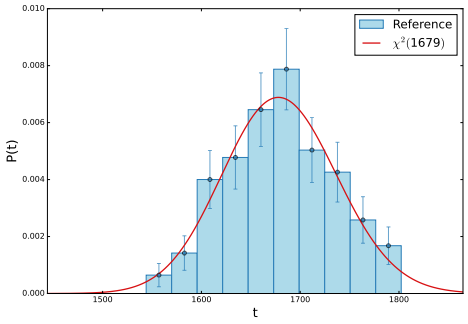


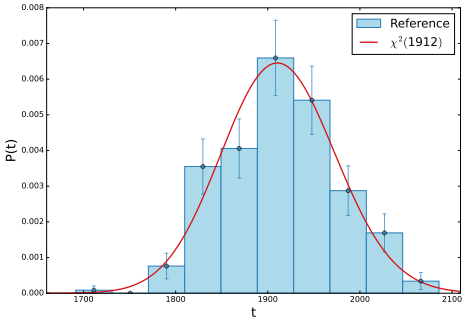
Figure 4.9: Comparing Z for various values of $N(0)$
($N_0 = 3 \times 10^3$)



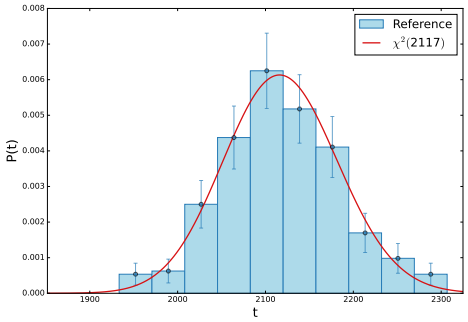
(a) $N(0) = 600$



(b) $N(0) = 800$



(c) $N(0) = 1000$



(d) $N(0) = 1200$

Figure 4.10: Compatibility of $p(t)$ with χ^2 distribution for various $N(0)$ ($N_0 = 3 \times 10^3$)

