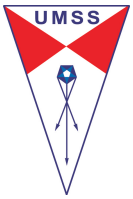
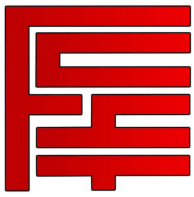


Mayor de San Simón University
Faculty of Science and Technology
Systems Engineering Degree



Driving Anomaly Detection

Degree project submitted in compliance with the requirements
to opt for the Degree in Systems Engineering

Presented by:

Evelyn CUSI LÓPEZ

Advisor:

Ph.D. Eduardo DI SANTI

COCHABAMBA - BOLIVIA
March, 2020

Acknowledgment

To my professor: Eduardo Di Santi, who gave me his valuable and selfless orientation and guidance in the preparation of this work, for his patience, support and ideas that motivated this research; to whom I owe much of my learning and my taste for the Artificial Intelligence's area.

To my family, for their constant understanding and encouragement, in addition to their unconditional support throughout my studies.

To my friends, who in one way or another supported me in carrying out this work.

To the Faculty of Science and Technology and the Mayor de San Simon University, for open me their doors and being the home of my training throughout my university career.

To my mother, for being with me, for teaching me to grow and to get up, for supporting and guiding me, for being the base that helped me get here.

Contents

List of Figures

List of Tables

Abstract

This paper describes the development of a mechanism to detect driving anomalies, which is implemented using a mobile device and Machine Learning techniques.

The objective is create a tool capable to find anomalous behaviors in the driving of human or autonomous agent, having a previous knowledge of normal driving conducts of them. Likewise it presents a background of driving anomalies detection's researches around the world, the driving parameters obtained by the mobile device are analized, and a proposal is presented to identify anomalies through the use of Neural Networks and Isolation Forests, a method of Machine Learning that is commonly used for anomalies' detection.

The work has two main parts: a model adjusted to the normal driving behavior of an agent, and a method of detecting anomalies, which were iteratively trained with 30,000 samples, which correspond only to normal driving behavior.

The detection accuracy of the complete mechanism proposed in this document is 67.68%, which was evaluated with 44040 samples, of which 164 correspond to anomalous samples, thus being one of the most outstanding contributions for the detection of driving anomalies with a semi supervised approach.

Chapter 1

INTRODUCTION

This document describes the development of a method for detecting anomalies in car driving. The use of Machine Learning techniques is proposed to generate a mechanism that identifies driving anomalies, so that they can be used to promptly alert agents and thus correct their driving behaviors.

The main idea is to generate a model that learns the normal driving behavior of a specific agent, to later autonomously detect those unexpected behaviors and report them as anomalies, so that a traffic accident can be avoided or the effects of it reduced.

1.1 Approach of the problem

Due to the serious consequences they cause on people and the high economic costs associated with them, traffic accidents are classified as a global social and public health problem.

According to the World Health Organization (WHO) each year there are approximately 1.25 million deaths due to traffic accidents, adding that half of all these victims are pedestrians, cyclists and motorcyclists (See Figure 1.1 page 2). It can also be said that they are one of the most important causes of death in the world, and the main cause of death among people between the ages of 15 and 29.

On the other hand, according to the Cochabamba Transit Operating Unit, the accidents recorded in 2017 caused the death of 200 people and left approximately 2200 injured.

Figure 1.2 shows the causes for which a traffic accident is caused, it can be seen that a large part of these are due to the human factor, however there are others that involve environmental and mechanical factors, so it is impossible completely avoid them.

That is why it is necessary to have mechanisms to prevent and/or act in a timely manner against possible traffic accidents, which is why this work focuses on studying driving behaviors, in order

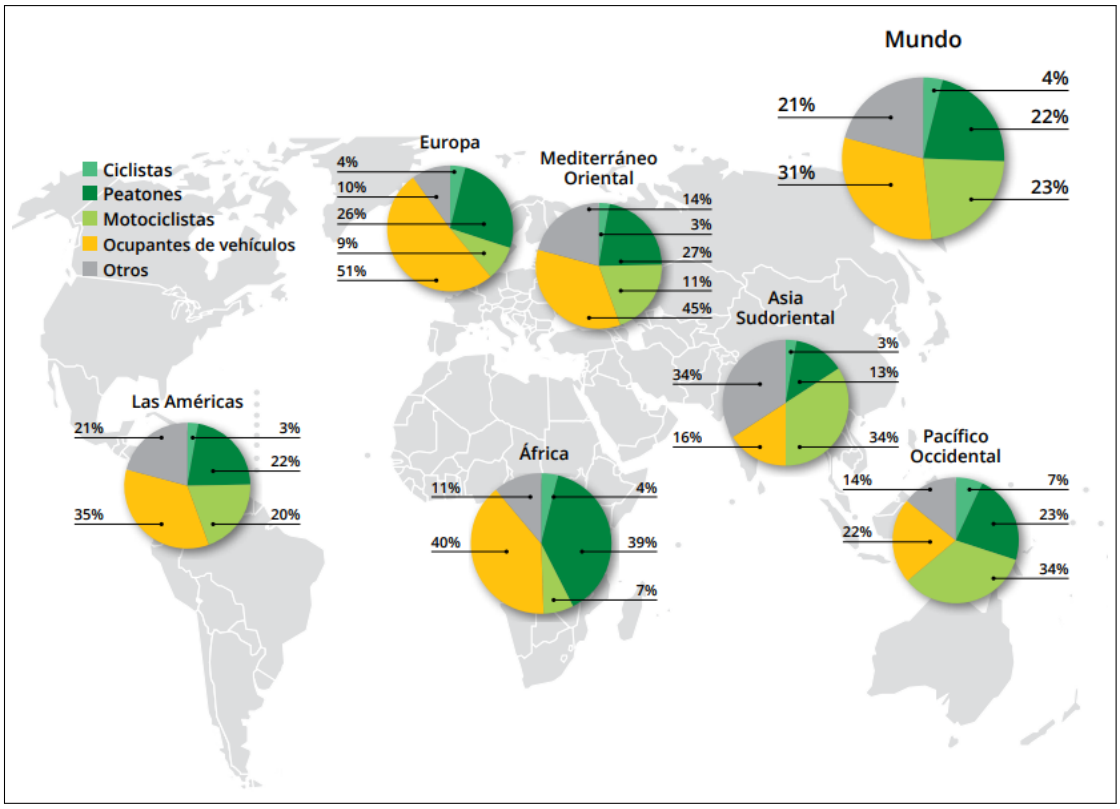


FIGURE 1.1: Deaths due to traffic accidents by region depending on user's type (de Salud (OMS), n.d.).

to generate alerts when finding a behavior anomalous in driving, so that the effects of it can be avoided or in any case minimized.

1.2 General objective

The general objective of the present work is to develop a mechanism for detecting driving anomalies, through the use of a mobile device and Machine Learning algorithms, in order to alert in a timely manner the finding of an anomalous driving pattern, such as tiredness, drunkenness, or health problems, eg. epilepsy.

1.3 Specific objectives

- Capture the driving parameters of a driver by using sensors of a mobile device.
- Scale parameters using data pre-processing techniques.
- Generate a Machine Learning model that adjusts to normal driving behavior.
- Define an anomaly detection method to generate an abnormal driving alert.

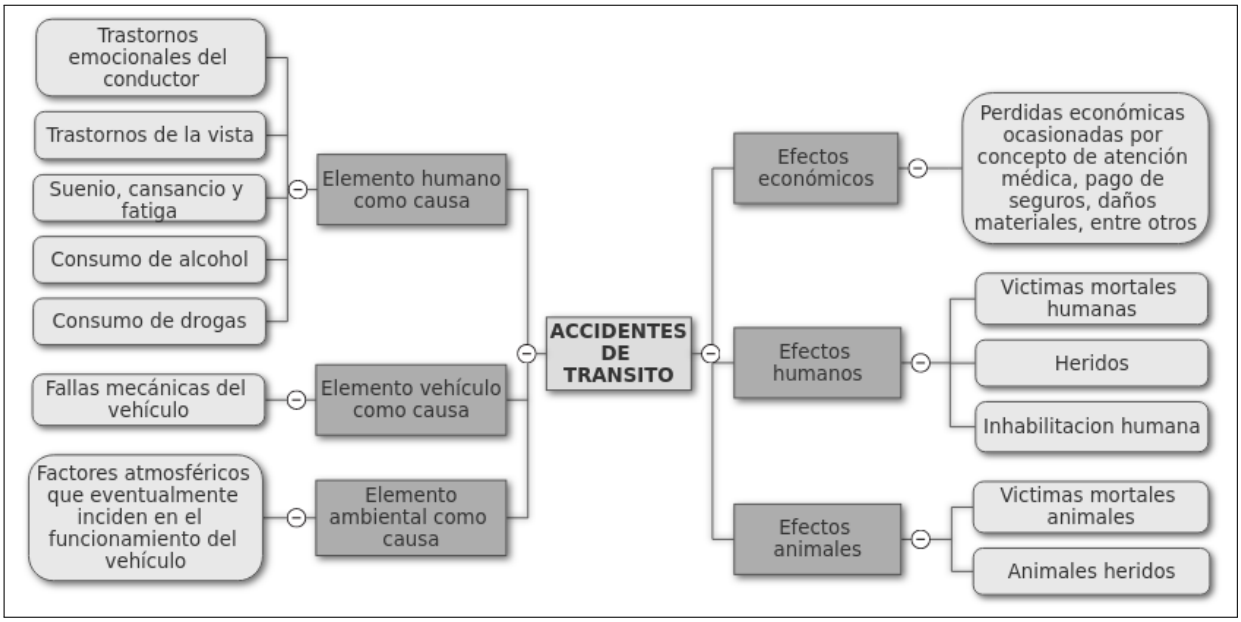


FIGURE 1.2: Problem tree (Own elaboration).

- Evaluate the anomaly detection method with new samples

1.4 Justification

Traffic accidents charge an unacceptable number of victims each year, especially in the poorest regions of the world. This is due to various aspects, but the main one lies in the low level of citizen awareness that exists, which leads to many people driving under the influence of alcohol, with excessive speed, manipulating their mobile devices, among others. For this reason, this work seeks to establish patterns of driving behaviors through the use of a mobile device and Machine Learning techniques, so that it is possible to detect timely driving anomalies.

1.4.1 Practical justification

Detect driving anomalies allows the authorities or agents to generate a timely alert so that they can correct their driving behaviors quickly. This way we can avoid traffic accidents or minimize their effects, thus reducing the amount of damage, both material and personal.

1.4.2 Methodological justification

The study carried out in the development of this research allows highlighting the efficiency of Artificial Intelligence techniques in detection of anomalies.

1.5 Limits and scope

Due conducting field tests for this investigation is quite dangerous, the examples of anomalous driving were limited to:

- Dry brakes.
- Turn right and left at high speed.
- Turn abrupt zig zag.

Thus, experiments and tests were performed only on a small set of anomalous samples, therefore it is not expected that the proposed detection model will work correctly on those examples that were not considered.

1.6 Research method

The present study was conducted with an experimental approach, with the following hypothesis:

Is it possible to detect driving anomalies by using a mobile device and Machine Learning algorithms?

Chapter 2

FUNDAMENTALS OF THE DETECTION OF ANOMALIES

This chapter discusses necessary concepts that are needed to understand the detection of anomalies, as well as different projects and investigations carried out in field of the detection of driving anomalies to date.

2.1 Anomaly detection

To understand what anomaly detection implies, it is necessary to assimilate what an anomaly is, and in what ways these can occur. Therefore, it can be said that anomalies, or outliers, are patterns in the data that do not fit a well-defined notion of normal behavior.

Anomalies can be classified into one of the following three categories:

1. **Point anomalies:** Point anomalies are simply unique and anomalous instances within a larger data set, that is, they are separated from the rest of the data. For example, in Figure 2.1, points o_1, o_2 and region O_3 are outside the limits of normal regions (N_1 and N_2), and therefore are point anomalies because they are different from the normal data set.

   This type of anomaly is considered the simplest and is the focus of most research focused on anomaly detection.

2. **Contextual (or conditional) anomalies:** These are points that are only considered anomalous in a specific context. The notion of this context is induced by the structure in the data set and must be specified as part of the problem formulation.

   These types of anomalies have been explored more frequently in time series and spatial data. Figure 2.2 shows an example of a time series of the monthly temperature of an area in the last 5 years, it should be taken into account that the temperature at time t_1 is the same as at time t_2, but occurs in a different context , therefore t_2 is considered an anomaly.

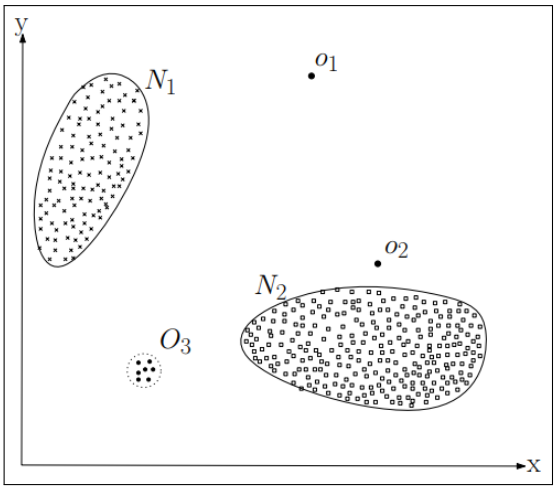


FIGURE 2.1: Example of point anomalies in a 2-dimensional data set (Varun & Arindam, 2009).

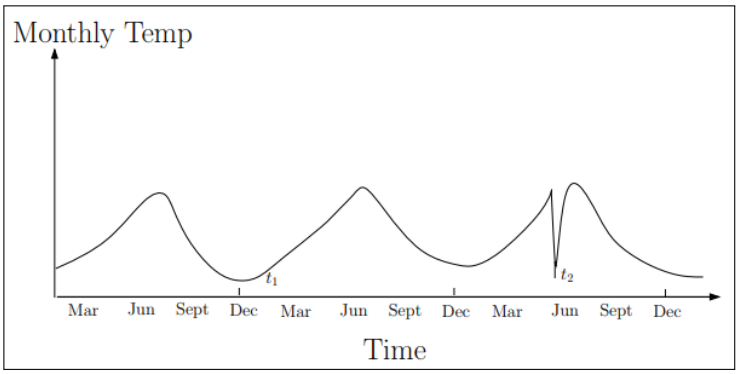


FIGURE 2.2: Contextual anomaly t_2 in a temperature time serie (Varun & Arindam, 2009).

3. **Collective anomalies:** If a collection of related data instances is anomalous with respect to the entire data set, it is called a collective anomaly. The instances of individual data in a collective anomaly may not be anomalies by themselves, but their joint appearance as a collection is anomalous.

   Figure 2.3 illustrates an example showing a human electrocardiogram output, it can be noted that the region highlighted in red denotes an anomaly because there is the same low value for an abnormally long time (corresponding to a premature atrial contraction). It should be taken into account that this low value by itself is not considered an anomaly.

In relation to the above, it can be defined as detection of anomalies, or outliers, to the identification of data points, elements, observations or events that do not fit the expected pattern of a particular group.

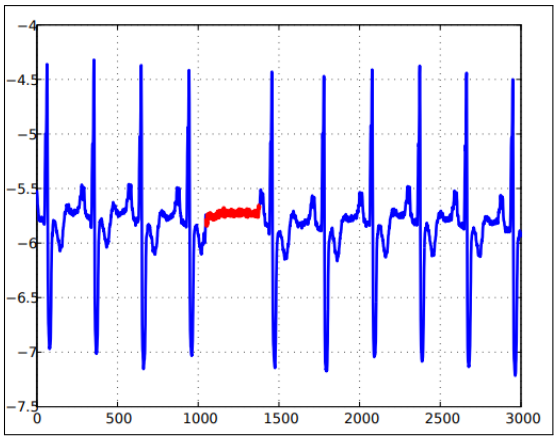


FIGURE 2.3: Collective anomaly corresponding to premature atrial contraction in a human electrocardiogram (Varun & Arindam, 2009).

Anomaly detection is used in different application domains, for example: image processing, card fraud detection, network intrusion detection systems, etc.

2.2 Challenges in anomaly detection

On an abstract level, anomaly detection may seem like a simple task. However, it can be a very challenging task. Below are some of these challenges.

- The definition of normal regions is quite difficult. In many cases, the boundaries between anomalies and normal data are not accurate. Therefore, normal observations could be considered anomalies and vice versa.
- Therefore, normal observations could be considered anomalies and vice versa.
- Most of the time the approaches for detecting anomalies in a specific field cannot be used in another field.
- The low availability of positive samples (anomalies) for training and validation of anomaly detection model.

2.2.1 Anomaly detection approaches

The approaches that can be used for this purpose are classified into following categories:

Supervised anomaly detection

The use of supervised learning techniques requires the availability of a set of labeled training data, for both normal and anomalous classes. The main focus is to build a predictive model for normal classes vs. anomalies, then take any instance of unseen data, compare with model and determine to which class it belongs.

There are two main drawbacks that arise with the use of this technique.

- The number of anomalous instances is much lower than that of the normal instances, which creates an imbalance of class distribution during training.
- Obtaining accurate and representative labels, particularly for anomaly class, is a challenge.

Semi-supervised anomaly detection

These techniques require a training set with labeled instances, but only for the normal class, this makes its use more applicable than the supervised techniques, since no labels are required for the anomaly class.

The typical approach used in these techniques is to build a model for the class corresponding to normal behavior, and use model to identify anomalies in the test data.

Unsupervised anomaly detection

Techniques that operate in an unsupervised manner do not require training data, which is why they are the most widely used. These techniques assume that normal instances are much more frequent than anomalies in test data, in case this assumption is not true, such techniques suffer from a high rate of false alarms.

2.3 Related work

The identification of abnormal driving behaviors is an indispensable part of improving driving safety, however, as previously described, this is not a simple task. In recent years, several techniques have been proposed to detect driving behaviors. This section is dedicated to review them.

Dang-Nhac et al. (2018), propose a combined system that consists of two modules: one to detect the type of vehicle of users and the other to detect events of instant driving, regardless of the orientation and position of smartphones, this system It achieves an average accuracy of 98.33% in detection of vehicles's type (car, motorcycle, bicycle, among others) and an average accuracy of 98.95% in recognition of motorcyclist driving events when using Random Forest as a classifier.

On the other hand Ferreira et al. (2017) present a quantitative evaluation of 4 Machine Learning algorithms (Bayesian Network BN, Artificial Neural Network ANN, Random Forest RF and Support Vector Machine SVM) with different configurations, applied in the detection of 7 types of driving events, between events normal and aggressive, using data collected from 4 Android smartphone sensors (accelerometer, linear acceleration, magnetometer and gyroscope); resulting in the gyroscope and accelerometer being the best sensors to detect driving events and that

Random Forest (RF) is by far the best-performing Machine Learning Algorithm, followed by the simplest form of ANN the Multi Layer Perceptron (MLP).

Johnson and Trivedi (2011) propose the MIROAD system which shows that Dynamic Time Warping (DTW) is a valid algorithm to detect potentially aggressive driving maneuvers, where almost all aggressive events (97%) were correctly identified, using the set of T sensors (accelerometer, gyroscope and tone of voice). Likewise, in the work of Kridalukmana, Yan-Lu, and Naderpour (2017), a system focused on developing driver awareness through notifications in critical situations that can trigger unsafe driving maneuvers is proposed, using a model to detect dangerous situations based on Object-Oriented Bayesian Network (OOBN).

As well as the works presented previously there is a large number of works (Bhoyar, Lata, Katkar, Patil, & Javale, 2013; Chen, Yu, Zhu, Chen, & Li, 2015; Eren, Makinist, Akin, & Yilmaz, 2012; Boonmee & Tangamchit, 2009; Koh & Kang, 2015) that use smart phone sensors (accelerometer and gyroscope) for aggressive driving detection, due of advantage of not buying or installing any device and besides being highly portable, however it depends a lot on the performance of GPS receiver and is not applicable in areas not available for GPS.

There are also other approaches to detection of aggressive driving of a driver, for instance in the work of Who-Lee, Sik-Yoon, Min-Song, and Ryoung-Park (2018), a method based on Convolutional Neural Network (CNN) is proposed) to detect the emotion of aggressive driving, by using a driver's facial images obtained with a NIR light camera and a thermal camera.

2.4 Focus on the problem

It is clear that this topic was extensively researched and that it has a wide variety of solution proposals, however most of these are based on detection by supervised learning techniques, which presents the great disadvantage of requiring data labeled to generate the model detection; In addition, much of related work proposes generalized models for detection and not specific models for each agent, which is crucial because each agent has individual driving behaviors and different conditions driving, that is, the driving of an agent that circulates through paved avenues will be different from driving of an agent that circulates through cobbled streets or driving of an agent that circulates through avenues or busy streets will be different from that of the agents that circulate through relatively decongested streets.

The proposal made in this work is intended to provide a prototype of a tool that helps analyze the sequences of motion sensors of a mobile device and allows detecting anomalies from information obtained from this analysis.

To achieve this goal, the data set of motion sensors of a smartphone is captured, using a mobile application, then data is divided and prepared with data pre-processing techniques.

Once these phases are completed, a model is trained with the training set and validated with the development set. Finally, an optimal model and a technique are chosen to classify outliers, in order to cover a full range of normal behaviors and exclude anomalies as accurately as possible. This method can best be seen in Figure 2.4.