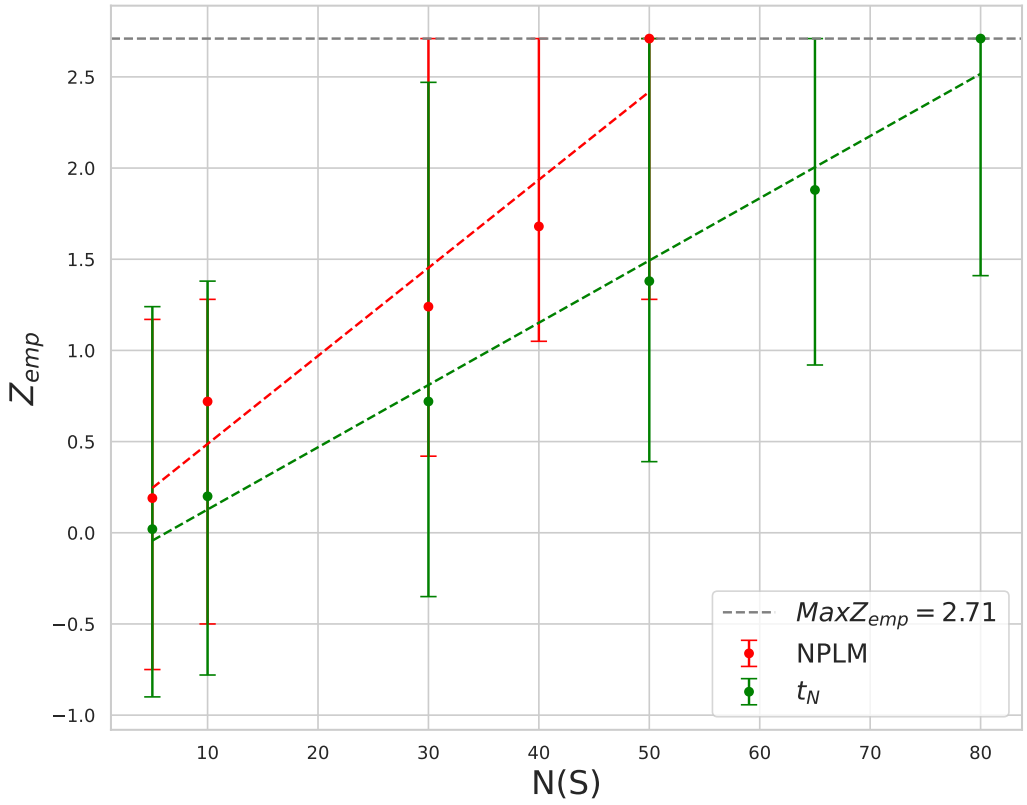


Figure 4.11: Comparison between the NPLM method and t_N
($N_0 = 3 \times 10^3$, $N(0) = 10^3$)

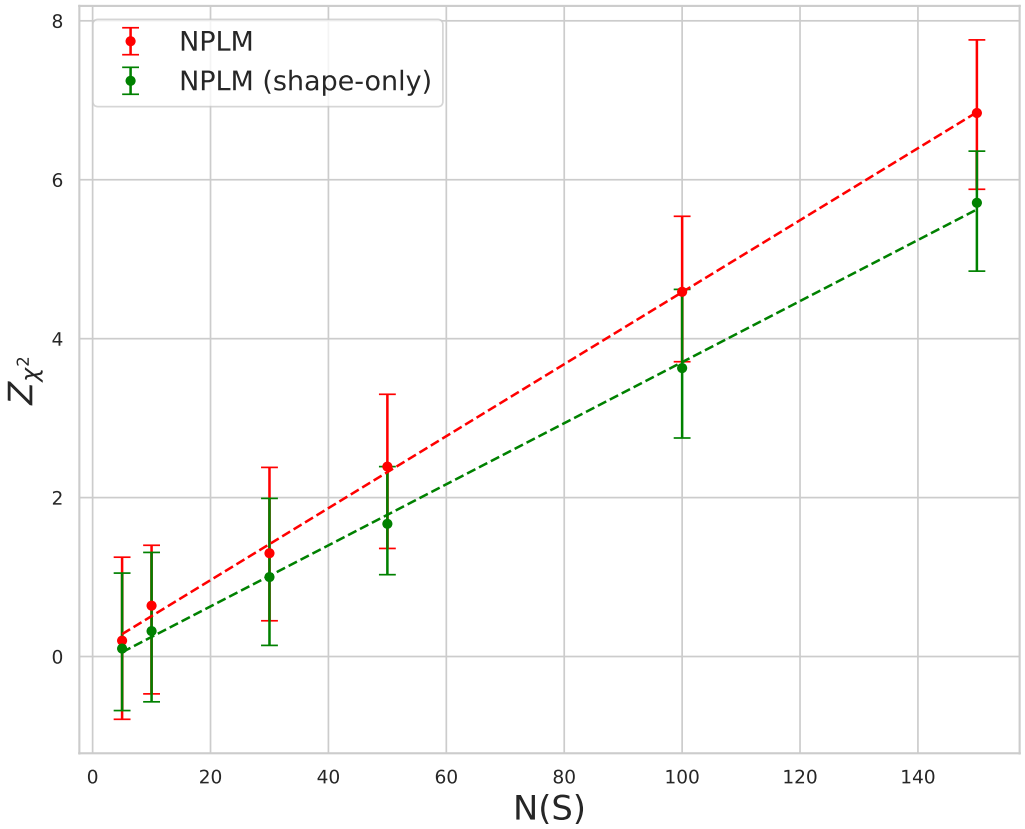


Figure 4.12: NPLM for shape-only and shape and normalization ($N_0 = 3 \times 10^3$, $N(0) = 10^3$)

Chapter 5

Conclusions

In this thesis, a recent machine learning method, known as the *New Physics Learning Machine*, developed within the context of [1, 2], has been applied to a completely new domain: the detection of fraudulent activities in credit card transactions. This approach, notable for its model-independent nature, offers a departure from the conventional path within the domain of anomaly detection. Traditional anomaly detection often pertains to identifying outliers, but the methodology we present here is distinct from that conventional trajectory. Rather, its primary objective revolves around determining whether the distribution of analyzed transactions adheres to the recognized standards of conventional or reference behavior.