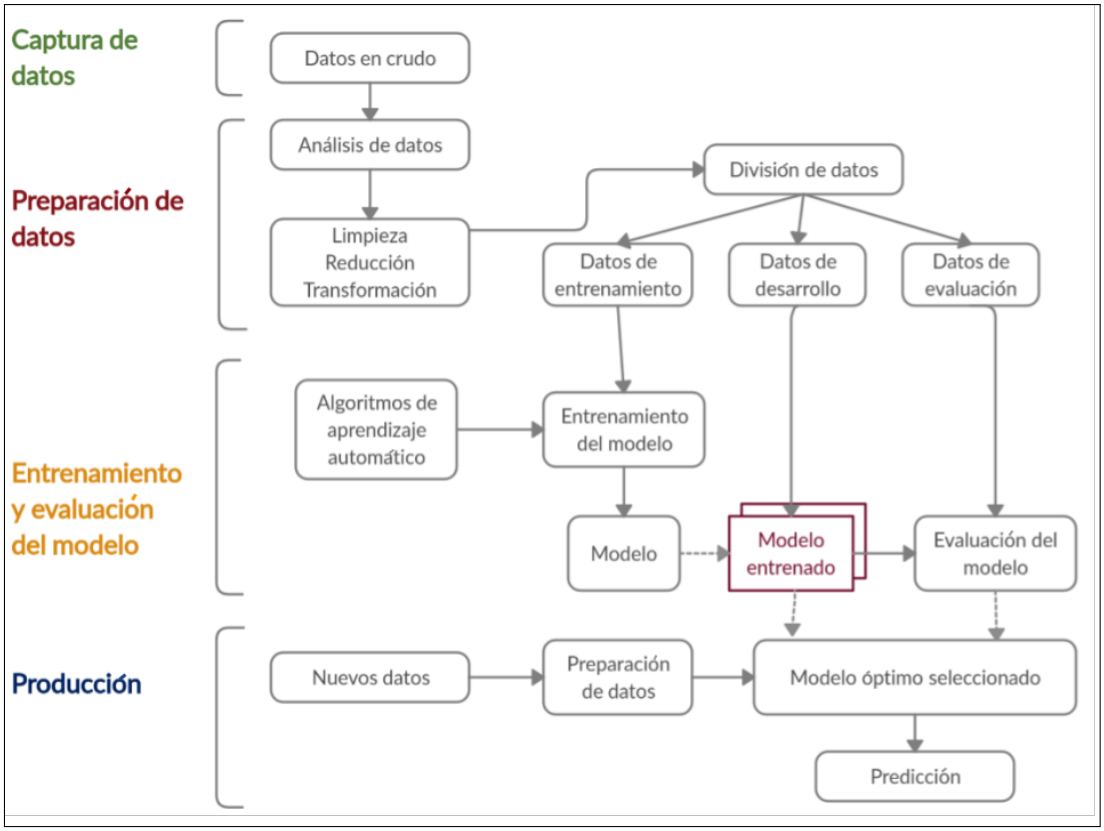


FIGURE 2.4: Proposed anomaly detection method (Own elaboration).

Chapter 3

MACHINE LEARNING FOR ANOMALY DETECTION

The term Machine Learning refers to the automatic detection of significant patterns within a data set (Shai & Shai, 2014). In recent decades it has become a common tool in almost any task that requires the extraction of information from a large amount of data, which is why it has become one of the fastest growing areas of information technology.

Although Machine Learning can solve some problems that are solved with traditional algorithms, it has overcome these problems such as image recognition, voice, language, writing, games, robotics, data analysis, time series analysis, etc. From this perspective, it is expected that through the application of Machine Learning a model can be generated that fits to a normal behavior expected for the agent.

Therefore, this chapter details the theoretical bases necessary to address the development of the method for driving anomaly detection. First, the different learning paradigms that exist and the different model approaches are described, then, a description of the functioning of neural networks is made and the different types of networks are shown, as well as different techniques of anomaly detection that exist and finally shows different evaluation metrics presented by machine learning models.

3.1 Supervised Learning, Unsupervised Learning and Semi Supervised Learning

There are several ways to classify learning paradigms that exist, however in this work only supervised, unsupervised and semi-supervised will be treated.

3.1.1 Supervised Learning

Supervised Learning is one that has input variables (X) and an output variable (Y), this type of learning uses an algorithm to learn the mapping function from input to output.

$$Y = f(X) \tag{3.1}$$

The objective of this type of learning is to approximate the mapping function so that when you have new input data (X) you can predict the output variables (Y) for that data.

This type of learning addresses two types of problems: classification and regression. Classification problems are those where the output variable is a category, such as: ”Red”, ”Blue”, or ”Healthy”, ”Sick”, on the other hand in Regression problems the output variable is a real value, such as: ”price” or ”height”. Some of the most common types of problems built on classification and regression include recommendation and prediction of time series.

3.1.2 Unsupervised Learning

On the other hand, Non-Supervised Learning is one where there is only input data (X) and there are no corresponding output variables, its main objective is to model the structure or underlying distribution in data to learn more about them.

As for learning problems without supervision, they can be grouped into two: grouping and association. **Grouping** is one where you want to discover the groupings inherent in data set, such as grouping customers by purchasing behavior. On the other hand, **Association** is one that wishes to discover rules that describe large portions of its data, for example, people who buy X also tend to buy Y. Some of the most popular unsupervised learning algorithms are: Kmeans (for clustering problems) and Apriori algorithm (for learning problems of association rules).

3.1.3 Semi Supervised Learning

Finally, there is Semi-supervised Learning, which covers those problems where there is a large amount of input data (X) and only some of data is labeled (Y). These types of problems are between supervised and unsupervised learning, it is also important to point out that many of Machine Learning’s problems in the real world are in this area, this is because it is expensive or it may take a long time to label the data set, while unlabeled data is cheap, in addition to being easy to collect and store. These types of problems can use a combination of supervised and unsupervised techniques to be solved.

Since Supervised Learning methods require a large amount of labeled training data, it is important to clarify that the collection of negative samples (abnormal driving) is difficult and risky for this particular study; In addition, the supervised approach has a potential limitation, which is: the detection of new atypical patterns, this because the resulting model is only trained to recognize a limited set of anomalous patterns, so at the time a new one is presented pattern this model will be unable to recognize it.

On the other hand, unsupervised approach has advantage of not requiring tagged information, however it often suffers from high false alarm rates and low detection rates (Xue, Shang, & Feng, 2010).

In many applications, including the one of present study, normal samples are easy to obtain, while anomalous ones are quite difficult to obtain, consequently, for implementation of this study, the application of the Semisupervised approach has been chosen. Thus, as mentioned in Chapter 2, Semi-supervised anomaly detection approach only has normal samples in the training set; that is, information about anomalies cannot be obtained, therefore unknown samples are classified as outliers, as long as their behavior is very different from that of normal samples already known.

As mentioned in this section, all of these learning approaches are based on generating a Model capable of helping either classification, grouping, etc; However, there is more than one type of model. The following section will detail in detail the different types of models that exist.

3.2 Generative and Discriminative Models

When using Machine Learning there are two main approaches to understand (model) the real world and make decisions. These two approaches are discriminative and generative models. More formally the generative and discriminative models represent two different strategies to estimate the probability that a particular object belongs to a category (Hsu & Griffiths, 2010).

Discriminative models are based on conditioned probability $P(Y|X)$, that is, they learn a direct map of a set of characteristics *X* to labels of *Y* classes. These types of models try to model simply depending on observed data (set of data), they also make less assumptions about distributions; however, they depend largely on quality of data. Some examples of discriminative models are: Logistic Regression, SVM (Support Vector Machine), Neural Networks, Random Forest, among others.

On the other hand the **generative models** point to a complete probabilistic description of data set, its objective is to develop joint probability distribution P(X, Y), either directly or by calculating $P(Y|X)$ and $P(X)$, then infer conditional probabilities required to classify new data. These models help to specify the uncertainty of a model, some examples of generative models are: Gaussian Mixture Model, Hidden Markov Model, Restricted Bolzmann Machine, Generative Adversial Networks (GAN), among others.

Discriminative models have been at the forefront of Machine Learning's success in recent years, since these models make predictions that depend on a given input, although they cannot generate new samples or data, so in the present study preference will be given to use of discriminative models.

Next, we will review the fundamental theoretical bases of some Semi-Supervised Learning techniques with a discriminative approach, to later detail what type of algorithms will be applied in the method proposed in this research work.

3.3 Artificial neural networks

The Artificial Neural Network or ANN[1] is a paradigm of information processing inspired by the way in which biological nervous system processes information. It consists of a large number of highly interconnected processing elements (neurons) that work in unison to solve a specific problem.

3.3.1 Neurons or nodes

Biological neurons (nerve cells) are fundamental units of brain and nervous system. Neurons are the cells responsible for receiving sensory information from external world through dendrites, processing it, and exiting through the axon (See Figure 3.1).

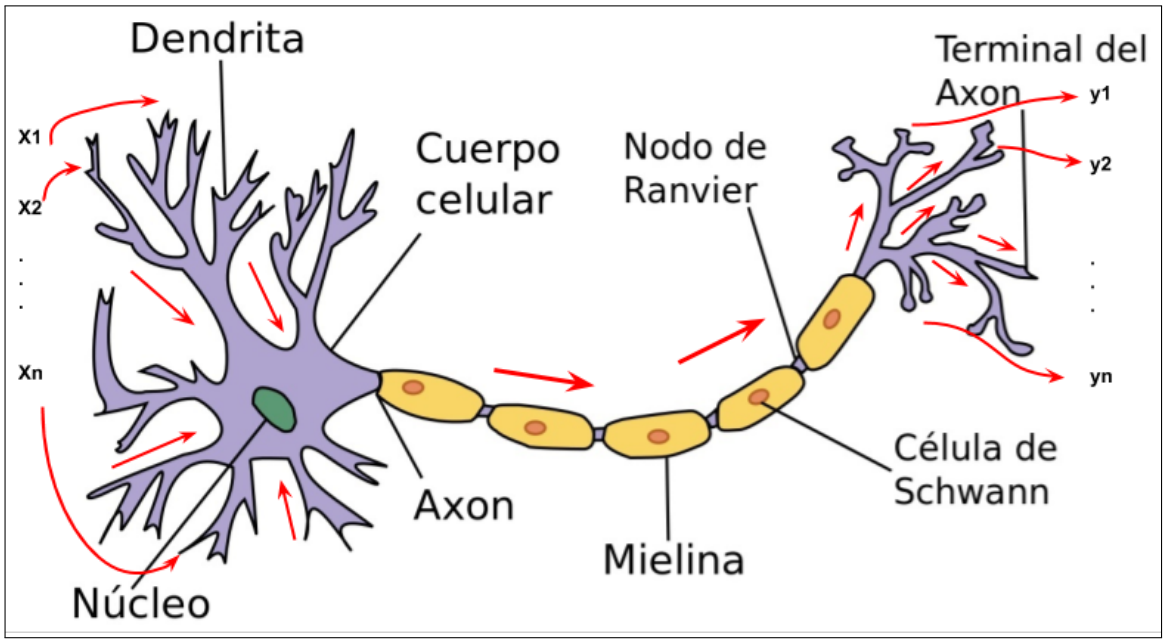


FIGURE 3.1: Graphic of a biological neuron. Reproduced from (Wikipedia, n.d.).

A brain neuron can receive about 10,000 entries and in turn send its output to hundreds of neurons.

The connection between neurons is called a **synapse**, this is not a physical connection, due there is 2 mm. of separation between neurons. These connections are unidirectional, where

[1] **ANN**, Artificial Neural Network

information's transmission is done electrically inside the neuron and chemically between neurons, thanks to neurotransmitters.

An **artificial neuron** is an elementary processor, because it processes a vector $x(x_1, x_2, ..., x_n)$ of inputs and produces a unique response or output. The main elements of an artificial neuron are the following:

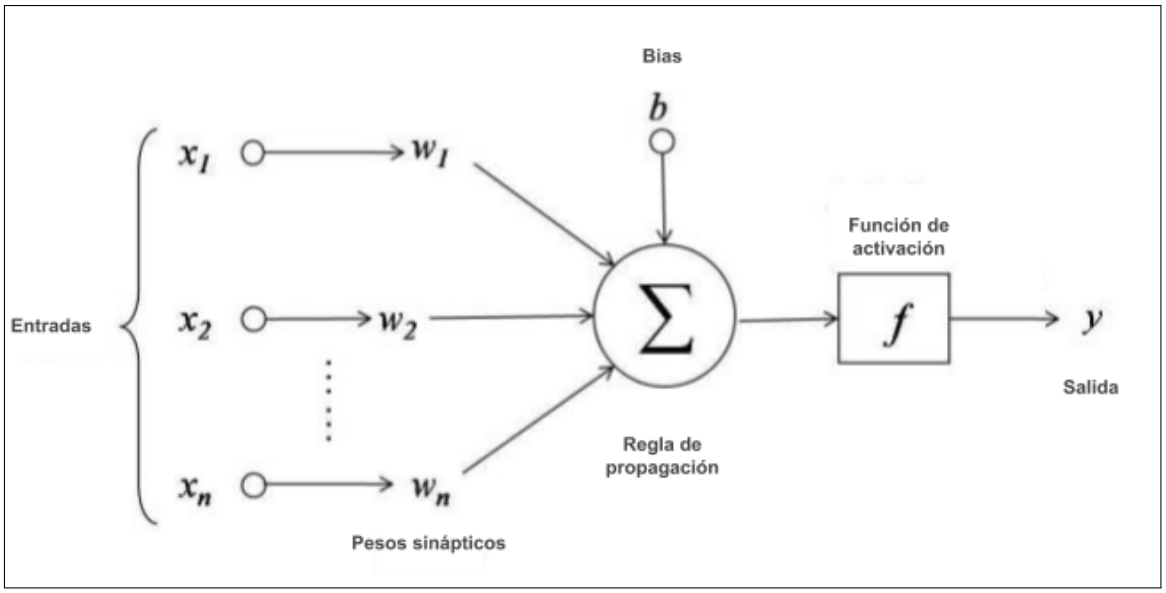


FIGURE 3.2: Graphic of an artificial neuron. Reproduced from (Jayesh, n.d.).

- **Inputs** that receive data from other neurons, these inputs would be dendrites of a biological neuron.
- **Synaptic weights** w_{ij}. In an artificial neuron, those inputs that come from other neurons are assigned a weight (importance factor). This weight is a numerical value that is modified during the training process of a neural network, and therefore it is here that information that makes the network serve one purpose or another is stored.
- **Propagation Rule**. With inputs and synaptic weights, some type of operation is usually done to obtain the potential postsynaptic value; one of the most common operations is to add up all entries, but taking into account importance (synaptic weight) of each one; This operation is called *weighted sum* 3.2, however other operations are also possible. Another propagation rule that is usual is Euclidean distance.

$$h_i(t) = \sum_j w_{ij} x_j \tag{3.2}$$

- **Activation Function**. The value obtained with the propagation rule is filtered through a function known as the *activation function* and is what gives the output of neuron. The activation function is important because it is the one that decides whether a neuron should

be activated or not, and if this function is not applied, the output signal of neuron would simply be a linear function.

3.3.2 Types of activation functions

There are different activation functions, then only the most used in field of neural networks will be presented.

Sigmoid activation function (logistics function)

A sigmoid function is a mathematical function that has a characteristic ”S” shaped curve or a sigmoid curve that ranges between 0 and 1 (See Figure 3.3a), so this function is usually used in models where you need to predict a Probability as an exit. This function is defined by the following formula:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

The sigmoid function was successfully applied in problems of binary classification, modeling of logistic regression tasks, as well as other neural network domains, however, it suffers significant drawbacks that include acute wet gradients during backward propagation from deeper hidden layers to input layers, gradient saturation, slow convergence and non-zero centered output, which causes gradient updates to propagate in different directions.

Hyperbolic Tangent Function - Tanh

It is quite similar to Sigmoid but has a much better performance compared to multilayer neural network training, its nature is nonlinear. This function is centered at 0 and its range is between -1 and 1 (See Figure 3.3b), therefore, its output is defined by:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.4}$$

Although this function has a better performance than the sigmoid, it could not solve the leakage gradient problem that sigmoid functions have. One of the main advantages of tangential function is that it produces a zero-centered output, which helps the backward propagation process.

Tangent functions have been used primarily in recurrent neural networks for natural language processing (Dauphin, Fan, Auli, & Grangiera, 2017) and speech recognition tasks (Mass, Hannun, & Ng, 2013).

Rectified Linear Unit function (ReLU)

The ReLU function was proposed by Nair and Hinton in 2010, and since then it has been the most widely used activation function for machine learning applications with neural networks.

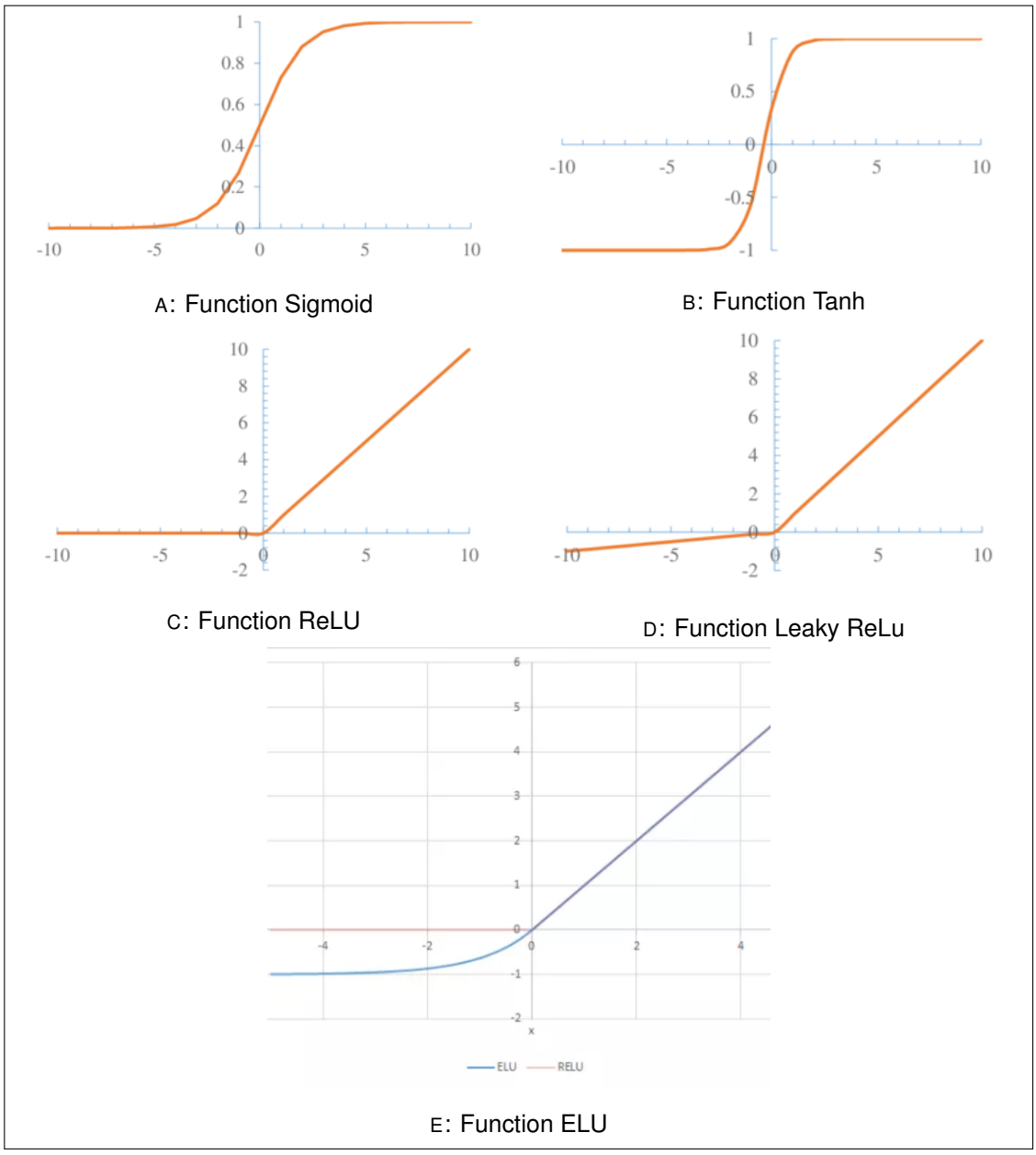


FIGURE 3.3: Activation functions (Jing & Guanci, 2018).

ReLU is a faster learning activation function (Lecun, Bengio, & Hinton, 2015), so it proved to be the most successful and most used function. This function offers better performance and generalization than sigmoid and tangent functions in learning with neural networks.

ReLU represents an almost linear function and, therefore, retains the properties of linear models that makes it easy to optimize, with gradient descent methods.

The ReLU activation function performs a threshold operation for each input element where values below zero are set to zero (See Figure 3.3c), so ReLU is defined by:

$$f(x) = max(0, x) = \begin{cases} \text{si } x_i \geq 0 & x_i \\ \text{si } x_i < 0 & 0 \end{cases} \tag{3.5}$$

This function rectifies the values of inputs below zero, forcing them to become zero, thereby eliminating leakage gradient problem observed in previous types of activation function. The ReLU function has been used within the hidden units of neural networks.

The main advantage of using ReLU is that it guarantees a faster calculation, since it does not calculate exponentials and divisions, with an improved general calculation speed (Zeiler et al., 2013). Another property of ReLU is that it introduces the shortage in hidden units, since it reduces values between zero and maximum. However, ReLU has the limitation that it is easily overfited compared to sigmoid function, although abandonment technique has been adopted to reduce overfited effect of ReLU and rectified networks improved performance of neural networks.

ReLU has a significant limitation that it is sometimes fragile during training, causing the death of some gradients. This makes some neurons also dead, to solve problems of dead neurons, the Leaky ReLU activation function was proposed.

Leaky ReLU (LReLU)

The year 2013 was proposed as an activation function, this function introduces a small negative slope to ReLU to keep and keep weight updates alive during the propagation process (Mass et al., 2013). The parameter α was introduced as a solution to the problems of dead neurons of ReLU. This function calculates gradient with a very small constant value for negative gradient α in the range of 0.01, so LReLU (See Figure 3.3d) is calculated as:

$$f(x) = \alpha x + x = \begin{cases} \text{si } x_i > 0 & x_i \\ \text{si } x_i \leq 0 & \alpha x_i \end{cases} \tag{3.6}$$

Exponential Linear Unit Function (ELU)

The ELU[2] function tends to converge the cost to zero faster and produces more accurate results. Unlike other activation functions ELU has an additional alpha constant that should be a positive number.

It is very similar to ReLU since both have an identity function for positive inputs, however in ELU negative inputs it softens slowly until its output is equal $-\alpha$ while in ReLU it softens sharply. ELU function is calculated according to equation 3.7.

$$f(x) = \begin{cases} \text{si } x_i > 0 & x_i \\ \text{si } x_i \leq 0 & \alpha * (e^{x_i} - 1) \end{cases} \tag{3.7}$$

3.3.3 Architecture of the Neural Networks

A regular neural network consists of a chain of interconnected layers of neurons, these layers are: an input layer, one or several hidden layers and an output layer.