The central focus of this work revolves around enhancing computational efficiency, which has been realized through the strategic utilization of state-of-the-art large-scale kernel methods, most prominently the Falkon library [20]. Employing these advanced techniques facilitates the application of efficient and expressive machine learning models while incurring minimal computational overhead.

The heart of our approach lies in training a classifier on a sample of reference data and the data of interest, thereby constructing a hypothesis testing procedure grounded in the likelihood ratio test. The assessment of data compatibility with the reference is conducted to identify indications of potentially fraudulent transactional cues. While the problem is presented in a classification context, the focus isn't on gauging classification accuracy; instead, the ultimate aim revolves around fitting the likelihood ratio and executing a hypothesis test. Consequently, the evaluation of performance entails comparing the ideal statistical significance against the observed significance.

Careful consideration is required when choosing a model. A significant portion of attention is directed towards adjusting hyperparameters, which hold a crucial position in achieving a subtle equilibrium between model intricacy and alignment with statistical distribution. This iterative tuning procedure, executed across various data sets, guarantees the strength and dependability of our method.

Our exploration of data parameters further enriches the depth of our investigation. The relationships and impacts of parameters such as the size of the reference sample or the amount of anomalous transactions, have been meticulously analyzed. By varying these parameters, we've uncovered their profound influence on our model's performance, providing valuable insights into fraud detection performance and the delicate interplay between signal identification and model behavior.

As we conclude this journey, it's evident that this machine learning strategy, coupled with thorough explorations of data parameters and hyperparameter tuning, contributes to the broader landscape of both fraud detection methodologies and the convergence of machine learning with complex real-world challenges. This work not only addresses the specific challenge of credit card fraud but also opens avenues for applying similar techniques to various other anomaly detection scenarios, where data-driven understanding is crucial. In this pursuit, we have striven not only to uncover fraudulent activities but also to uncover the potential of cutting-edge machine learning methods in unraveling hidden insights across diverse domains. This thesis serves as a remarkable illustration of how methodologies crafted for fundamental research can make meaningful contributions to society, just as machine learning provides indispensable advantages to sciences.

Looking ahead, our research lays the foundation for several promising directions of future work. One promising avenue involves conducting a comprehensive comparison between our proposed approach and alternative methods. By subjecting our approach to rigorous benchmarking against established techniques, one can gain a deeper understanding of its strengths and identify areas for refinement. Additionally, extending our study to encompass diverse datasets with carefully controlled variations in feature definitions will enhance our understanding of the method's adaptability and robustness. By manipulating feature characteristics systematically, one can disentangle their individual impacts on performance, yielding insights that contribute to more precise model design and interpretation. Embracing such avenues of investigation not only advances our current methodology but also propels the field

forward by fostering deeper insights into the intricate dynamics between features and their effects on analytical outcomes.

Bibliography

[1] R. T. D'Agnolo and A. Wulzer, "Learning new physics from a machine," *Physical Review D*, vol. 99, no. 7, p. 075016, 2019.

[2] M. Letizia *et al.*, "Learning new physics efficiently with nonparametric methods," *European Physical Journal C*, 2022.

[3] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] G. Casella and R. L. Berger, *Statistical Inference*. Duxbury Press, 2002.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013.

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

[8] M. A. Nielsen, "Neural networks and deep learning," Online resource, 2015.

[9] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio, "Toward causal representation learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 612–634, 2021.

[10] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2006.

[13] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLOS ONE*, vol. 11, no. 4, p. e0152173, 2016.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[15] L. Wasserman, *All of Statistics*. Springer, 2004.

[16] R. T. D'Agnolo *et al.*, "Learning multivariate new physics," *European Physical Journal C*, vol. 81, no. 4, p. 89, 2021.