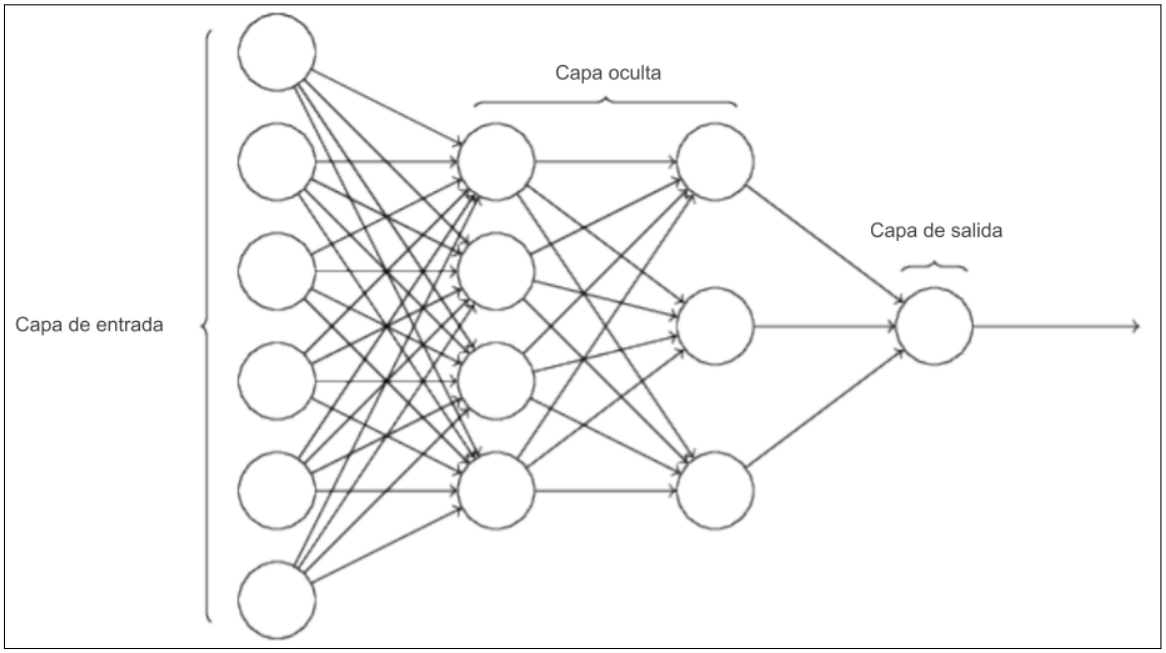


FIGURE 3.4: Architecture of an artificial neuron. Reproduced from (Michael, 2015).

Figure 3.4 is an example of a very simple neural network; In this figure the leftmost layer is called **input layer**, and the neurons within this layer are called input neurons. The rightmost or **output layer** contains output neurons. And finally the two intermediate layers are **hidden**

[2] **ELU,** Exponencial Lineal Unit

layers of neural network, these are called that because neurons of this layer are not input or output; A neural network can have one or more hidden layers.

Unlike the human brain, an Artificial Neural Network has a fairly strict predefined structure, connections between neurons are always forward (feedforward): connections range from the neurons of a given layer to neurons of next layer, that is, There are no side connections or back connections. This means that a neuron that was activated in layer 3 cannot activate a neuron in layer 2 or earlier.

3.3.4 Learning process of Neural Networks

A key feature of neural networks is their iterative learning process, that is, each sample of training set is presented to the network, one at a time, so that the weights associated with input values are adjusted each time. During this learning phase, the network trains by adjusting the weights to predict a correct output for input samples.

Neural networks have the advantage of having a high tolerance for noisy data, as well as a high capacity to classify patterns with which they have not been trained. The most popular neural network training technique is the backpropagation algorithm.

Once the structure of a network is defined for a particular application, it is ready to be trained. To begin this process, initial weights are chosen at random, and then proceed with training (learning).

Backpropagation

A neural network propagates the signal of input data forward through its parameters at the time of decision; and then spread back the information about the error, so that parameters can be altered. This happens by following steps below:

- The network guesses output data, using its parameters.
- The network measures its accuracy with a loss function.
- The error is propagated backwards to adjust the wrong parameters.

Therefore it can be said that Backpropagation algorithm takes the error associated with an erroneous assumption by neural network, and uses that error to adjust parameters of neural network in the direction that generates the least error.

3.4 Types of Neural Networks

3.4.1 Autoencoders

An Autoencoder is an Artificial Neural Network used for unsupervised machine learning, it is trained to reconstruct its own inputs, that is, predict the value of output $\hat{x}$ given an input x via a hidden layer h, see Figure 3.5. Autoencoders are usually used for dimensionality reduction and feature learning.

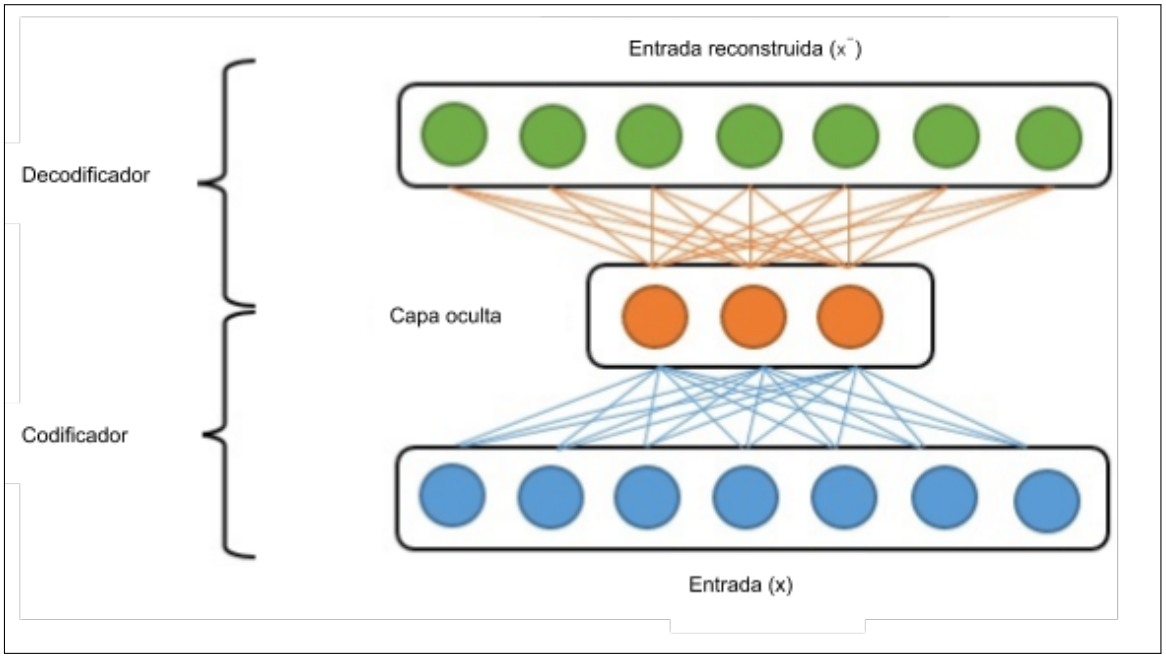


FIGURE 3.5: Graphic of an Autoencoder (Own elaboration).

Autoencoders are composed of two parts: the encoder and decoder. Encoder learns a compressed representation of input data, this can be defined with the coding function $h = encoder(x)$, which is defined by a linear or nonlinear function. If function of encoder is non-linear, auto-encoder will be able to learn more features than a linear PCA. The purpose of decoder is to reconstruct its own input via the decoding function, $\hat{x} = decoder(h)$.

The difference between input and reconstructed input is the reconstruction error. During training, autoencoder minimizes the reconstruction error as an objective function. Autoencoders are often used for data generation as generative models. Decoder of an autoencoder can generate an output given an artificially assigned compressed representation.

3.4.2 Convolutional neural networks

For some types of data, specifically for images, conventional neural networks are not well adapted; which implies that in the study Lecun et al. (1998) propose convolutional neural

networks (CNN[3]) to solve this problem. CNNs have revolutionized image processing and eliminated manual feature extraction. A CNN acts directly on matrices, or even on tensors for images with three RGB color channels; so currently, CNNs are widely used for image classification, object recognition, face recognition, among others.

CNNs not only provide better performance compared to other detection algorithms; but even in some cases they outnumber humans, such as in classification of objects in specific categories such as particular breed of a dog or a species of bird (Russakovsky, 2014).

By stacking multiple and different layers in a CNN, complex architectures are created for classification problems. The four types of layers that are most common are: convolution layer, grouping/subsampling layer, nonlinear layer and fully connected layer. An example of a CNN can be seen in Figure 3.6, where the first and third layers are convolutional layers, the second and fourth are subsampling layers and finally the fifth and sixth layers with completely connected layers.

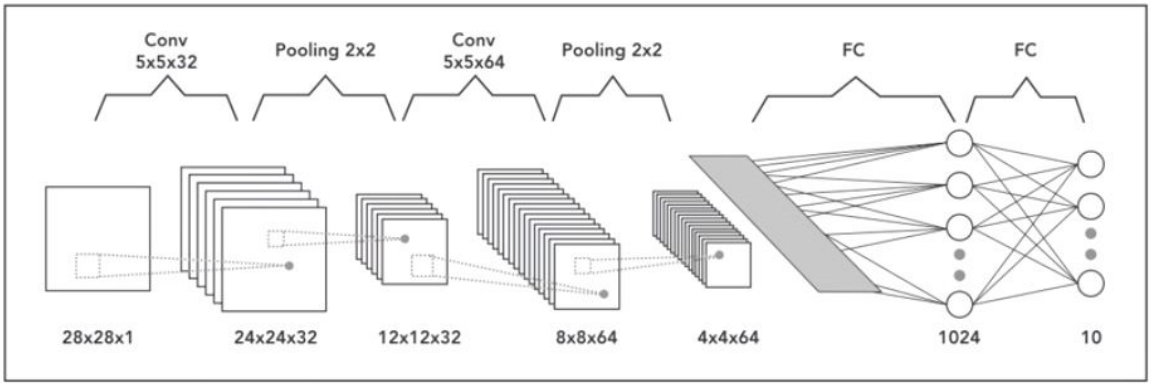


FIGURE 3.6: Architecture of a Convolutional Neural Network (CNN) (Muhammad, n.d.).

3.4.3 Recurrent neural networks

To understand the importance of time series, the following analogy can be taken, human beings do not start to think from scratch every second, so when reading a document each word is understood based on understanding of the previous words, that is to say , everything is not eliminated and you start thinking of zero every time, given this statement you can say that thoughts of human beings have persistence.

Traditional neural networks do not have data persistence, which for some specific problems, including the one of this work, is a major deficiency. In order to solve these types of problems, Recurrent Neural Networks (RNN[4]) appear, which are a type of artificial neural network proposed in the 80s (Rumelhart, Hinton, & Williams, 1986; Elman, 1990; Werbos, 1988) designed to

[3]**CNN,** Convolutional Neural Network
[4]**RNN,** Recurrent Neural Network

recognize patterns in data streams, such as text, genomes, handwriting, numerical time series data emanating from sensors, among others.

RNNs are a particular family of neural networks where the network contains one or more feedback connections, so that the activation of a group of neurons can flow in a loop. This property allows model to retain information about past, which allows it to discover temporal correlations between events that are far from each other in data.

RNN have a certain memory of what happened previously in a sequence of data, this helps system to gain context of data. Theoretically it is said that RNN has infinite memory, that is, these types of networks have ability to look back indefinitely; however, in practice you can only look back a few last steps.

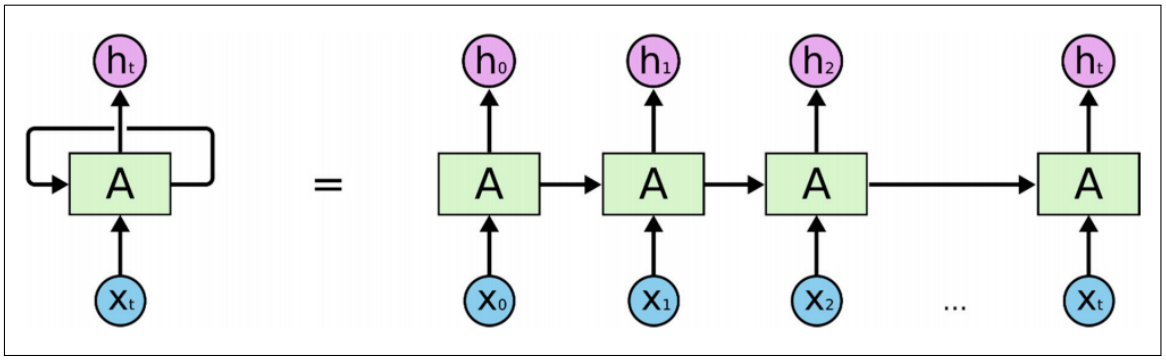


FIGURE 3.7: Sequential processing in a recurrent neural network (RNN) (Olah, n.d.).

Figure 3.7 illustrates a simple RNN with an input unit, an output unit and a recurring hidden unit expanded in a complete network, where x_t is input in the time step t and y_t is output in the time step t. On the other hand, in training process, RNNs use the Backpropagation algorithm over time (BPTT[5]). The BPTT process uses a backward, layer by layer, work approach of final output of network, adjusting the weights of each unit according to calculated unit portion of total output error. The repetition of information loops results in large updates of neural network model weights and leads to an unstable network due to accumulation of error gradients during the update process. Therefore, BPTT is not efficient enough to learn a pattern of long-term dependence due to disappearance of gradient and the explosion of gradient problems (Bengio, Simard, & Frasconi, 1994). To overcome disappearance and explosion gradient problems that standard RNNs have, LSTM and GRU can be used (Pascanu, Mikolov, & Bengio, 2013).

3.4.4 LSTM

LSTM[6] is an evolution of RNN, it was introduced by Hochreiter and Schmidhuber in (1997) to board the problems of standard RNNs that were mentioned before, adding additional interactions

[5] **BPTT,** Backpropagation Through The Time
[6] **LSTM,** Long Short-Term Memory

per module (or cell). LSTMs are a special type of RNN, capable of learning long-term dependencies and remembering information for extended periods by default.

The LSTM model is organized in form of a chain structure. However, the repetition module has a different structure. Instead of a single neural network like a standard RNN, it has four interactive layers with a unique method of communication. The LSTM's structure is shown in Figure 3.8.

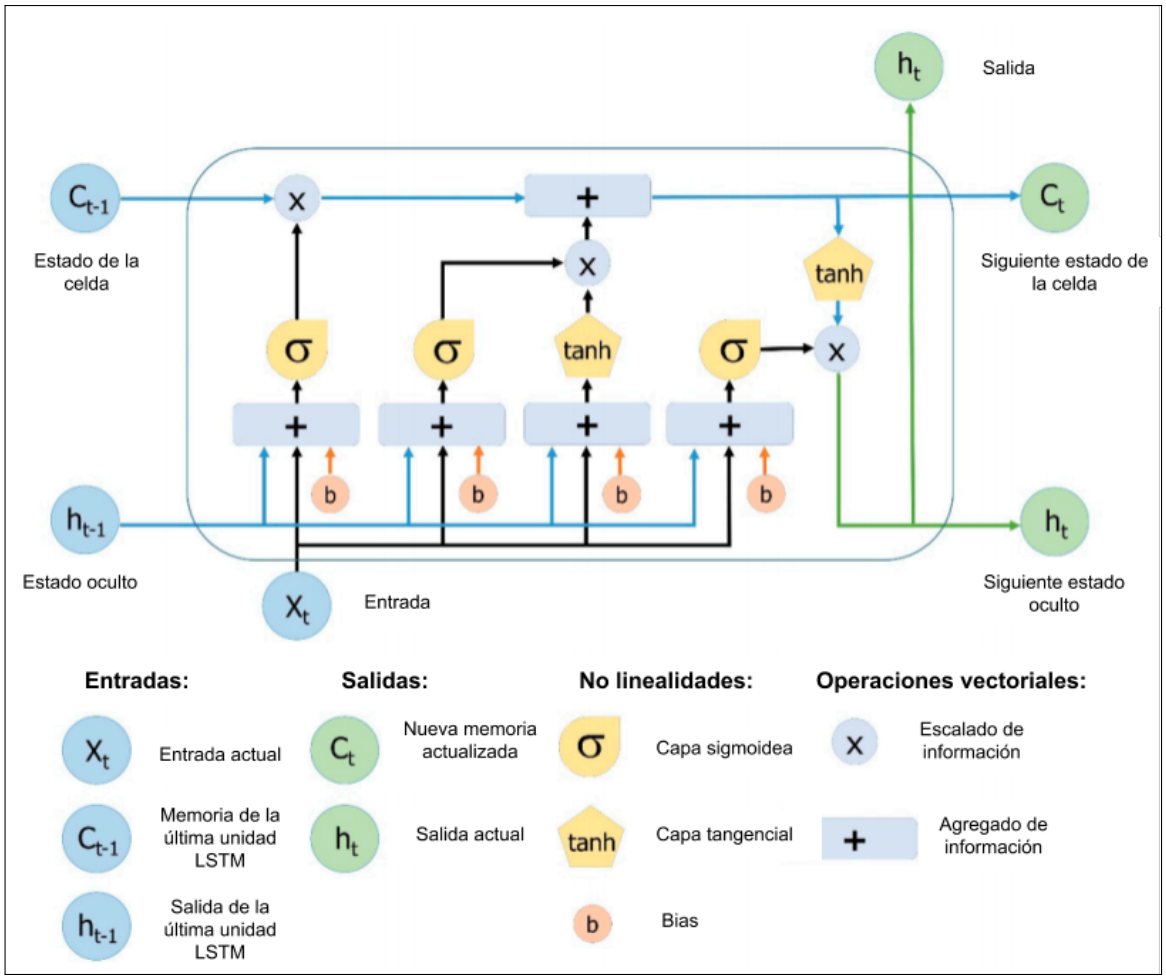


FIGURE 3.8: LSTM structure. Played from Yan (Yan, n.d.).

A typical LSTM network is made up of memory blocks called cells. Two states are transferred to next cell, the state of cell and hidden state. The **cell's state** is the main chain of data flow, which allows data to flow forward, essentially, without changes. However, some linear transformations may occur. Data can add or remove the cell's state through sigmoid gates.

A door is similar to a layer or a series of matrix operations, which contains different individual weights. LSTMs are designed to avoid the problem of long-term dependence because it uses doors to control memorization process.

3.4.5 GRU

GRU[7] was first designed by Kyunghyun Cho in (2014a). This RNN structure only contains two doors. The update door controls information that flows into memory, while the reset door controls information that flows out of memory. Similar to LSTM, GRU has gate units that modulate flow of information within the unit; however, without having a separate memory cell. One could say that GRU is a slightly more simplified variant of RNN that often offers comparable performance to LSTM and is significantly faster to calculate.

In summary, GRUs have following two distinctive characteristics:

- **Reset doors** help capture short-term dependencies in time series.
- **Update doors** help capture long-term dependencies in time series.

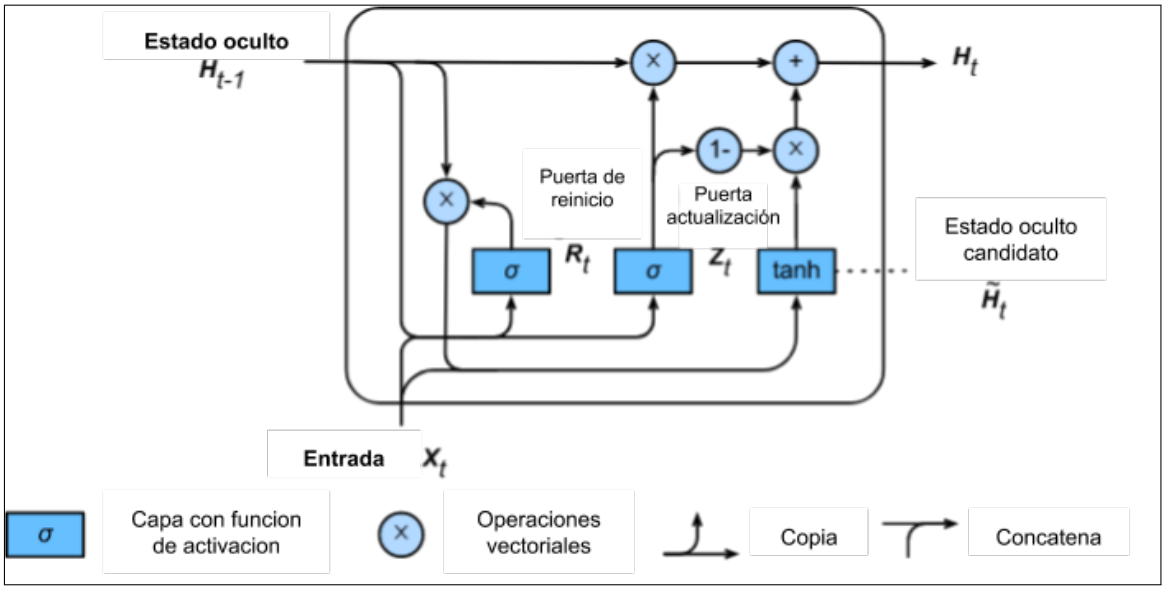


FIGURE 3.9: GRU structure. Reproduced from (Zhang et al., 2019).

3.5 Anomaly detection techniques

There are several approaches available for anomaly detection, in this section some of the most used algorithms will be described.

[7] **GRU,** Gated Recurrent Unit

3.5.1 One-Class SVM

One-Class SVM[8] (OC-SVM) is a widely used approach to discover anomalies in an unsupervised manner (Schölkopf and Smola, 2002). OC-SVMs are one of the most widely used semi-supervised learning techniques, because it gives good results even for large data sets. However, this algorithm has the disadvantage that it requires a lot of time and memory in practice and its complexity grows quadratically with number of records.

This algorithm is only trained with positive examples (normal classes). The general idea of this algorithm is to transform the attribute space and draw a divisional hyperplane so that observations are as far as possible from origin, as can be seen in Figure 3.10.

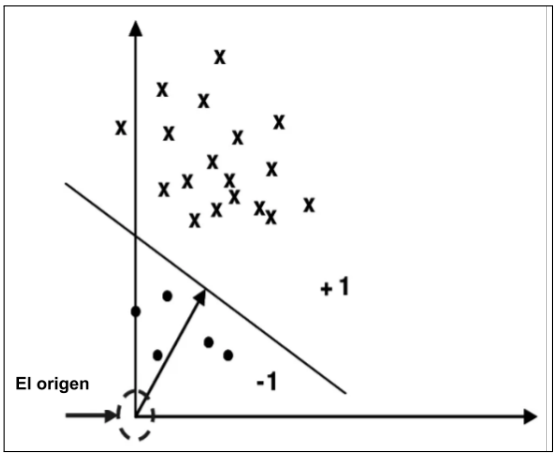


FIGURE 3.10: One-Class SVM. Reproduced from (Alashwal et al., 2006)

As a result, a margin is obtained, on one side of which observations of training sample are grouped as densely as possible (normal observations $\hat{y}_i = 1$), and on the other, abnormal values are found ($\hat{y}_i = -1$) , not similar to what the algorithm saw during training.

3.5.2 Isolation Forest

This model is used in a scenario similar to One Class SVM, specifically in an unsupervised environment. The isolation forest takes a different approach to the OC SVM, since instead of grouping normal data, it tries to isolate the anomalous data.

The basic component of Isolation Forest is the isolation tree, which is a simple binary tree where in each T_i node both characteristic and threshold for division rule are randomly selected. An existing node stops generating children if and only if there is only one example following the division rule for that specific route (which means that the example has been isolated) or a maximum height has been reached. This means that at the end of the training process we

[8] **SVM,** Support Vector Machine

will have a completely over-adjusted random classification tree, which can be used for anomaly detection purposes. The main intuition of this algorithm is that if an example is anomalous, it will be isolated after some cuts in features space, which translates into having a low height in isolation tree.

Unlike other methods such as grouping or classification, isolation forests do not learn a profile of what is normal, but instead directly attack anomalies. No distance metric is used in this algorithm which saves time in calculations; so isolation forests have a linear temporal complexity.

A comparison of the ability of Isolation Forest and One-Class SVM algorithms to cope with different two-dimensional data sets is presented in Figure 3.11, with the aim of giving some intuition about behavior of these algorithms.

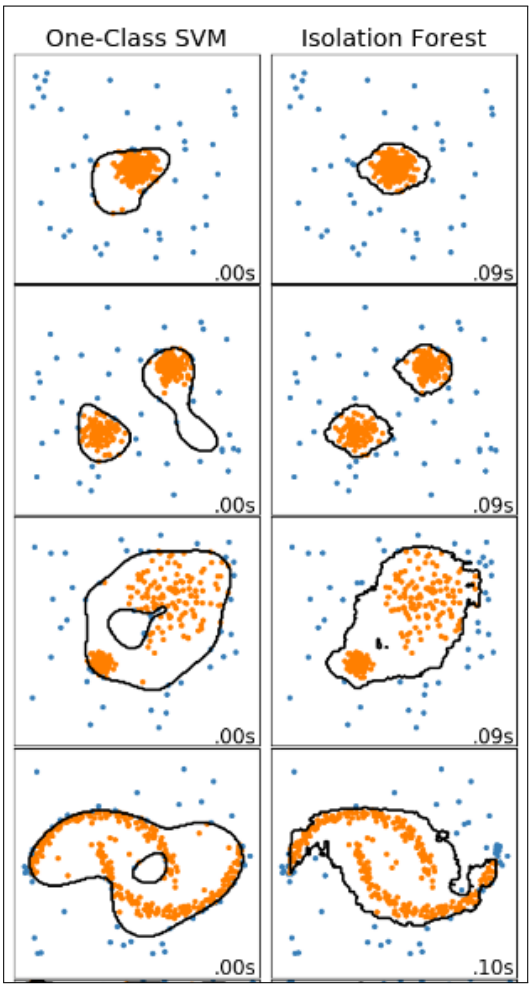


FIGURE 3.11: Performance comparison between the One-Clas SVM and Isolation Forest algorithms. Reproduced from (0.22, n.d.).

3.5.3 Autoencoders

Today, autoencoders have been widely used in image classification, machine translation and voice processing; This is due to its ability to compress data without supervision. As far as is known, Hawkins et al. (2002) and Williams and Baxter (2002) were the first to propose autoencoders for anomaly detection. Since then, the ability of auto-encoders to detect outliers was demonstrated in different domains such as the detection of anomalies in X-rays.

The traditional method of autoencoder-based anomaly detection is mainly based on reconstruction error, considering as anomalies those samples that present a high reconstruction error. In training phase, only normal data is used to train the auto-encoder, in order to minimize the reconstruction error, so that auto-encoder can recognize characteristics of normal data. In test phase, the trained auto-encoder will be able to reconstruct normal data with small reconstruction errors, but they will fail with anomalous data that auto-encoder has not encountered before and, therefore, have relatively higher reconstruction errors compared to normal data. Therefore, when comparing whether the reconstruction score of an anomaly is above a predefined threshold, auto-encoder will determine if the data presented for test is anomalous (Guo et al., 2018) (See Figure 3.12) . Equation 3.8 shows how this technique determines what an anomaly is and what is not; where S_z represents the reconstruction.

$$C(z) = \begin{cases} \text{if } S_z \leq \text{Threshold} & \text{Normal behavior} \\ \text{if } S_z > \text{Threshold} & \text{Anomaly} \end{cases} \tag{3.8}$$

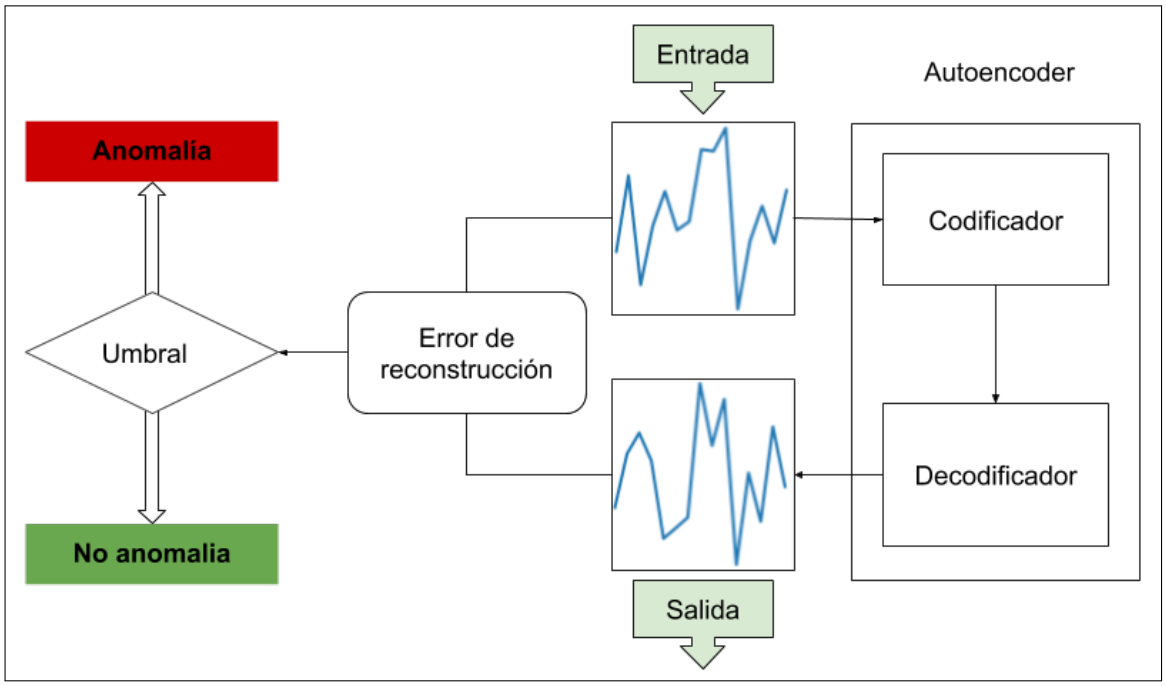


FIGURE 3.12: Detection of anomalies with autoencoder (Own elaboration).

3.6 Evaluation Metrics

Evaluating machine learning algorithms is an essential part of any project, because a model can provide satisfactory results when evaluated with a metric; but it can give a poor result when another metric is used. Classification accuracy is commonly used to measure the performance of a model, however, it is not enough to judge a model. Different types of evaluation metrics will be covered below.

3.6.1 Classification Accuracy

The classification accuracy is the relationship between the number of correct predictions and the total number of input samples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \tag{3.9}$$

This metric works well if there are the same number of samples that belong to each class; for example, if you consider that you have a data set that contains 98% of samples that belong to class A and 2% that belong to class B, the model could easily obtain 98% training accuracy by simply predicting each training sample as class A.

This implies that precision can give the false feeling of achieving high precision, which becomes a real problem if it involves problems that involve the detection of high-risk things; for example, a rare but deadly disease since cost of not diagnosing a sick person's disease is much higher than cost of sending a healthy person for further analysis.

3.6.2 Logarithmic Loss

Logarithmic loss, also known as Log Loss, works by penalizing false classifications, and also has a good performance for the classification of various classes. When working with Log Loss, the classifier must assign a probability to each class of all samples; assuming that there are N samples that belong to M classes, Log Loss is calculated according to the following equation:

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * log(p_{ij}) \tag{3.10}$$

where:
y_{ij}, indicates whether sample i belongs to class j or not.
p_{ij}, indicates the probability that the sample i belongs to class j.

Log Loss has no upper limit and exists in the range $[0, \infty)$. When a Log Loss close to 0 is obtained, it indicates greater precision, while if it is far from 0 it indicates less precision. In general, it can be said that minimizing Log Loss provides greater accuracy for classifier.

3.6.3 Confusion Matrix

Confusion matrix, also called error matrix, is the most common method to evaluate accuracy of a classification result (Smits, Dellepiane, & Schowengerdt, 1999). This matrix is a cross-tabulation of expected data and results of classification model. The number of columns and rows is equal to number of categories in classification and different statistical measures are derived from them.

		Prediction	
		Positive Class	**Negative Class**
Real	**Positive Class**	True Positive (TP)	False Negative (FN)
	Negative Class	False Positive (FP)	True Negative (TN)

TABLE 3.1: Confusion matrix, for a binary classification (Own elaboration).

In Table 3.1 the rows of matrix represent the real values, while columns are associated with the data classified by the model (predictions). The main diagonal, which appears in green, indicates the successes or True Positives (TP) and True Negatives (TN), which are all those data where model obtains the same result that was expected to be obtained. As for all the other values of the matrix, they belong to those data that were classified incorrectly, these are classified into two classes: False Positives (FP), which inthe matrix are presented in red and False Negatives (FN) that in the matrix they were represented in orange.

The overall accuracy of matrix is calculated by dividing the sum of correctly classified samples by the total number of samples taken 3.11. This value is a measure of classification as a whole, since it indicates the probability that a sample is correctly classified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.11}$$

Accuracy is not an adequate measure for the evaluation of atypical value detection algorithms, because most of the time a false negative is much more expensive than a false positive, in addition to training sets presenting a large amount of data normal compared to amount of anomalous data. There are two common metrics to evaluate outlier detection algorithms, AUC and F1 Score; These two metrics will be deepened in detail.

3.6.4 Area Under Curve (AUC)

Area Under Curve (AUC) is one of the most used metrics for evaluation. It is used for binary classification problems.

The AUC of a classifier is equal to the probability that the classifier classifies a positive example chosen at random higher than a negative example chosen at random. Before defining AUC, following terms must be understood:

Recall or Sensitivity or True Positive Rate (TPR)

True positive rate, also known as sensitivity, corresponds to the proportion of positive data points that are correctly considered positive, with respect to all positive data points. It is defined according to equation 3.12.

$$Sensitivity = \frac{TP}{FN + TP} \tag{3.12}$$

Specificity or True Negative Rate (TNR)

True negative rate, also known as specificity, corresponds to the proportion of negative data points that are correctly considered negative, with respect to all negative data points. It is defined according to equation 3.13.

$$Specificity = \frac{TN}{FP + TN} \tag{3.13}$$

ROC (Receiver Operating Characteristics)

ROC is the curve drawn by connecting the points on the X axis = FPR (False positive rate) and the y axis = TPR (True positive rate) for different values of a discrimination threshold (decision limit to determine if a value corresponds to a class or not) for a model, that is, different thresholds are chosen for a model, the TPR and FPR are calculated for each threshold, then it draws the ROC curve and finally the AUC is calculated, which is the area under the curve ROC . An example of the AUC-ROC curve is shown in Figure 3.13.

There are two main reasons why this curve is needed, first is that it reflects how good the model is to separate two classes and second is that it helps to choose the best threshold; for example, an AUC equal to 0.5 means that model separates two possible random results and an AUC of 1 (maximum value) implies a perfect separation; therefore, it can be said that the higher the value of AUC, better will be the performance of evaluated model.

3.6.5 F1 Score

F1 Score defines how accurate is a model, that is, how many instances it classifies correctly, as well as indicating how robust is model. This metric is necessary when you want to find a balance between accuracy and recovery, since it gives a fair evaluation even when the data set is unbalanced.

Precision

Precision is the number of correct positive results divided by the number of positive results predicted by classifier.

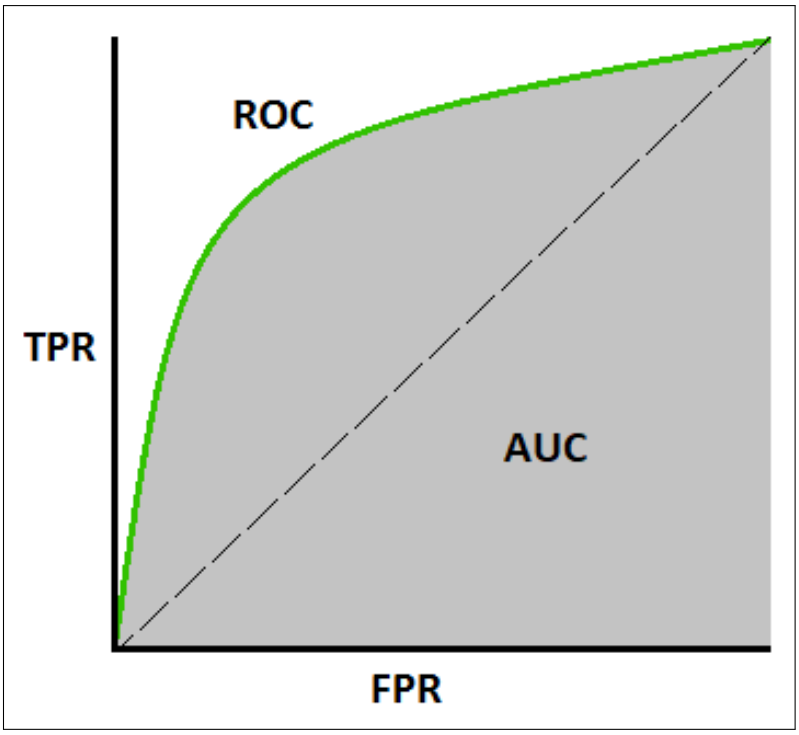


FIGURE 3.13: Example of an AUC-ROC curve (Özler, n.d.).

$$Precision = \frac{TP}{TP + FP} \tag{3.14}$$

Recall

It is the number of correct positive results divided by the number of all samples that should have been classified as positive.

$$Recall = TPR = \frac{TP}{TP + FN} \tag{3.15}$$

Therefore, F1 Score is expressed mathematically as:

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{3.16}$$

Next chapter details the process of capturing and preparing the data set, an important stage, because this set will be the one with which proposed anomaly detection mechanism will be trained.

Chapter 4

DATA CAPTURE AND PREPARATION

Having a large amount of data in any anomaly detection problem is what makes it possible to generate more accurate models, because you never know what characteristics can indicate an anomaly, having multiple types of data is what allows you to go more beyond a mere detection of specific anomalies and being able to identify more sophisticated contextual or collective anomalies. However, obtaining this data is not always a simple task, so you often have to find a way to generate it.

This chapter will detail the method of data collection that was performed for this research and the different data analysis techniques that were applied.

4.1 Data Capture

Currently there are several approaches to access the information of driver (agent) and vehicle. In the first approach, a set of sensors and additional hardware are previously implemented in the vehicle, for example, telematic boxes (black boxes provided by car insurance companies), on-board diagnostic adapters (OBD-II) plugged into the controller of network area vehicle (CAN) (Zaldivar, Calafate, Cano, & Manzoni, 2011; Araujo, Igreja, R., & Araujo, 2012), the information recorded by these devices can be retrieved or sent via Internet. However, this strategy requires that vehicles install additional devices, which implies a higher cost. To overcome these inconveniences, there is an alternative approach which is to use smartphones to collect data through a set of integrated sensors, such as inertial sensors (accelerometers and gyroscopes), global positioning systems (GPS), magnetometers, microphones, sensors Image (cameras), light sensors, proximity sensors, direction sensors (compass), among others.

For this research work, the use of smartphones was chosen to access the type of driving information, for reasons presented above, with this approach an Android-based mobile application was developed to collect sensor data: accelerometer and gyroscope, in intervals of 1 second, which in the first instance will be stored internally in the mobile device. (Ver Figura 4.1)

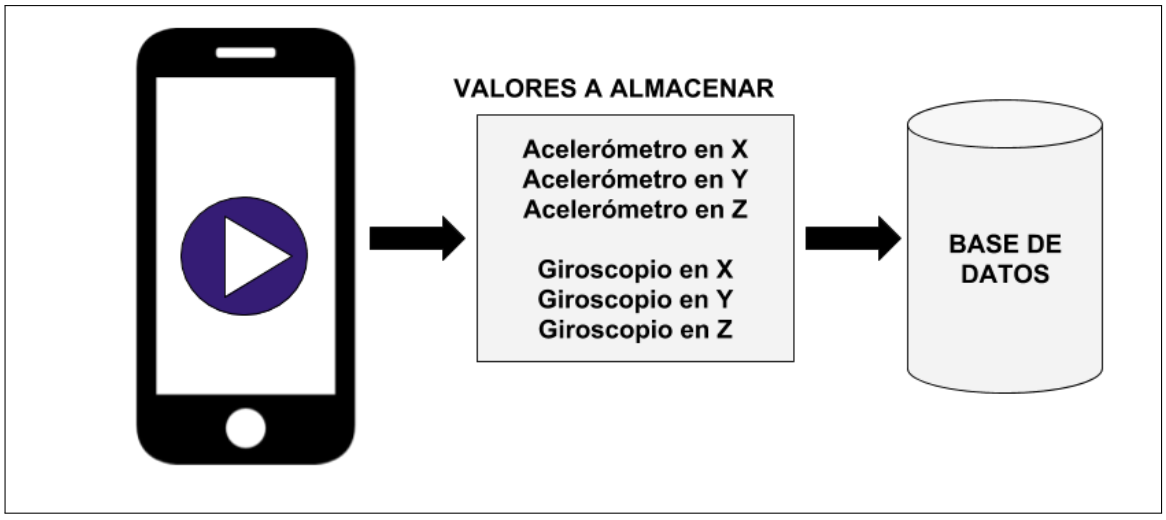


FIGURE 4.1: Data collection, with interval of one second (Own elaboration).

For data collect, a windshield cell phone holder was used as seen in Figure 4.2; the capture was made in two different positions (vertical and horizontal).

FIGURE 4.2: Windshield cell mount, horizontal position (Own elaboration).

Each capture, regardless of position in which it was made, resulted in a dataset (dataset), where for each time T (1 sec.) There are six variables: accelerometer in X (acc x), accelerometer in Y (acc y), accelerometer in Z (acc z), gyroscope in X (gyr x), gyroscope in Y (gyr y) and gyroscope in Z (gyr z). A fragment of data set that was obtained in a capture is shown in Figure 4.3

4.2 Data preparation

Machine Learning depends heavily on the data. They are the most crucial aspect that makes algorithm training possible and explains why machine learning has become so popular in recent

_id	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z	fecha
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	3.69699096...	-0.4218902...	9.07658386...	0.00050354...	7.62939453...	0.00115966...	2019-10-19
2	3.68769836...	-0.3159942...	9.03315734...	0.00183105...	-0.0009918...	9.15527343...	2019-10-19
3	3.69015502...	-0.3061065...	9.10673522...	0.00210571...	-0.0009918...	0.00035095...	2019-10-19
4	3.68812561...	-0.3355712...	9.43148803...	0.00343322...	-0.0135345...	-0.0023040...	2019-10-19
5	3.69512939...	-0.3061065...	9.13452148...	0.00263977...	-0.0009918...	-0.0009765...	2019-10-19

FIGURE 4.3: Fragment of data set obtained (Own elaboration).

years. The main problem is that all data sets have failures, which makes data preparation a very important step in the Machine Learning process.

The main purpose of data preparation is to manipulate and transform raw data, so that data can be exposed or made more easily accessible (Pyle, 1999), to achieve this purpose a process that involves selection must be followed, Pre-processing and data transformation.

4.2.1 Data selection

Data selection involves the following steps:

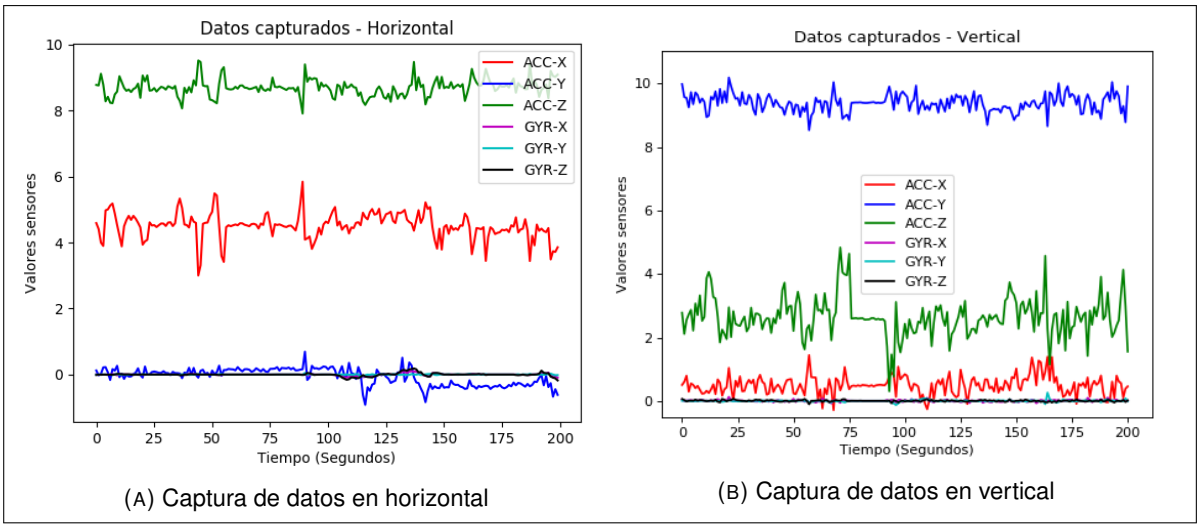
- Select only a subset of available data.
- Derive or simulate some data from available data, if necessary.
- Exclude data that is not relevant to the problem.