[17] R. T. d'Agnolo *et al.*, "Learning new physics from an imperfect machine," *European Physical Journal C*, vol. 82, no. 4, p. 275, 2022.

[18] S. S. Wilks, "The large-sample distribution of the likelihood ratio for testing composite hypotheses," *Annals of Mathematical Statistics*, vol. 9, no. 1, pp. 60–62, 1938.

[19] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, "Asymptotic formulae for likelihood-based tests of new physics," *European Physical Journal C*, vol. 71, p. 1554, 2011, [Erratum: European Physical Journal C 73, 2501 (2013)].

[20] A. Rudi, L. Carratino, and L. Rosasco, "Falkon: An optimal large scale kernel method," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[21] U. Marteau-Ferey, D. Ostrovskii, F. Bach, and A. Rudi, "Beyond least-squares: Fast rates for regularized empirical risk minimization through self-concordance," in *Proceedings of the Thirty-Second Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, vol. 99, 2019, pp. 2294–2340.

[22] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *Journal of Machine Learning Research*, vol. 7, pp. 2651–2667, 2006.

[23] A. Christmann and I. Steinwart, *Support Vector Machines*. Springer, 2008.

[24] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi, "Kernel methods through the roof: Handling billions of points efficiently," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 14 410–14 422.

[25] A. Rudi, R. Camoriano, and L. Rosasco, "Less is more: Nyström computational regularization," *Journal of Machine Learning Research*, vol. 18, no. 26, pp. 1–47, 2017.

[26] U. Marteau-Ferey, F. Bach, and A. Rudi, "Globally convergent newton methods for ill-conditioned generalized self-concordant losses," *SIAM Journal on Optimization*, vol. 29, no. 4, pp. 2715–2739, 2019.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A

Weighted Histograms

Below, one will find the code snippet designed to visualize the weighted distributions effectively:

```
1
2  # Weighted Histogram
3
4  for col in data.columns:
5      if col not in ["Class"]:
6          sns.set_style("whitegrid")
7          fig, ax = plt.subplots()
8
9          # Calculate histogram bins for the specific column's data combined
10         combined_data = np.hstack((data[data['Class']==0][col],
11                                    data[data['Class']==1][col]))
12         hist, bins = np.histogram(combined_data, bins=20)
13
14         freq_G, _ = np.histogram(data[data['Class']==0][col],
15                                  weights=1/len(data[data['Class']==0][col]) *
16                                  np.ones(len(data[data['Class']==0][col])),
17                                  bins=bins)
18         freq_F, _ = np.histogram(data[data['Class']==1][col],
19                                  weights=1/len(data[data['Class']==1][col]) *
20                                  np.ones(len(data[data['Class']==1][col])),
21                                  bins=bins)
22
23         # Plotting the bars for Reference data
24         ax.bar(bins[:-1],freq_G,width=np.diff(bins)[0],
25         color='deepskyblue',alpha=0.5,label='Reference')
26
27         # Plotting the bars for Fraud data with an offset
28         ax.bar(bins[:-1],freq_F,width=np.diff(bins)[0],
29         color='none',edgecolor='red',linewidth=4,alpha=0.5,label='Fraud')
30
31         # Increase the size of ticks
32         ax.tick_params(axis='both', which='major',
33         labelsize=default_font_size * 0.8)
```

```
34
35          ax.set_ylabel('Frequency', fontsize=default_font_size)
36
37          # Set y-axis to logarithmic scale
38          ax.set_yscale('log')
39
40          # Move the legend to the upper right corner
41          ax.legend(loc='upper right')
42
43          if col == "Time":
44              # Change font size for "Time" column
45              plt.rcParams.update({'font.size': 10})
46              ax.set_xlabel("Time (seconds)", fontsize=default_font_size)
47              ax.tick_params(axis='x', which='major', labelsize=10)
48          else:
49              ax.set_xlabel(col, fontsize=default_font_size)
50
51          if col == "Amount":
52              # Format x-axis labels with Euro symbol
53              formatter = FuncFormatter(euro_formatter)
54              ax.xaxis.set_major_formatter(formatter)
55              ax.tick_params(axis='x', which='major', labelsize=14)
56
```