For present work, only the first step of this phase will be emphasized, because the data available is limited since they were captured for investigation and as indicated in section 4.1, this capture was made, both vertically and horizontally, the differences between them will be analyzed below.

Fragments of obtained captures by mobile device of the same user (agent) from different positions are shown in Figure 4.4.

Although captured values are very similar to each other, data that was captured with mobile device in a horizontal position, presents less noise, this because this position favors the inertia of device when vehicle is in motion, which is a great advantage over data that were captured vertically, since these were more susceptible to shaking while vehicle was moving causing the values captured in this position to present movement values not only of vehicle but also of mobile device, which is not what is sought in the present work.

For reasons presented in previous paragraph, it was decided to work with the data captured with mobile device in a horizontal position, thus discarding those data captured vertically.



(A) Captura de datos en horizontal

(B) Captura de datos en vertical

FIGURE 4.4: Graph of sensors captured in different positions (Own elaboration).

4.3 Data preprocessing

Once data with which you will work will be selected, you must proceed to preprocess them, thus entering the Pre-processing phase of data, the objective of this phase is to reduce amount of data, find relationships between them, normalize them, remove outliers and extract features of data. This phase includes several techniques such as cleaning, integration, transformation and data reduction (Suad & Wesam, 2017).

4.3.1 Data cleaning

Row data may have incomplete records, noisy values, outliers and inconsistent data. Data cleaning in most cases is the first step of data pre-processing. This technique is used to compensate for missing values, soften the noise in the data, recognize outliers and correct inconsistencies.

Compensation techniques for incomplete records

Many real-world data sets may contain missing values for different reasons; which can drastically affect the quality of machine learning model.

Below are four compensation techniques for data sets, with the aim of coping with the problem of missing values.

- **Ignore** / **Delete:** In some cases it is better to ignore or eliminate the tuple that contains missing values instead of filling it. Generally this technique is practiced in data sets that are very large, where deleting some data will not affect the information transmitted by data set. However, when working with a small data set, removing tuples that contain missing values could cause you to lose important information.

- **Fill out missing values manually:** Another option is also to complete missing values if you understand the nature of them, this is usually done in a small data set, since in large sets this task would require a lot of time.
- **Fill the missing values with central values (Medium / Medium):** This technique is much better than those presented above. In this technique, the mean or median of respective attribute is inserted to missing values.
- **Interpolation:** It is a reliable, accurate and scientific way to complete missing values. To use this technique, a relationship between attributes must first be developed and then the most probable and precise value for missing places is predicted, this can be achieved through regression, bayesian formulation and induction by decision trees.

Remove noise from data (smoothing)

To understand this technique you must first define what is noise in data. Noise in data is any type of random error or variation in measured attributes; on the other hand outliers present in data can also be considered as noise.

It is important to note that noise present in data set can greatly affect result of Automatic Learning algorithms, therefore those data that contain noise are not considered good data and should be eliminated as much as possible. However before removing these, you should be able to detect them; To achieve this goal there are many techniques that can be used, one of these techniques is Visualization of data, which consists in obtaining a visual representation of data, to be able to show if there is noise in data and/or outliers.

In this work, histograms of frequencies of each characteristic of data set were plotted and thus visualized of a better way if there was noise or outliers in it. In Figure 4.5 it can be shown that values of each sensor have negative and positive asymmetries, in addition to having many values far from the average, which gives an indication of possible outliers.

A table with descriptive statistics of data set is presented in Figure 4.6, this table presents: the amount of data, its mean, its standard deviation, the minimum and maximum value of data set and the 50, 25 and 75 percentiles, it should be noted that the percentile 50 is same as median.

With the information provided in Figure 4.6, it is now easier to perform an analysis of data set, first it is evident that the values obtained during the capture, have very small standard deviations between 0.05 and 0.81 and yet difference between the values minimum and maximum are really large, for example for the Accelerometer in X, difference is approximately 10 (Minimum value: -3.40 and maximum value: 7.17), this means that set of data with which it works presents a lot of noise, considering as noise or outliers, those values far from the average.

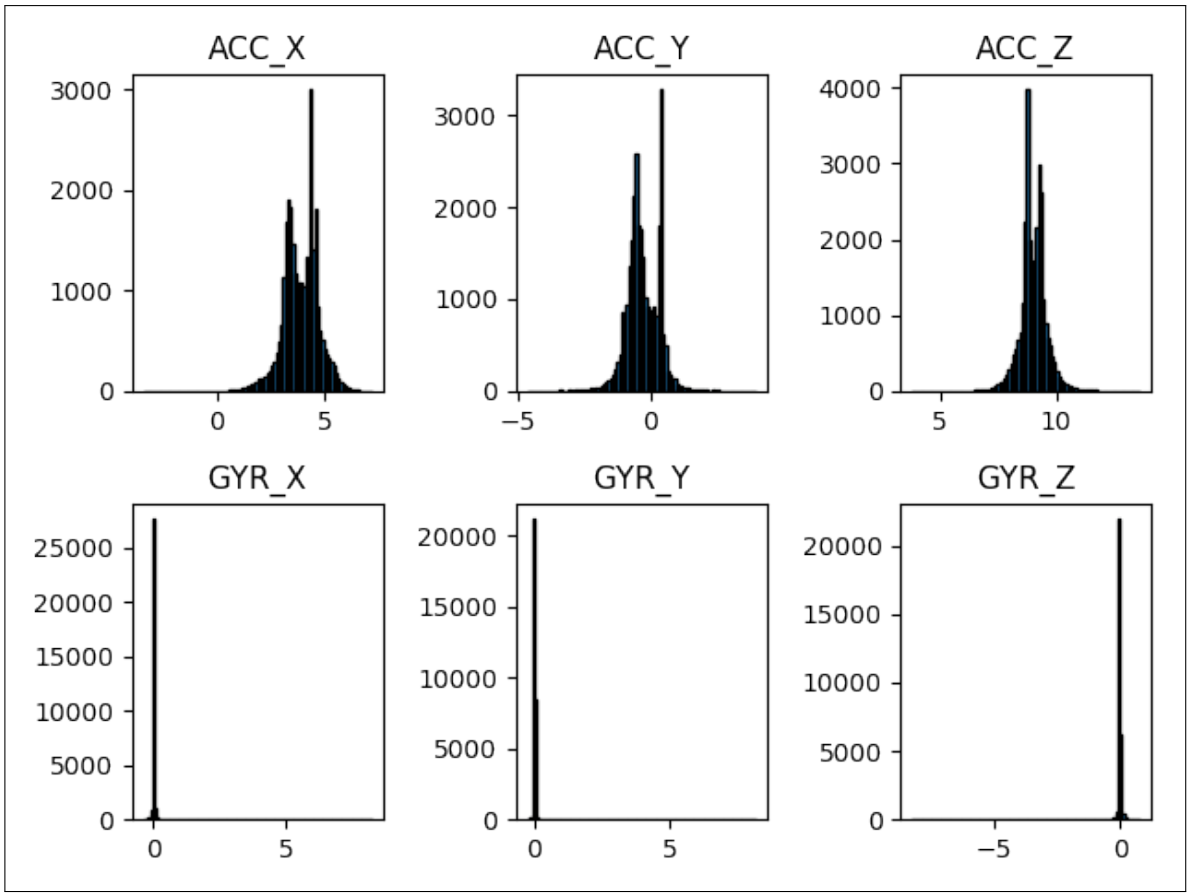


FIGURE 4.5: Histogram of data set's frequencies (Own elaboration).

	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	3.874731	-0.302470	8.997619	0.000086	0.000281	-0.000389
std	0.809801	0.614389	0.563649	0.055470	0.052041	0.074062
min	-3.403488	-4.633270	3.801849	-0.388611	-0.251450	-8.214310
25%	3.311012	-0.684528	8.707027	-0.005192	-0.004990	-0.004837
50%	3.907730	-0.381134	8.988884	-0.000015	-0.000015	-0.000031
75%	4.412102	0.235142	9.307396	0.004745	0.004807	0.004578
max	7.164932	3.947861	13.609085	8.208740	8.212128	0.818634

FIGURE 4.6: Table of data set's statistical results (Own elaboration).

To eliminate the noise presented by data set, Rule 68-95-99.7, also known as Empirical Rule, was applied, assuming that data set with which it works has a normal distribution, standard deviation can be used to determine the proportion of values that fall within a particular range

of average value. For such distributions, it is always the case that 68% of the values are less than a standard deviation (1SD) of average value, that 95% of values are less than two standard deviations (2SD) of average and that the 99% of values are less than three standard deviations (3SD) from average. Figure 4.7 shows this concept schematically.

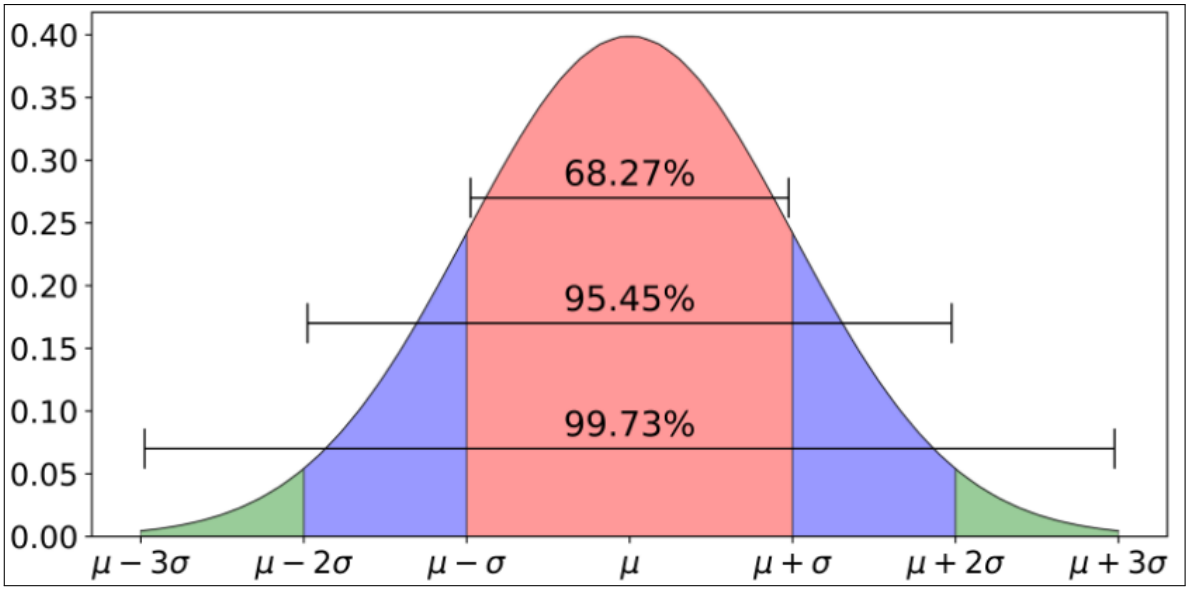


FIGURE 4.7: Rule 68-95-99.7 (Galarnyk, n.d.).

In Figure 4.8, it can be observed how rule 68-95-99.7 behaves on the values of accelerometer sensor in Z, so to eliminate noise from data set there are two options, eliminate values after three standard deviations from the mean, in case of considering that data set presents few anomalies or outliers, or eliminating values after two standard deviations in case of being sure that data set presents a great amount of noise in data, in this case most of data belong to normal driving behaviors so that only values found after three standard deviations from mean will be eliminated.

Fusion or data integration

When working with real-world data, it is possible that the required data is not found in same data set, in these cases, it is necessary to collect data from different sources and merge them into a single data set; this process is called Fusion or Data Integration. One of the most common problems of this process is redundancy.

This process was not applied in investigation because data was only captured through mobile device, so there is no problem of having more than one data source.

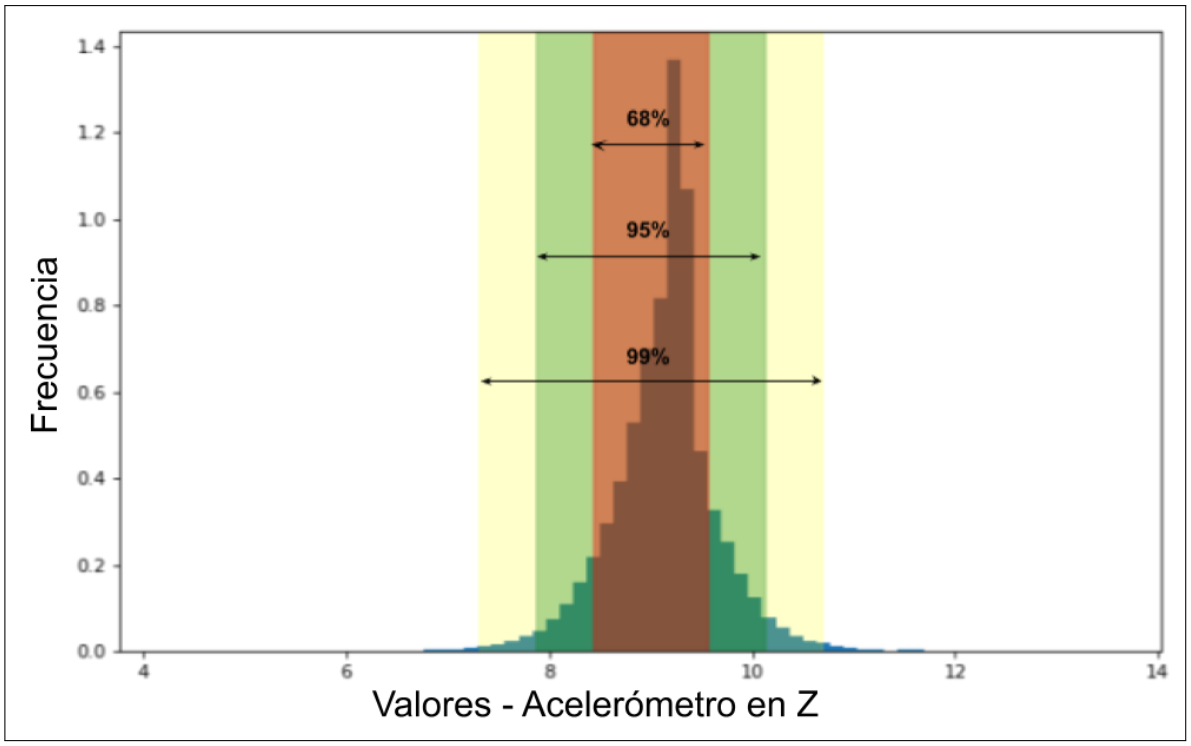


FIGURE 4.8: Result of applied Rule 68-95-99.7 on values' accelerometer sensors in Z (Own elaboration).

Data transformation

Data transformation is the process where the nature of data is changed, this process uses some strategies to be able to extract important information from data set; some of strategies for data transformation are:

- **Aggregation:** In this technique, the summary or aggregation operation is applied to data. For example: daily sales data can be used to calculate monthly and annual sales amount, and then add these calculated data to data set.
- **Discretization:** With this technique, raw values of a numerical attribute are constructed and replaced by interval values.
- **Attribute Construction/Feature Engineering:** This technique is useful for generating additional information from those data that are not representative enough by themselves, and may also be adequate when there are fewer features but still contain hidden information to extract.
- **Normalization / Standardization:** Normalization or standardization is defined as process of rescaling original data without changing its behavior or nature. A new limit is defined (generally between 0 and 1) and data is converted accordingly. This technique is useful in classification algorithms that involve neural networks or distance-based algorithms (for

example, KNN[1], K-Means [2], which is used for clustering. This algorithm is able to gradually learn how to group unlabeled values into groups by an analysis of average distance of these values.). Some standardization techniques are:

- *Normalization Min-Max:*s In this method, each input is normalized between defined limits:

$$x_{normalize} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

  It presents the problem that it compresses the input data between fixed limits, which are usually 0 and 1. This means that if there is noise, it will be expanded, which makes this method not suitable for stable signals.

- *Standard Scaler:* It is an alternative method to the variables' scaling, it consists in subtracting from each data the variable's average and dividing it by standard deviation.

$$x_{normalize} = \frac{x - x_{average}}{x_{std}} \tag{4.2}$$

  This method is suitable for normalizing stable signals, however, both mean and standard deviation are very sensitive to anomalous values. An alternative solution to this is the elimination of anomalies before normalization.

- *Scaling over the maximum value:* This method presents the idea of scaling data by dividing it by its maximum value.

- *Robust scaler:* Robust scaling involves eliminating the median and scaling data according to interquartile range (IQR). This method is robust for outliers.

Data reduction

This process is based on adoption of some strategies, such that analysis of reduced data produces the same information produced by original data. Some of strategies include: principal component analysis (PCA), selection of a attributes' subset, grouping and sampling among others.

Principal Component Analysis (PCA)

[1] **KNN**, is a classification algorithm (or regression) that, to determine the classification of a point, combines the classification of the nearest K points.

[2] textbfK-Means, is a clustering algorithm that attempts to divide a set of points into K groups; so the points in each group tend to be close to each other.

Principal Component Analysis (PCA) is a technique that is widely used for applications such as dimensionality reduction, lossy data compression, feature extraction and data visualization (Bishop, 2006).

PCA is an unsupervised and non-parametric statistical technique, which is used for dimensionality reduction. This technique is an important step because high dimensionality in the field of machine learning can lead to model's overfitting, thus reducing its ability to generalize, Bellman (2003) describes this phenomenon as the "Curse of Dimensionality". In addition, the use of this technique can directly improve performance of Machine Learning models.

PCA combines the input variables in a specific way, then gets rid of the "less important" variables and at the same time preserves the most valuable parts (or principal components [3]) of all variables.

When using PCA with a machine learning aproach, we follow these steps:

1. Divide *d*-dimensional dataset into a training, development and test set.
2. Standardize / Normalize the dataset according to training set.
3. Construct the covariance matrix.
4. Decompose the covariance matrix into its vectors and eigenvalues.
5. Order the eigenvalues by decreasing order to classify corresponding eigenvectors.
6. Select *k* eigenvectors that correspond to *k* largest eigenvalues, where *k* is the dimensionality of new entity subspace ($k <= d$).
7. Construct a projection matrix **W** from the "top" *k* eigenvectors.
8. Transform *d*-dimensional input training set **X** using the projection matrix **W** to obtain the new *k*-dimensional feature subspace.

Dataset division

Given that there is a dataset to generate Machine Learning model, it is divided into three parts: training[4], development[5] and testing set[6], however this is not a trivial task; because if it is not done correctly the result can be disastrous.

[3] **Principal Component**, it is a normalized linear combination of the original features in dataset.

[4] **Training set**, is the data sample used to fit the Machine Learning model.

[5] **Development set**, is the data sample used to provide an unbiased evaluation of a fitted model with the training set while adjusting the model hyperparameters.

[6] **Test set**, is the data sample used to provide an unbiased evaluation of a final fit of model in the training set.

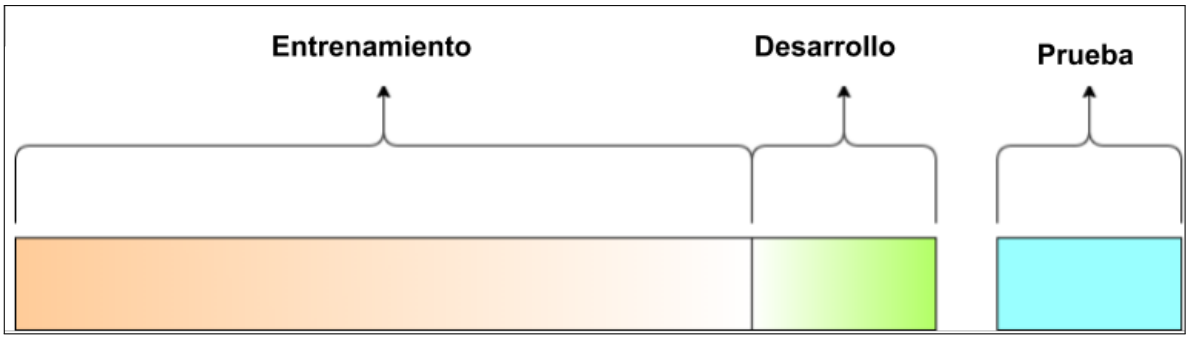


FIGURE 4.9: Division of the dataset (Own elaboration).

Training set	70%	21000
Development set	15%	4500
Test set	15%	4500
Data set	100%	30000

TABLE 4.1: Table of dataset's division (Own elaboration).

The work of Moindrot and Genthial (2018) saids that division of dataset has a great impact on productivity, so it is important that when choosing subsets they must have the **same distribution** and must be chosen randomly from dataset.

On the other hand, the size of development and test set must be large enough so that development and test results are representative for the performance of model. For large data sets (greater than one million), the development and test set can have around 10000 examples each, that is, 1% of the total data.

Other considerations to be taken into account in practice are:

- The division of training/development/test set must always be the same for all experiments, therefore a reproducible script must be available to create the training/development/test division.
- Must be probed that development and test sets came from the same distribution.

In the present investigation, the data set has 30,000 examples, so the division of data set will be as shown in Table 4.1.

Standardize / Normalize the dataset

In section 4.3.1 we have already described what is the standardization or normalization of data and some of scaling's types that exist; therefore this section will only be limited to the elaboration of an analysis to decide which scaling technique is the most suitable for data set, in Figure 4.10

a fraction of captured data set can be seen, that is the basis with which the comparative analysis will be carried out with the different scaling's types.