The provided code systematically iterates through every column in the dataset. For each column, with the exception of the Class column, the code generates a histogram plot using the matplotlib and seaborn libraries. This histogram visualization showcases data distribution for two well-defined categories: reference (Class 0) and fraud (Class 1). The script computes the bin divisions and frequency counts for both categories, modifies the plotting style to whitegrid, and establishes fresh figure and axis components for each column. Additionally, the y-axis scale in each plot is configured to apply a logarithmic representation, enhancing the clarity of the data distribution. The histogram bars that correspond to reference data are displayed in a semi-transparent blue hue, while the bars indicating fraud data are outlined in red with increased line thickness and a comparable level of transparency.

This weighting technique ensures that the frequencies derived from the histograms accurately depict the relative proportions of genuine and fraud cases present in the dataset. By assigning equal weights, each data point contributes equally to the overall frequency calculation, eliminating any potential bias that could arise from differences in sample sizes between the genuine and fraud classes. Consequently, the resulting frequencies provide a reliable representation of the distribution and relative occurrence of genuine and fraud cases within the dataset.

Appendix B

Prepossessing

Presented below is the provided code snippet tailored for preprocessing:

```
'''
Since most of our data has already been scaled,
we should only scale the columns that are left (Amount and Time)
'''

from sklearn.preprocessing import StandardScaler

std_scaler = StandardScaler()
creds['scaled_amount'] = std_scaler.fit_transform(creds['Amount'].values.reshape(-1,1))
creds['scaled_time'] = std_scaler.fit_transform(creds['Time'].values.reshape(-1,1))

creds.drop(['Time','Amount'], axis=1, inplace=True)
scaled_amount = creds['scaled_amount']
scaled_time = creds['scaled_time']

creds.drop(['scaled_amount', 'scaled_time'], axis=1, inplace=True)
creds.insert(0, 'scaled_amount', scaled_amount)
creds.insert(1, 'scaled_time', scaled_time)
```

The provided code encompasses a data preprocessing operation that leverages the StandardScaler module from the sklearn library. The central objective revolves around conducting transformations on features within the dataset. Initially, the code applies the standard scaling technique, a widely adopted method for normalization, to the amount and time columns. As a result, two fresh attributes are generated: scaled-amount and scaled-time, containing the standardized values of the original attributes correspondingly.

Following this, the code proceeds to eliminate the original time and amount columns from the dataset, signifying a refining process geared towards excluding superfluous or non-standardized data. By carrying out this

sequence of steps, the code guarantees that the numerical attributes within the dataset are rendered on a standardized scale. This standardization is particularly advantageous when working with a diverse range of machine learning algorithms, as scaled input features can lead to improved performance and convergence during model training and evaluation. This systematic approach helps mitigate the impact of varying feature magnitudes on the learning process, thereby enhancing the effectiveness of the subsequent algorithmic operations.

Appendix C

Selecting Sigma

```
1
2  def candidate_sigma(data, perc=90):
3      # this function estimates the width of the gaussian kernel.
4
5      pairw = pdist(data)
6      return round(np.percentile(pairw,perc),1)
7
8  # to estimate flk_sigma from data use candidate_sigma on a reference sample
9
10 ref_data = genuine_samples.sample(n= 30000, replace=False, random_state=123)
11 candidate_sigma(ref_data)
12
```

In the provided code, a function named candidate-sigma plays a pivotal role in estimating the width of a Gaussian kernel. This estimation process is achieved by leveraging the pairwise distances derived from a given dataset. The function operates with two principal parameters: the dataset, denoted as 'data,' and an optional percentile value referred to as perc (which defaults to 90 percent if not explicitly specified). By engaging in distance calculations between individual data points, the function generates an output value that is rounded to represent the designated percentile among these computed distances.

This function's operation entails the evaluation of how spread out the data points are, a characteristic crucial for Gaussian kernel applications. By considering the percentile of distances, the function provides insights into the data's distribution within the dataset.