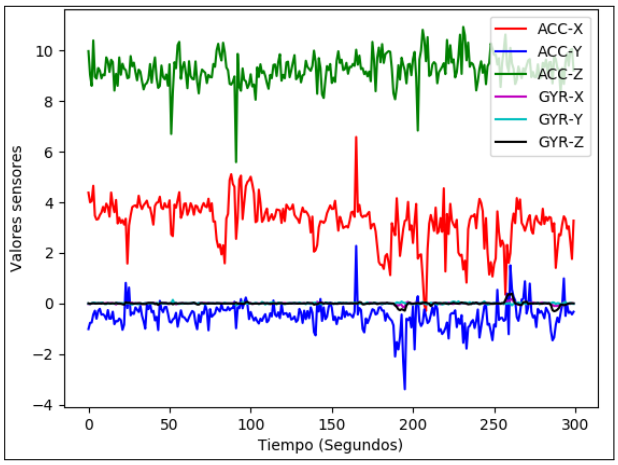


FIGURE 4.10: Visualization of the captured driving parameters (Own elaboration).

To test the different scaling's types, we fit them with data that no longer has noise or outliers from training set.

The first type of scaling that was performed on captured data set was **Min-Max Normalization**, which was performed between the limits 0 and 1, the results obtained are shown in Figure 4.11a, where it can be seen that the values of Accelerometer, in its three axes, are not deformed after being scaled with this technique and gyroscope values, which are more stable, become deformed, considering as stable data those data that are presented as a line in zero with few fluctuations; This deformation can be advantageous by making small curves that were previously imperceptible more visible, however it carries the danger that it could amplify existing noise in data that we could not eliminate in previous step to this task.

The second normalization technique that was applied to data was **standard scaling**, the result can be seen in Figure 4.11b, observing in detail the results can be seen to be very similar to those obtained with Min-Max normalization, the only differences that can be seen are that the new range of the data is broader with a mean of zero and values that oscillate mainly between 2 and -2, and that some of fluctuations presented by gyroscopes are expanded a little more.

For the third data normalization, technique applied was the **scaling on maximum value**, the results are totally different from those obtained previously, however it is clearly observed that accelerometer values do not show much change, with the difference that average of these values are different for each one, and gyroscope values behave as in the previous ones, this can be seen in figure 4.11c. This technique presents the worst results, since it does not leave data set in the same range, which complicates the work with them.

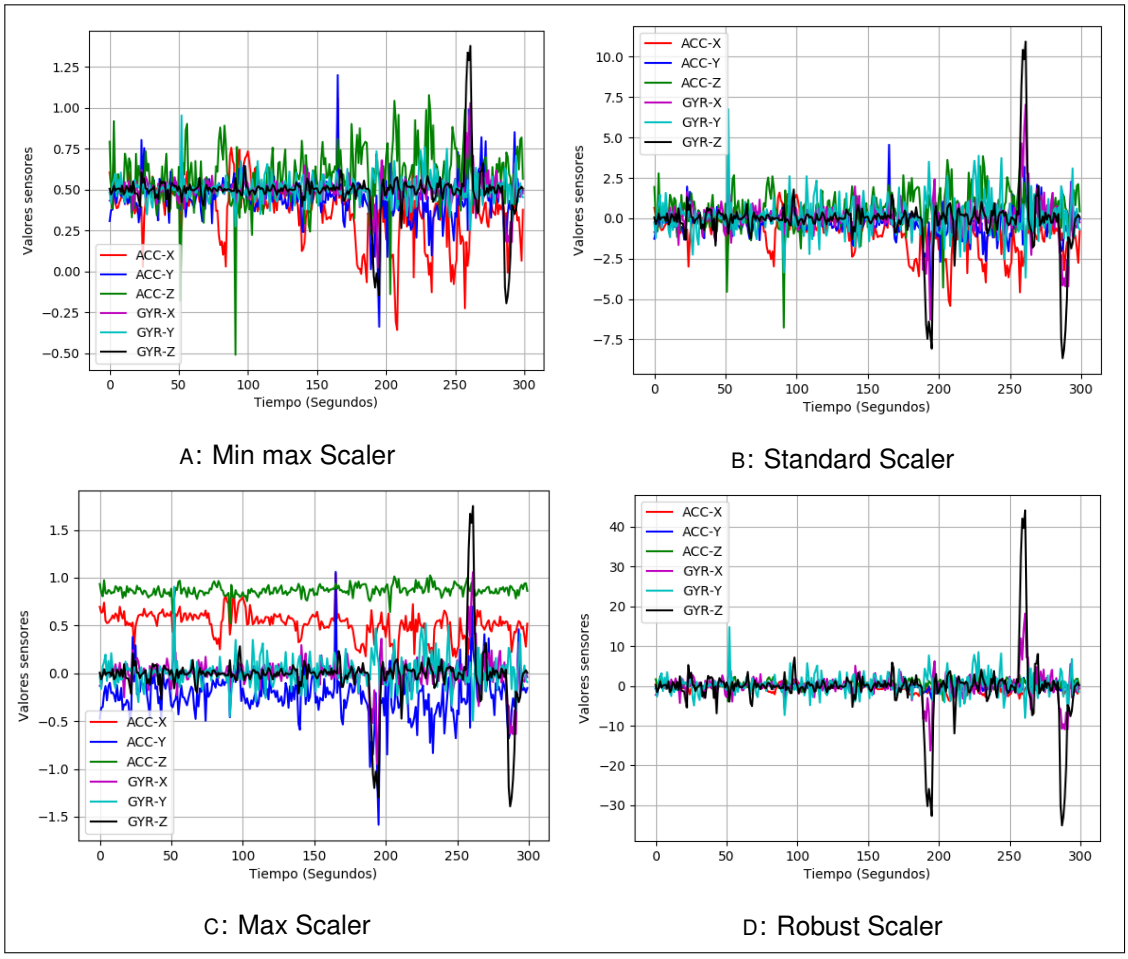


FIGURE 4.11: Graph resulting from applying different types of normalizations to data set (Own elaboration).

The last technique applied is **robust scaling**, its result can be seen in figure 4.11d, which can be considered quite similar to the first two techniques, with the difference that this scaling pronounces more the fluctuations of gyroscope's values, converting them even to a higher scale than that of accelerometer's values.

To decide the best normalization method that can be applied to the captured data, the **scaling method on the maximum value** will be completely discarded first because the results it presented were totally discouraging since the values after scaling did not share the same limits, on the other hand, the remaining three types of scaling are very similar, which complicates the choice of correct scaling, however, when using PCA, you must be very careful, because if you have a variable with a high standard deviation, this will have a greater weight in the calculation of the axis than a variable with a low standard deviation, so this can be an important decision parameter to choose the most appropriate type of scaling.

Table 4.2 shows the mean and standard deviation of each variable after the different types of scaling have been applied to the data. As can be seen in the scaling of the maximum and standard value, the standard deviations are very different, as for the Min-Max scaling, the standard deviations of each variable are almost the same since they all range between 0.166814 and 0.169386, which It suits very well for the analysis of main components, also mentioning that this technique is the one that best preserves the relevant information in the data set, so this technique was selected because it is the most convenient for this stage.

		ACC X	ACC Y	ACC Z	GYR X	GYR Y	GYR Z
Min-Max	Average	0.500369	0.500051	0.501112	0.497923	0.499352	0.500396
	St. Deviation	0.166847	0.166814	0.167300	0.168245	0.169386	0.166852
	Minimum	-0.999198	-0.675814	-1.041074	-0.681029	-0.319990	-18.004503
	Maximum	1.178265	1.654068	1.869867	25.395520	27.227514	2.345548
Standard	Media	-0.011266	-0.009209	-0.00570	0.013265	0.009458	0.005644
	St. Deviation	1.050384	1.086118	1.117567	2.224617	2.518566	2.076506
	Minimum	-9.451768	-7.665204	-10.307538	-15.575414	-12.173164	-230.291158
	Maximum	4.256420	7.504531	9.137616	329.221266	397.424938	22.968890
Maximum Abs. Value	Average	0.615065	-0.141064	0.842598	0.000519	0.001827	-0.001747
	St. Deviation	0.128546	0.286536	0.052784	0.334925	0.337715	0.332858
	Minimum	-0.540261	-2.160843	0.356031	-2.346416	-1.631746	-36.917638
	Maximum	1.137343	1.841185	1.274447	49.564032	53.291415	3.679194
Robust	Average	-0.028625	0.083278	0.008976	0.010491	0.031468	-0.040602
	St. Deviation	0.739033	0.672602	0.956915	5.752011	5.518751	8.397460
	Minimum	-6.670812	-4.657858	-8.811955	-40.295886	-26.663430	-931.368512
	Maximum	2.974050	4.736319	7.837925	851.216772	870.859223	92.823529

TABLE 4.2: Table with descriptive statistics of scaled data with different techniques (Own elaboration).

Apply PCA to dataset

Although some of steps in the internal workings of PCA are explained in detail at the beginning of this sub-section, there is a class called PCA implemented in SCIKIT-LEARN, which automates all following steps; however, it is necessary to determine how many characteristics (principal components) to maintain and how many to eliminate; for this task there are three commonly used methods, which will be described below.

- Arbitrarily select how many dimensions you want to keep, depending of use case. For instance, in a viewing approach you could choose 2 or 3 features.
- Calculate the variance ratio for each feature, choose a threshold, and add features until the chosen threshold is reached or exceeded.
- This method is closely related to previous one, because the variance ratio for each feature is calculated, the features are ordered by variance ratio and the cumulative variance ratio explained is plotted while it maintains the features (this diagram is known as a screen diagram). You can choose how many features to include by identifying the point at which a new feature is added that has a significant drop in variance from previous feature, and choose the number of features that exist up to that point. This method is usually known as ”Find elbow”.

For the development of this investigation, both the first and last method of those listed above were discarded, this because the first method does not have the certainty that the quantity of features chosen is descriptive enough for dataset; while the last method does not have a mathematically precise definition, since it only finds ”elbow”, so it also removes control of amount of total variability in data that is finally obtained.

Once the methods that do not conform to the requirement of this work have been discarded, the only remaining option is the second method, thus being the one that will be applied for present work. Therefore, the total variability threshold to be preserved in data set was defined as 90%, later using the PCA class of SCIKIT-LEARN, the variance of each characteristic is calculated and then the results are plotted (See Figure 4.12).

And finally, it must be defined how many principal components retain at least 90% of data’s variance. Figure 4.12 indicates that selecting 4 components can preserve about 97.7% of data’s total variance, selecting 3 components conserves around 92.3% and selecting 2 conserves 85% of variance; therefore, the use of 3 features was decided, which will conserve 92.3% of dataset’s total variance.

In various researches (Zenon, 2011; Klos & Waszczyszyn, 2011) it was proven that the application of PCA as pre-processing of the data set is an important stage, because it not only serves to compress the input data, but which also provides a satisfactory improvement in accuracy of Machine Learning models; reason why this technique is part of pre-processing stage of this work.

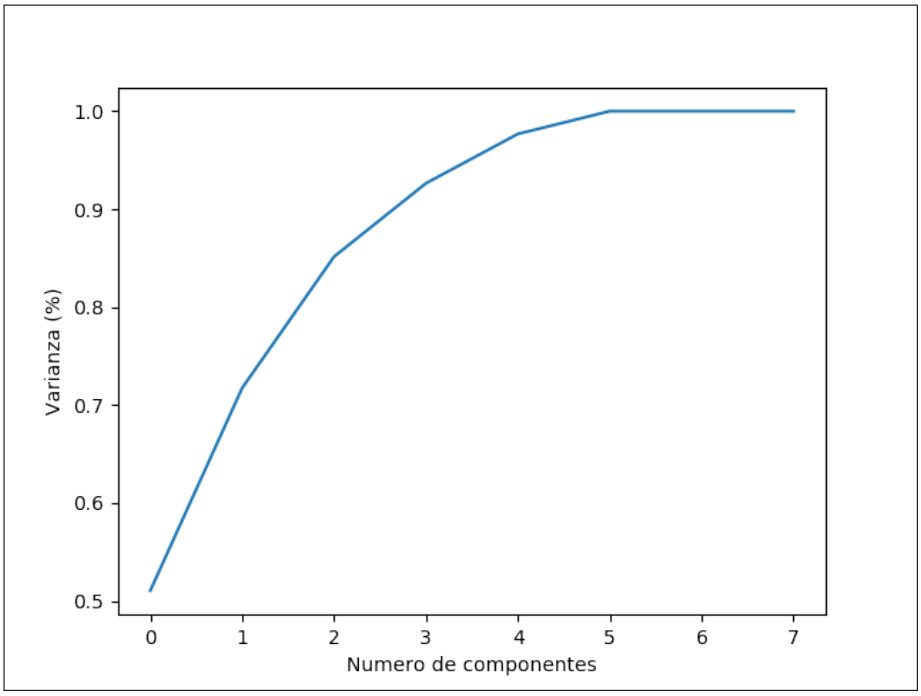


FIGURE 4.12: Variance's graph vs.components' number (Own elaboration).

Chapter 5

GENERATION OF THE ANOMALY DETECTION MECHANISM

This chapter describes the process that was followed to generate the conduction anomaly detection mechanism. As reviewed in Chapter 3, in this document, an outlier detector with a semi-supervised approach is proposed; however, before delving into the proposed method, a description of the development environment with which the experiment was used should be carried out, in addition to reviewing dataset available in this study.

5.1 Development environment

The experiment of this study was developed on a laptop with following characteristics:

- Intel Core i5-5200U 2.2GHz processor (c / TB 2.7 GHz).
- 8 GB of RAM.
- ArchLinux Operating System version 4.15.15-1-ARCH (64 bits).

It is important to clarify that methods' types used in this study are usually much more efficient in computers that have a graphics processing unit (GPU), since this unit allows parallel processing. The code developed in this work was written in PYTHON, an interpreted programming language that emphasizes code simplicity and readability, as well as being powered with support of powerful scientific libraries such as NUMPY, SCIPY, OPENCV, KERAS, MATPLOTLIB, SEABORN, etc. as for the experiments' development and the generation of detection mechanism, Jupyter Notebook was used, which is a local Python-based web application that allows you to view and execute documents that contain source code and equations. The versions of tools used are detailed below.

- **Python:** 3.6.5
- **Jupyter:** 4.3
- **Keras:** 2.2.2
- **Tensorflow:** 1.11.0

- **Scikit-learn:** 0.19.1
- **Matplotlib:** 2.0.2

5.2 Normal and anomalous dataset

In Chapter 4 capturing's process and preparing dataset was described, as well as its division into training/development/test set; however, it should be noted that in that chapter was focused only on the normal dataset.

Although there is a large amount of normal data, it is necessary to collect samples corresponding to anomalies, in order to validate the method proposed in this project. Therefore, the capture of a anomalous dataset was carried out, which is compose according to Table 5.1.

Anomaly's type	**No. anomalies**	**No. data**
Zig Zag turns	5	105
Right and left turns at high speed	7	35
Dry brakes	6	24

TABLE 5.1: Table of anomaly dataset (Own elaboration).

As mentioned in previous paragraph, anomaly set was captured to validate the proposed method, therefore this set was labeled as positive (with label 1) and normal dataset as negative (with label 0).

5.2.1 Generation of time series

For the generation of anomaly detector model, it was decided to go beyond a simple point anomalies' detection and thus be able to detect contextual or collective anomalies; due to this, the use of data in time series is required.

Data captured by mobile device is dependent on chronometric time in which it was captured (one data per second); therefore, first step is generate small time series fractions. In Figure 5.1 results of different time series' sizes are presented, observing these results in first instance, the time series that has two steps are discarded; because they are not descriptive enough. As for remaining time series, it is not possible to define yet what is the correct number of steps, therefore, it will be a parameter to optimize in the different experiments that will be carried out in following sections. It should be noted that domain of this variable is between 3 and 5 steps.

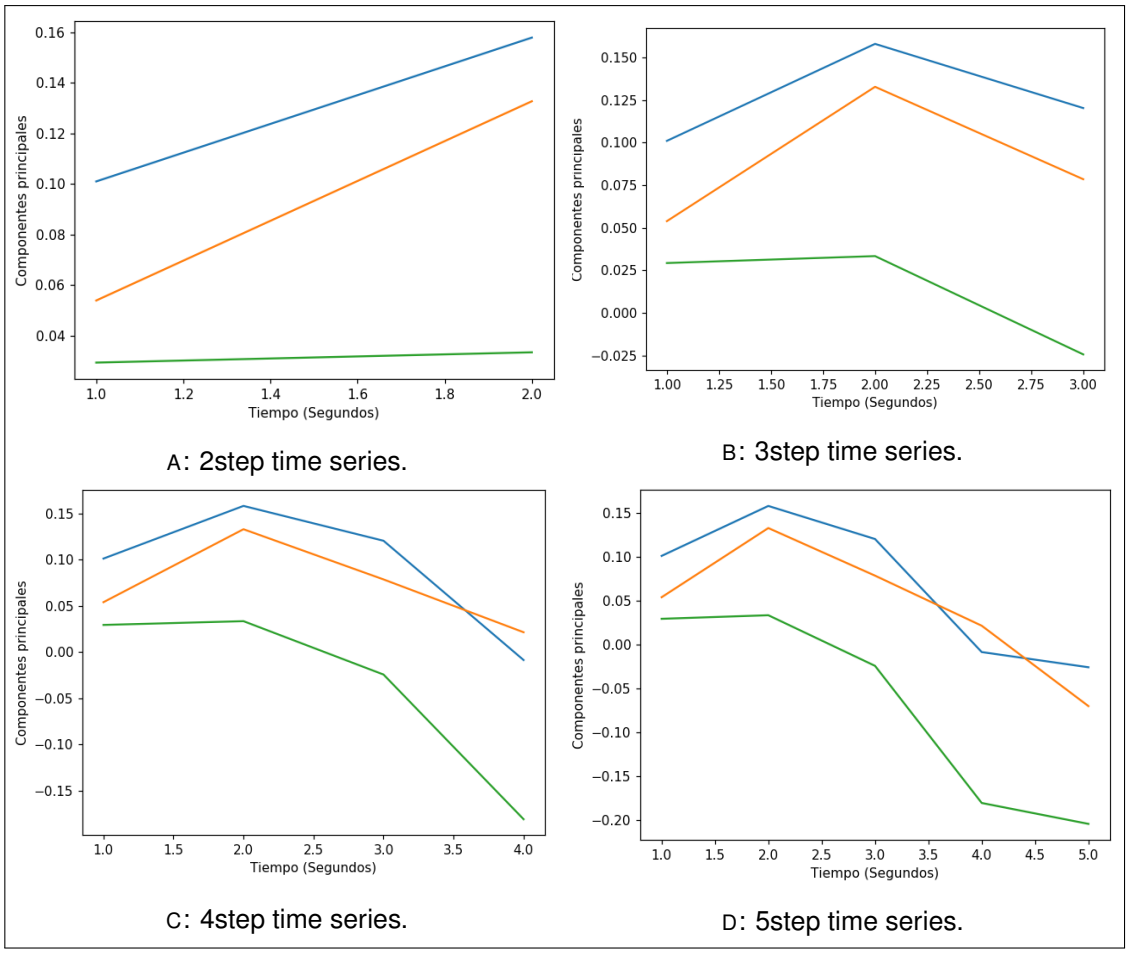


FIGURE 5.1: Results from different sizes of time series (Own elaboration).

5.3 Anomaly detection model

Present investigation proposes a method of detecting conduction anomalies following a semi-supervised approach, which consists of two components: a **normal behavior's model** and a **method for outliers' detection**.

Therefore, a comparison was made between 3 different detection methods, and according to each's performance, the best option was chosen. Table 5.2 presents the three different methods that were compared; where it can be seen that in all cases an autoencoder is used as a normal behavior's model, in this way, following sections will describe the choice of autoencoder and the choice of one of three different proposed anomaly detection methods.

Method	Description
AE_T	Detection method based on autoencoders and thresholding.
AE_IF	Detection method based on autoencoders and application of Isolation Forest.
AE_OC-SVM	Detection method based on autoencoders and application of One-Class SVM.

TABLE 5.2: Table of compared methods (Own elaboration).

5.3.1 Normal behavior model

This stage is one of the most important parts of this work, because performance of anomaly detection model depends largely on precision of this stage.

Model architecture

As mentioned in previous section, at this stage an autoencoder will be used as a model adjusted to normal driving behavior. So the autoencoder was trained with normal dataset, so that model learns to generate only the classes that are considered normal and, hopefully, it will have problems to reconstruct anomalies, because these samples were not presented during training.

To do this, different architectures were tested, first the simplest way that is based only in the use of dense (fully connected) layers, then tests were carried out with convolutional networks and finally with recurrent networks making specific use of LSTM layers. For each network's type, tests was performed with 3 different types of input, that is, a different number of steps (between 3 and 5) were tested in the time series. Therefore, 9 different experiments were carried out, of which for each network's type one stood out (using networks' precision as a type of evaluation to develop comparisons).

Table 5.3 shows the network that obtained the best result of all Dense Networks that were tested, this network corresponds to network that was fed with 3step sequences. This network has an input layer, an Flattening layer (Flatten) this because input layer receives a two-dimensional input, a set of dense layers (Dense) that compress information of input data to later rebuild them, and finally output layer is only a layer to modify the shape of output (Reshape); another important point to highlight is that inner layers use elu as activation function and last dense layer uses a tangential activation function ($tanh$), this is because data set, after obtaining principal components, is found in range $(1, -1)$.

		Dense Architecture			
		NN_33			
		Type	**Output**	**Activation**	**# Parameters**
PCA	**3**	Input	(3,3)		0
		Flatten	9		0
		Dense	8	elu	80
		Dense	5	elu	45
		Dense	8	elu	88
		Dense	9	tanh	81
		Reshape	(3,3)		0

TABLE 5.3: Dense architecture for a 3steps sequence and 3 principal components (Own elaboration).

On the other hand, table 5.4 presents network that obtains the best precision of all Convolutional Networks that were tested, this network, like previous one, corresponds to network that was fed with 3step sequences. Its architecture consists of an input layer (Input), a combination of one-dimensional convolution layers (Conv1D) and grouping (MaxPooling1D) until data is compressed to a dimension of (2,4), then a set of convolutional layers and upstream sample (Upsampling1D) to decode compressed information. It should be noted that this network also uses tangential activation function in its last layer for reasons explained in previous paragraph.

		Convolutional Architecture			
		CNN_33			
		Type	**Output**	**Activation**	**# Parameters**
PCA	**3**	Input	(3,3)		0
		Conv1D	(3,2)	elu	20
		MaxPooling1D	(2,2)		0
		Conv1D	(2,4)	elu	28
		MaxPooling1D	(1,4)		0
		Conv1D	(1,6)	elu	54
		UpSampling1D	(3,6)		0
		Conv1D	(3,3)	tanh	57

TABLE 5.4: Convolutional architecture for a 3steps sequence and 3 principal components (Own elaboration).

Table 5.5 shows network that obtained the best precision of all Recurring Networks tested, as previous cases, this network is fed with 3step sequences. This network has an input layer (Input), two LSTM layers, one that returns its sequences and one that does not, then comes a resizing layer, later two LSTM layers, and finally a container (TimeDistributed) of a dense layer.

		Recurrent architecture			
		RNN_33			
		Type	**Output**	**Activation**	**# Parameters**
PCA	**3**	Input	(3,3)		0
		LSTM	(3,9)	elu	468
		LSTM	6	elu	384
		Reshape	(3,2)		0
		LSTM	(3,3)	elu	72
		LSTM	(3,9)	elu	468
		TimeDistributed(Dense)	(3,3)	tanh	30

TABLE 5.5: Recurrent architecture for a 3steps sequence and 3 principal components (Own elaboration).

It is important to emphasize that the resizing, grouping, ascending sample and containers layers were only used to control the correct compression and decompression of autoencoders, that is why their operation is not detailed in depth.

Autoencoder evaluation

Previously, the best representatives by type of network were presented; now we will proceed to the evaluation and comparison of these 3 autoencoders' types, with aim of choosing the architecture that best suits normal driving behavior.

Chapter 3 presented the different evaluation's types that exist, at this stage the most appropriate evaluation's type is model **precision**, due to the fact that there is a large balanced data set (due that we only have normal driving behaviors that correspond to "Normal" class). The evaluation's results of the three networks' types are shown in Table 5.6; this table presents the precision, logarithmic loss and execution time of each autoencoder according to test set; observing these results, it can be seen that the two best networks are dense network **NN_33** and the recurrent network **RNN_33** with accuracies of approximately 90%, in addition to presenting a relatively low loss value compared to **CNN_33** network.

Network	**Precision**	**Loss**	**Execution time**
NN_33	0.9000740711953905	0.003956934471097257	26us/step
CNN_33	0.8437777761353387	0.006740666443275081	31us/step
RNN_33	0.8899259290695191	0.003611267575787173	101us/step

TABLE 5.6: Evaluation of the NN_33, CNN_33 and RNN_33 networks (Own elaboration).

On the other hand, the biggest difference between the two best networks (**NN_33** and **RNN_33**) is execution time since first one is only 26 seconds/step and second one is 101 seconds/step,

due to these similarities between two networks it necessary to verify visually the reconstruction results of each network's type, so that the most suitable network can be chosen for this problem.

Figure 5.2 shows autoencoders' results of seven sequences taken randomly from test set, at the top of each figure is the input sequence and at the bottom the model reconstruction, as could already be expected the **CNN_33** network presents the worst results, which causes that network to be discarded; as for remaining two networks, the **NN_33** network presents reconstructions very similar to input sequences, with some small errors; on the other hand, **RNN_33** presents slightly more notable errors than those obtained by **NN_33**. Therefore, it was concluded that **NN_33** network adjusts better to normal driving behavior, in addition to having the great advantage of having a much shorter execution time than that of **RNN_33**, which is really important for real-time systems as well as those with limited execution resources, as is the case in this work.

Once the normal behavior model has been defined, the method of detecting outliers can be chosen.

5.3.2 Anomaly detection method

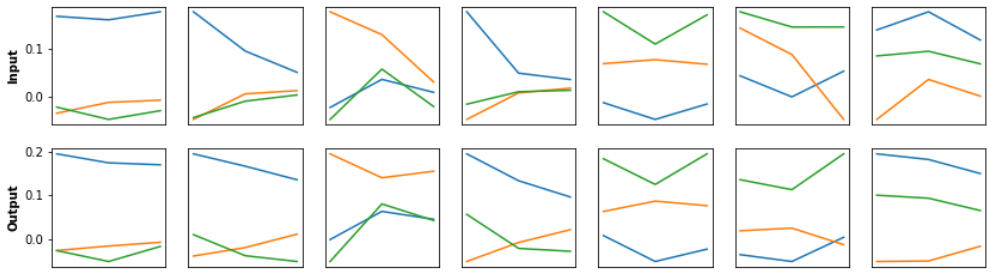
At the beginning of this section, three different approaches to anomaly detection were defined: thresholding, applying isolation forests, and finally applying One-Class SVM.

Thresholding

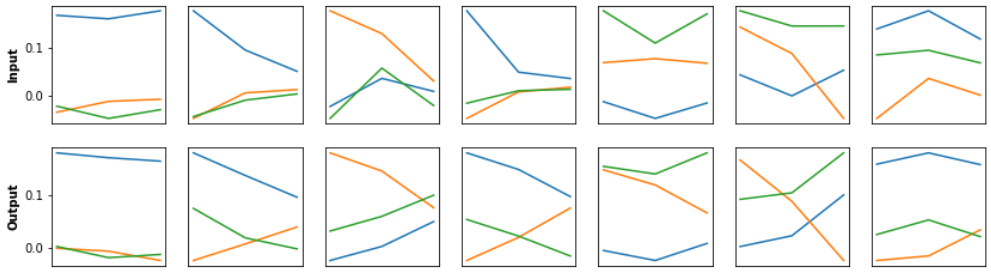
This technique is based on the definition of a threshold to determine if reconstruction error obtained by autoencoder (normal behavior model) is high enough to be considered an outlier. Therefore reconstruction error equation must first be defined for model. In the present work, reconstruction error is defined according to equation 5.1, where x_i represents the real value (autoencoder's input) and $\hat{x_i}$ represents the value obtained by autoencoder (autoencoder's output).

$$\text{Reconstruction error} = S_z = |x_i - \hat{x_i}|^2 \tag{5.1}$$

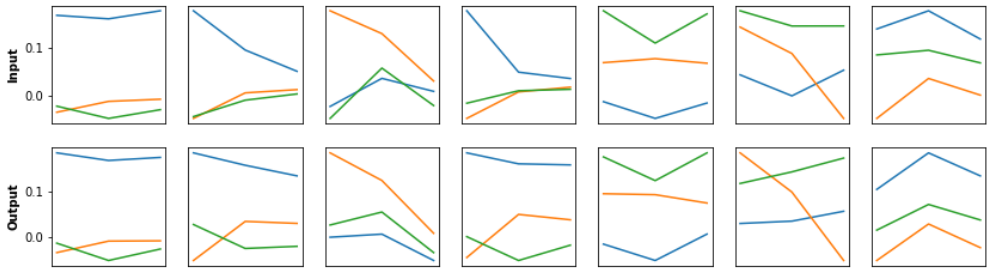
Figure 5.3 shows the curve of reconstruction errors obtained with normal behavior model for normal samples set.



A: Results of **NN_33** network



B: Results of **CNN_33** network



C: Results of **RNN_33** network

FIGURE 5.2: Results (Own elaboration).

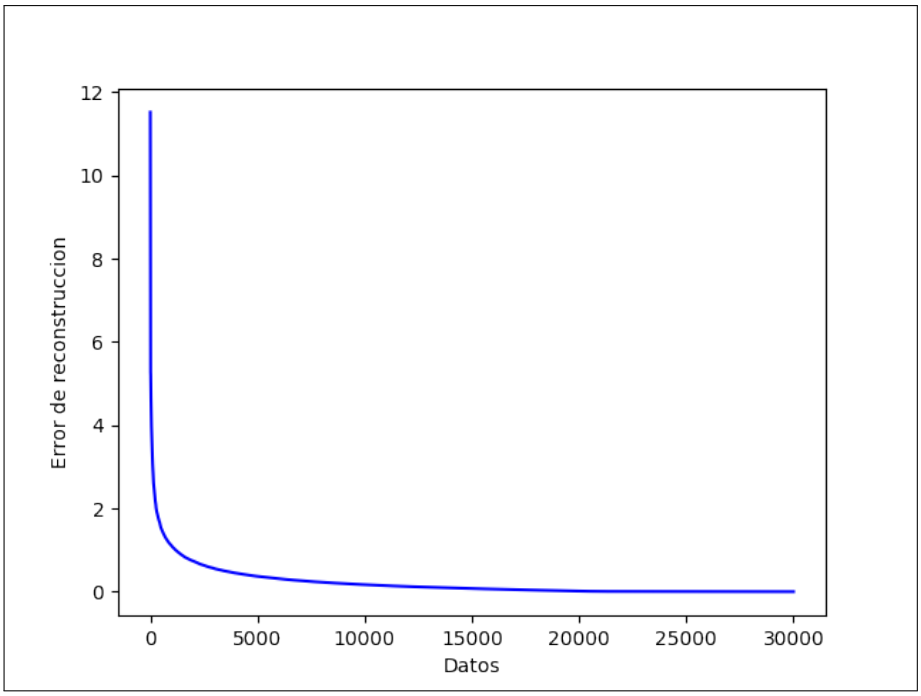


FIGURE 5.3: Curve of reconstruction values obtained with the normal behavior model (Own elaboration).

Once the reconstruction equation has been defined, a threshold must be defined capable of detecting as many anomalies as possible. This task can be made simple in a supervised learning environment, however automating this task in an unsupervised learning context is a challenge that can be difficult to overcome. In present work a technique based on finding an *elbow point* of a curve was used, which in this case the curve is built based on autoencoder reconstruction errors.

There are different ways to find the elbow point, however in this work a Python tool was used, which automates this task, called Kneedle. This tool returns the inflection point of curve function obtained by values' set provided x and y, it should be noted that elbow point is maximum curvature's point, on the other hand this tool has a sensitivity parameter (S), this parameter allows adjusting how aggressive you want to be when detecting elbows, the smallest values for S detect elbows faster, while the largest are more conservative, that is, S is a measure of how many "flat" points are expect to see unmodified data curve before declaring an elbow.

Thus, in present project, experiments were carried out with different sensitivity values to find the most appropriate elbow for data set with which we worked. Figure 5.4 shows different elbows found for sensitivity values provided (values between 0 and 2).

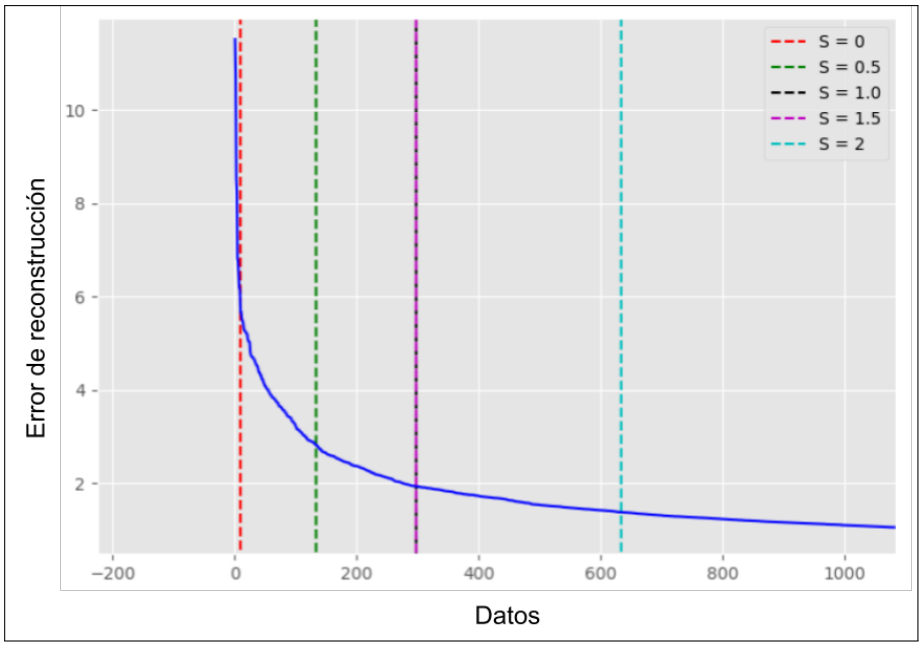


FIGURE 5.4: Results of obtaining elbows with different Sensitivity values, for reconstruction values obtained with normal behavior model (Own elaboration).

Once the elbows were obtained, each of them was evaluated. Table 5.7 shows threshold, the confusion matrix values, the sensitivity and specificity for each elbow. Confusion matrix values are result of applying the threshold of each elbow to reconstruction errors obtained from total data set (normal and anomalous data set equivalent to 44204 data).

S	Threshold	TP	VN	FN	FP	Sensitivity	Specificity
0.0	5.665	92	43993	72	47	0.5610	0.9989
0.5	2.806	111	43814	53	226	0.6768	0.9949
1.0	1.920	120	43562	44	478	0.7317	0.9891
2.0	1.369	128	43100	36	940	0.7805	0.9787

TABLE 5.7: Assessment of anomalies' detection for each elbow obtained with the different sensitivity values (Own elaboration).

According to results shown in Table 5.7, it can be said that when the threshold is smaller the sensitivity (proportion of anomalies correctly detected as anomalies) increases, however, in turn it reduces the specificity (proportion of normal values correctly detected as normal values). Therefore, an intermediate point must be found, where as many anomalies as possible can be detected and the number of false positives (normal data that are detected as anomalies) reduced as much as possible. Thus, the most appropriate threshold for proposed objective was 2.806, since 111 abnormalities out of 164 are detected with this threshold and false positives are approximately twice the outliers detected.

It follows that, to automatically detect threshold, the use of 0.5 as parameter S is the most appropriate, however if you want to increase percentage of anomalies detection at the cost of increasing the number of false positives, you can use a higher value 0.5 for S and if you want the fewest possible false positives, a value less than 0.5 should be used.

Isolation Forest

Before presenting how experiments with this algorithm will be carried out, it is necessary to illustrate its operation in more detail. Therefore Figure 5.5 depicts how an anomalous data point is expected to quickly become isolated with the use of this algorithm, whereas a normal data point needs more partitions to be isolated.

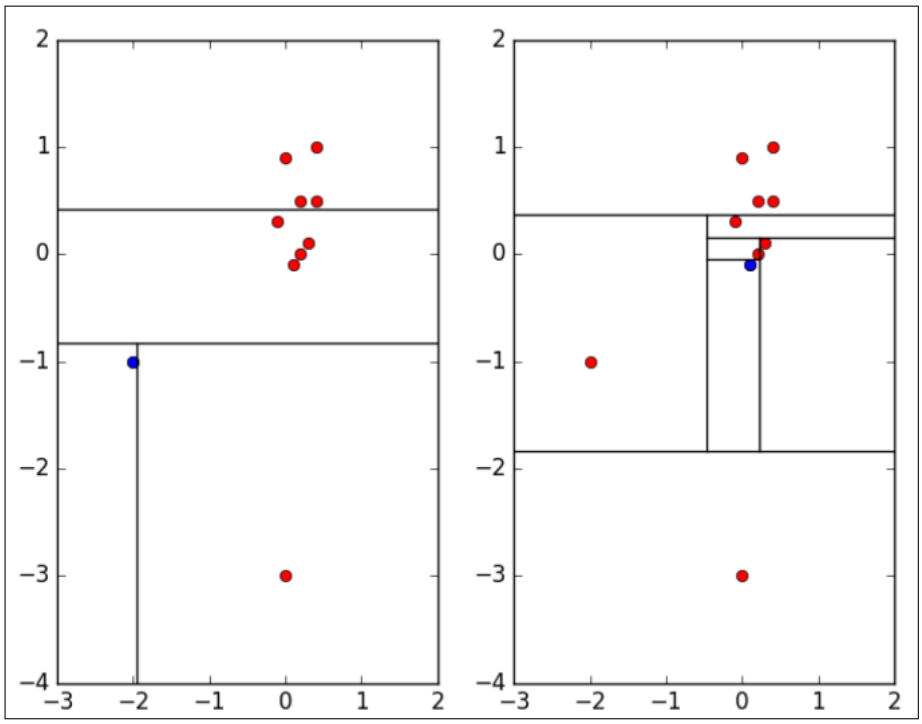


FIGURE 5.5: The figure on left shows the isolation of an anomaly, requiring only three partitions. On right, isolation of a normal point requires six partitions (Wolpher, n.d.).

Once isolation forest's operation has been briefly detail, the different approaches to experiments to be carried out with Isolation Forest can be continued.

There are two approaches that can be done with this technique; first one trains the model with compressed values of autoencoder's encoder and second one trains with autoencoder reconstruction errors. The following is a graph (See Figure 5.6) of autoencoder (normal behavior model) in order to have a better understanding of how experiments will be carried out both for isolation forests and for One-Class SVM.

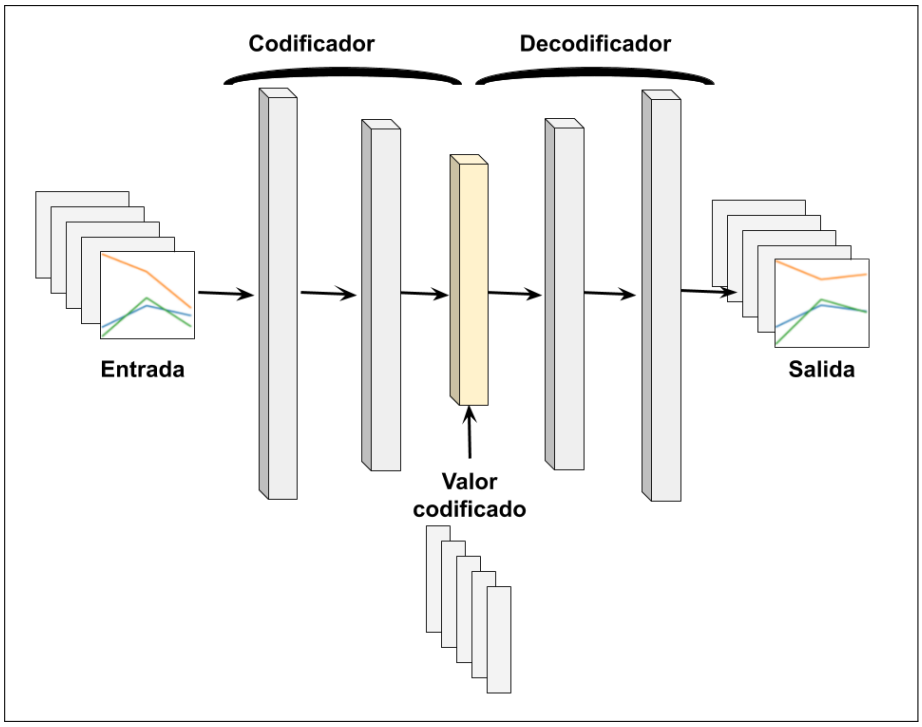


FIGURE 5.6: Graphical representation of normal behavior model or autoencoder (Own elaboration).

- ***Isolation forest for encoded values:*** This technique trains an isolation forest model with the encoded values (using autoencoder's encoder, see Figure 5.6) of normal training set. For experimentsm IsolationForest class of SCIKIT-LEARN was used, this class has a parameter called CONTAMINATION which serves to define how much of data set is contaminated, that is, it defines how much of training data can be outliers; in present investigation, several tests were carried out with different values for contamination parameter. Table 5.8 presents the results, where it is evident that none of results is encouraging, since the number of anomalies detected is very low for the three cases with which it was experimented.

Contamination (C)	TP	VN	FN	FP	Sensitivity	Specificity
0.0025	3	43944	161	96	0.0183	0.9978
0.0050	17	43817	147	223	0.1037	0.9949
0.0075	17	43738	147	302	0.1037	0.9931

TABLE 5.8: Evaluation of anomalies' detection using Isolation forest for compressed values (Own elaboration).

- ***Isolation forest for reconstruction errors:*** For this technique, isolation forest training was performed with the difference of the input values with the values obtained by autoencoder (See Figure 5.7), it should be clarified that the difference mentioned above will also be called Error reconstruction in this and the next subsection. Results of this technique

for different contamination values are presented in Table 5.9, where these results can be considered as optimal, since they range between 62 and 67% of correct anomaly detections, in addition to presenting a really high specificity of approximately 99%, which means that these models have a low false positive rate.

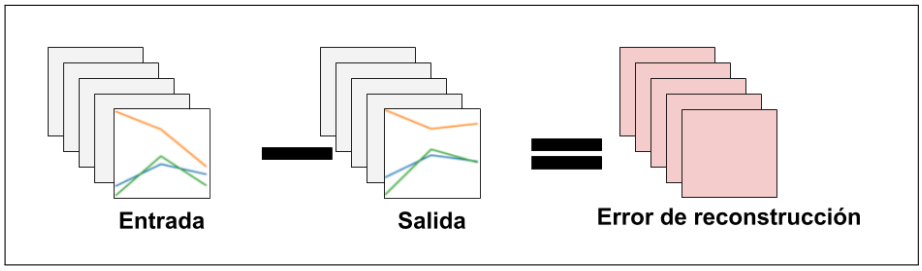


FIGURE 5.7: Graphical representation of reconstruction error used for the training of isolation forests and One-Class SVM (Own elaboration).

Contamination (C)	TP	VN	FN	FP	Sensitivity	Specificity
0.0025	103	43898	61	142	0.6280	0.9968
0.0050	108	43792	56	248	0.6585	0.9944
0.0075	111	43688	53	352	0.6768	0.9920

TABLE 5.9: Evaluation of anomaly detection using Isolation forest for reconstruction errors (Own elaboration).

One-Class SVM

In the same way as in isolation forests, two different types of experiments were performed with One-Class SVM, each of which is detailed below.

- ***One-Class SVM for encoded values:*** A one-class SVM model be trained for compressed values obtained by autoencoder; experiments were performed using OneClassSVM class of SCIKIT-LEARN, where there are different parameters that can be customized, for present investigation different kernels were tested, thus obtaining the results shown in Table 5.10, where clearly none of obtained results could be taken into account to be the method of detection conduction anomalies since the sensitivity in none of cases is greater than 50%.

Kernel	TP	VN	FN	FP	Sensitivity	Specificity
rbf	49	41746	115	2294	0.2988	0.9479
poly	56	22532	108	21508	0.3415	0.5116
sigmoid	13	42378	151	1662	0.0793	0.9623

TABLE 5.10: Anomaly detection evaluation using One-Class SVM for compressed values (Own elaboration).

- ***One-Class SVM for reconstruction errors:*** Like one of experiments that were performed with Isolation Forest, in this technique the reconstruction errors are used (See Figure

5.7) to carry out training of the One-Class SVM model. As in experiments carried out in previous technique, different tests were carried out with different types of kernel, the follow Table 5.11 presents results obtained by experiments.

Kernel	TP	VN	FN	FP	Sensitivity	Specificity
rbf	134	41887	30	2153	0.8170	0.9511
poly	97	1559	67	42481	0.5915	0.0354
sigmoid	123	1683	41	42357	0.7500	0.0382

TABLE 5.11: Anomaly detection evaluation using One-Class SVM for autoencoder's reconstruction error (Own elaboration).

Observing results of Table 5.11, it can be seen that the sensitivity, or number of anomalies detected correctly, was significantly increased, however, it drastically reduced the specificity since in some cases it only reaches 3.5%, which is very far from objective pursued in the present work.

Evaluation of the anomaly detection method

Once the different types of experiments had been carried out, the best exponent of each type was evaluated, in order to choose the most suitable one for this investigation. Below is a Table 5.12 with results of the best representatives for each type of technique.

Method name	TP	VN	FN	FP	Sensitivity	Specificity
Thresholding with S=0.5	111	43814	53	226	0.6768	0.9949
Isolation Forest for reconstruction errors with C=0.0075	111	43688	53	352	0.6768	0.9920
One-Class SVM for reconstruction errors with kernel RBF	134	41887	30	2153	0.8170	0.9511

TABLE 5.12: Comparison of the best anomaly detection methods (own elaboration).

Obviously, the best anomaly detection result is the one obtained by SVM model for a class with a sensitivity of 81.7%, however, this method has the disadvantage of having a high number of false positives, that is, for each anomaly detected, there will be approximately 13 false positive alerts, which is a very high value; and it is the main reason why this method is discarded.

Due to this, there are only two methods to compare, where both results are very similar; since these have a sensitivity of 67.68%, on the other hand, the specificity has a small variation between both techniques, giving a slightly better result for the thresholding technique with 99.49% compared to 99.20%.

At this point you can choose either of these two methods due to the similarities that both present. For reasons of simplicity, the isolation forest method was chosen in this study since this method makes it easier to detect anomalies because one can specify the amount of contamination expected from dataset, this is much more advantageous than the search for elbows with thresholding method since it has the disadvantage that it is really complex to define the threshold when we use an unsupervised approach, as is case in this stage, in addition to the fact that definition of threshold depends a lot on how clean or contaminated data set is, making the correct treatment more complex when applying this method.

Once the best methods for conforming the outlier detection mechanism have been chosen, this mechanism is formalized by means of a graph (See Figure 5.8), which provides a visual representation of the flow of proposed anomaly detector; since it is important to know how it is composed and how it works, especially before carrying out its respective evaluation.

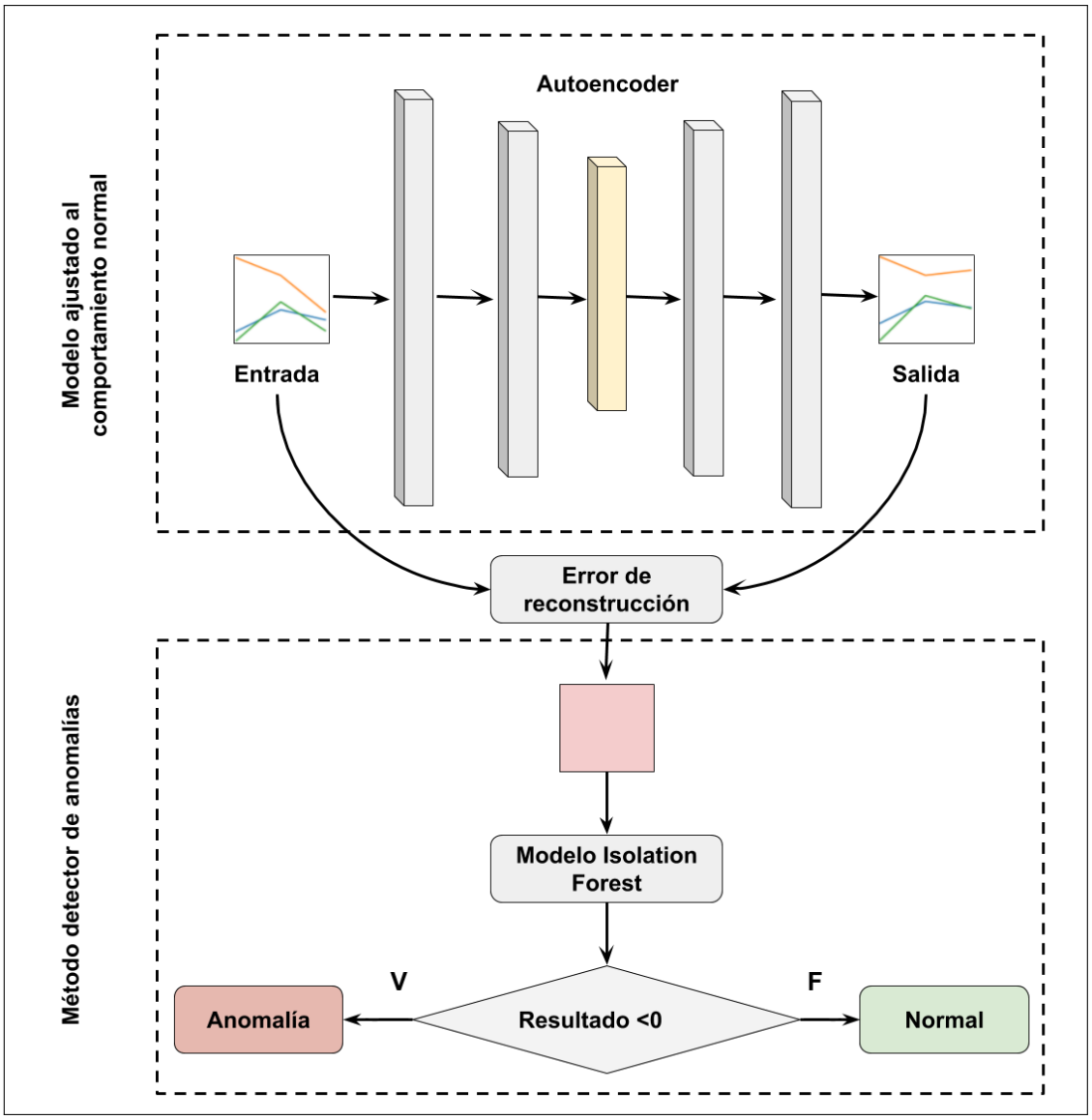


FIGURE 5.8: Anomaly detection mechanism (Own elaboration).

Looking at Figure 5.8, the components that compose the anomaly detection mechanism can be clearly seen: the **normal behavior model** and the **anomaly detection method**. This mechanism works in a very simple way, firstly the autoencoder is provided with an input sequence with 3 steps for 3 principal components (it should be noted that input data has been previously pre-processed), this autoencoder returns reconstruction as output of the input, with which the reconstruction error is obtained (difference between real input and reconstructed value), this value is in turn the input of the isolation forest model, which can return two types of values (-1 , 1); when this model returns the value 1, it means that input provided corresponds to a value considered normal, and if it returns -1, it means that this input is an anomaly, thus ending the flow of detection mechanism proposed in this work.

Chapter 6

RESULTS AND EVALUATION

Purpose of this chapter is to present the evaluation's results of proposed anomaly detection mechanism, to later show some of results obtained with it.

6.1 Performance evaluation

This study will evaluate the effectiveness of anomaly detection techniques from following two perspectives:

- The ability of approach to distinguish between normal and anomalous data.
- The efficiency of method according to time required to train model and time spent during detection process.

6.1.1 Evaluation in detection performance's terms

Before evaluating the mechanism proposed in this study, it is important to highlight what:

- **Normal behavior model** was trained with 21000 samples, during 50 iterations, with 4500 samples that were used to validate model during training stage, and finally test set with which the final evaluation of this model was made is compose with 4500 samples.
- On the other hand, **anomaly detection method** was trained with all dataset that were used in the development of normal behavior model generation, that is, with 30,000 samples.

To evaluate the anomaly detection mechanism proposed in this study, following criteria were used: detection rate and false positive rate. Detection rate is defined as the number of detected anomalies divided by total number of anomalies. False positive rate is defined as the number of "normal" series that are classified as anomalies divided by the total number of "normal" series. It is important to clarify that outliers' set, available in this research, was not used for training the proposed method; however, this set was used to validate its precision, therefore data set with which this mechanism is validated has 44204 data.

Table 6.1 presents confusion matrix obtained by the proposed mechanism, from which following statements can be obtained:

- Upper left entry in matrix shows that 111 anomalies out of 164 were correctly labeled, that is, that 67.68% of anomaly samples were correctly recognized.
- Bottom row shows that 43688 of 44040 data were correctly labeled as normal values, i.e. 99.20%. Therefore false positive rate for normal class is $10099.20\% = 0.80\%$.

		Prediction	
		Anomaly	**Normal Class**
Real	**Anomaly**	111	53
	Normal Class	352	43688

TABLE 6.1: Confusion matrix, for the anomaly detection mechanism (Own elaboration).

These results are a great advance for conduction anomalies' detection with a semi-supervised approach, since without having samples of outliers in training, it is difficult to have a higher precision; considering also that one of the most important added values presented by this research work, is to be able to generate a personalized model for each agent's type, which is really outstanding, because related work that was reviewed (prior to carry out this investigation) does not have an exemplary that contemplates a semi-supervised approach, much less with models that are adjusted and personalized for each agent.

6.2 Results

Results of this study provide an essential contribution in the field of automation of early abnormal behaviors' detection in car driving; However, these present an overview of the behavior of the proposed model, so it is necessary to carry out a more specific analysis of said behavior with each type of anomaly presented in data set, as well as the analysis of those values that were detected wrongly as anomalies (false positives). The results of previously mentioned analyzes are presented below.

6.2.1 Anomalies' detection of zig zag type

This anomaly corresponds to a common behavior for agents who drive under the influence of alcohol; it consists of driving that presents sudden zig zag movements, that is, constant changes of direction and at a relatively high speed. Below are some of results that were obtained with anomaly detection mechanism proposed in this research work.

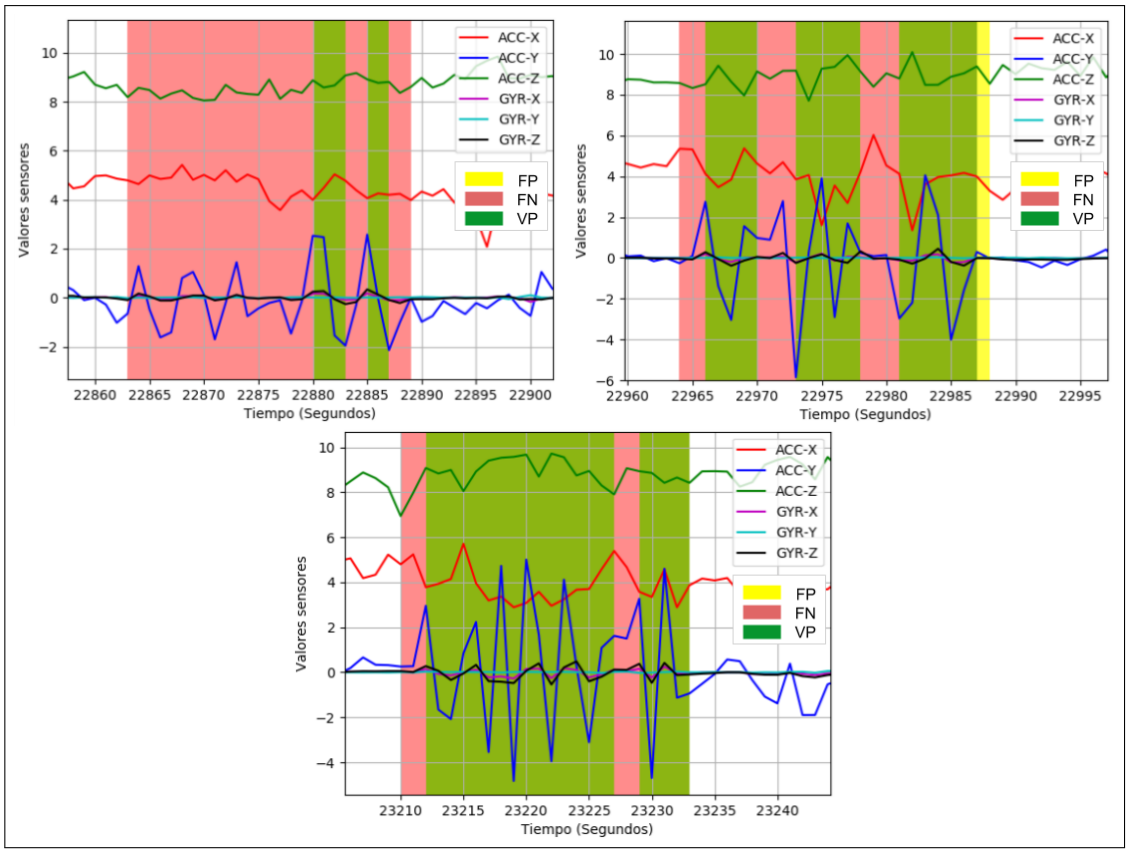


FIGURE 6.1: Results of anomalies' detection of zig zag type (Own elaboration).

Before carrying out the results obtained's analysis, it should be clarified that those sections of following graphs that appear in red are the values that belong to anomalies' set that were not correctly detected (False negatives), the sections in yellow correspond to false positives and finally the green sections are true positives, that is, those values that were correctly detected as anomalies.

As seen in Figure 6.1, upper left image presents a large number of false negatives, this is because the movements' oscillations in Zig Zag were not sudden enough, compared to others, on the other hand, image upper right presents a moderate number of false negatives and one false positive specimen, although result does not seem quite good really, it is good, since many of false negatives are among correctly detected values, which would lead to a correct alarm generation despite not detecting as an outlier all anomalous data, a similar example is presented in image below, although in this case no false positives are detected.

In order to formalize results for Zig Zag anomalies, in table 6.2 it can be seen that 69 anomalies out of 105 were correctly detected, that is, 65.71%.

Zig Zag turns		
TP	**FN**	**Total**
69	36	105

TABLE 6.2: Results of anomalies Zig Zag type (Own elaboration).

6.2.2 Anomalies' detection for turns at high speed

This type of anomalies are usually common in agents who drive under the influence of alcohol, drugs or with an altered emotional state, these data are considered anomalies since turns are normally made by lowering of vehicle's speed, and when performing these types of acts an agent is prone to being the cause of a traffic accident.

Figure 6.2 shows results obtained for high speed turns anomalies, the three images present very similar results, all have a section in initial part that is presented as false negative, that is, they have a number of data that are not correctly detected, later they have a block of true positives, and finally, two of three images have a false positive specimen after the anomaly. Although these types of anomalies are not completely detected, they all have a section that is detected as an anomaly, which is enough to generate an alarm in a timely manner.

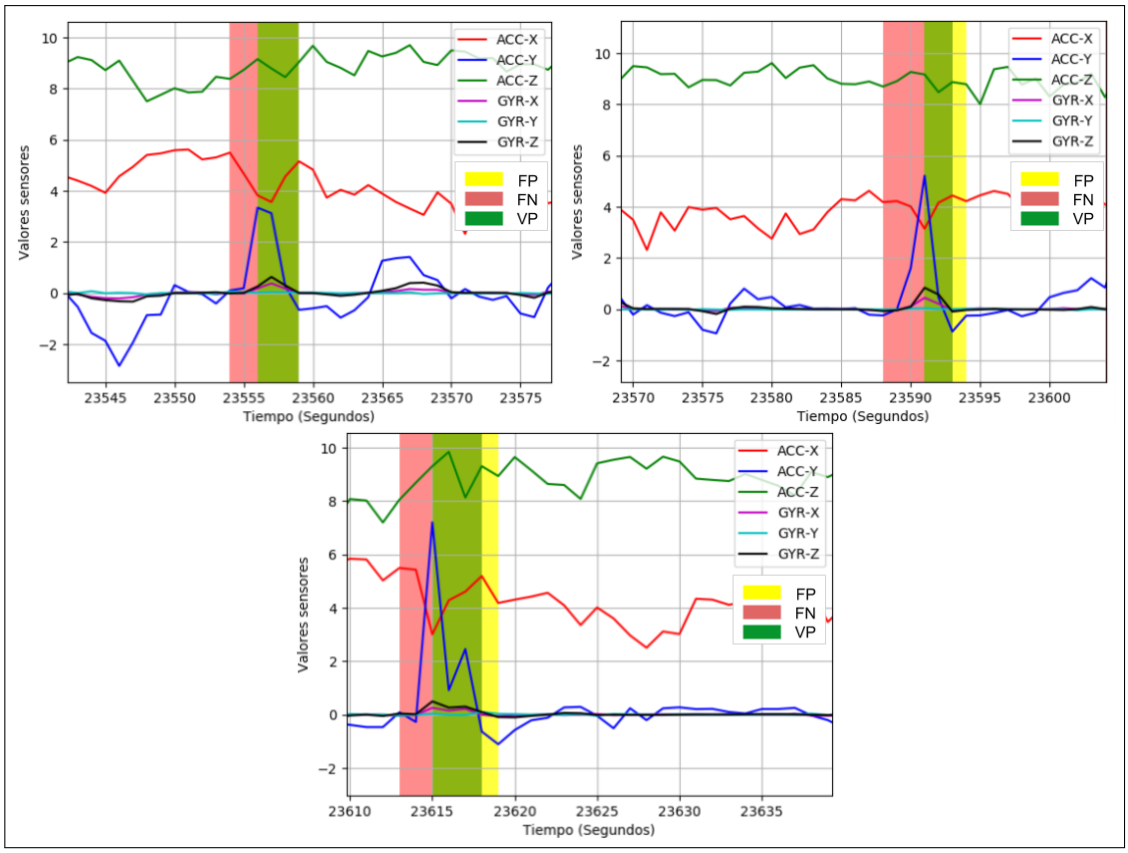


FIGURE 6.2: Results of anomalies' detection of the high speed turns type (Own elaboration)

Table 6.3 presents the general results obtained for high speed turns anomalies, where of 35 anomalies, 23 were correctly detected, that is, 65.71%.

High speed turns		
TP	**FN**	**Total**
23	12	35

TABLE 6.3: Results anomalies' detection of the type High speed turns (Own elaboration).

6.2.3 Anomalies' detection of the type dry brakes

This type of anomaly is usually one of the most common outliers that exist, since they not only occur under the influence of alcohol, drugs or mechanical failures, but also occur in contexts of driver distraction due to the cell phone's use or other distraction's type, when a pedestrian or pet appears suddenly in the lane that the agent drives, among other cases.

Detection's results of this type of anomaly are presented in Figure 6.3, where, as in previous case, this type of anomaly presents a section of false negatives, later a group of anomalies correctly detected and finally false positives; with which it is enough to generate alerts in a timely manner and thus be able to avoid any possible traffic accident.

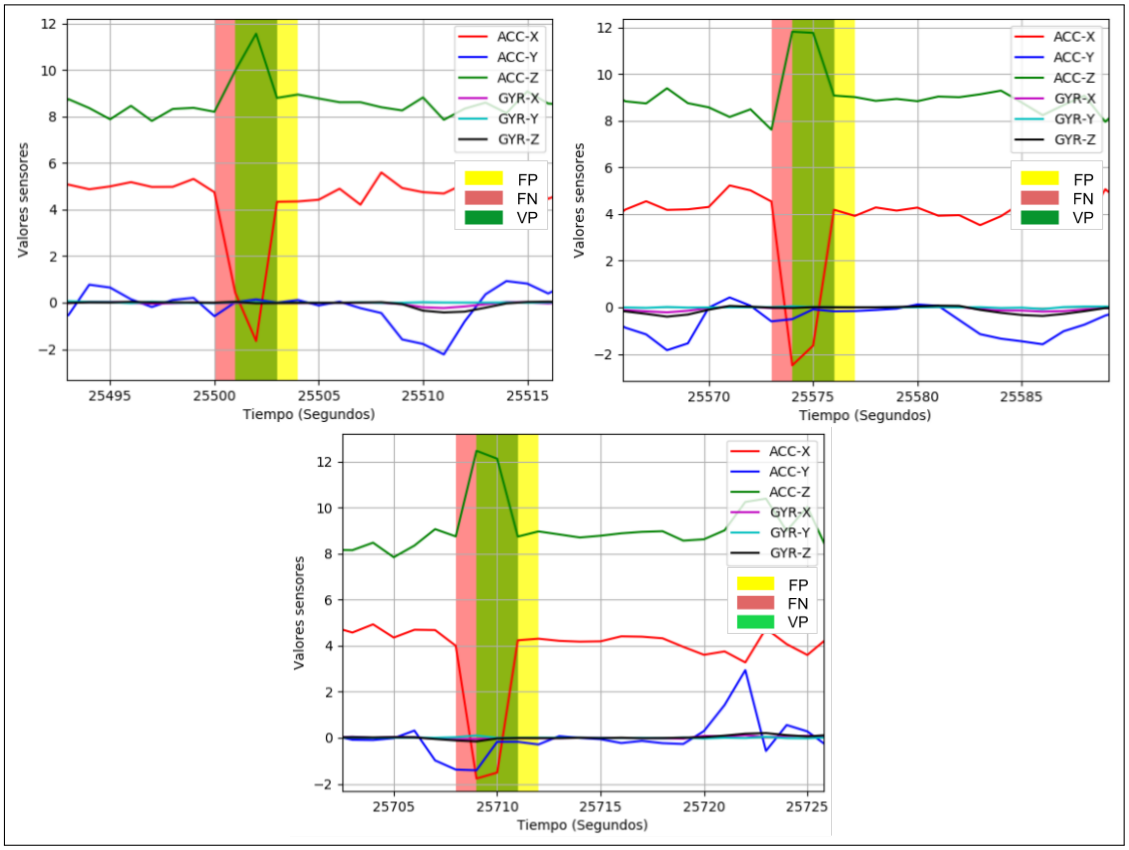


FIGURE 6.3: Results of anomalies' detection of type dry brakes (Own elaboration).

Below in Table 6.4 it can be seen that 19 anomalies out of 24 were correctly detected (79.17%), thus being this anomaly's type that has the highest detection percentage.

Dry brakes		
TP	**FN**	**Total**
19	5	24

TABLE 6.4: Results of anomalies' detection of the type dry brakes (Own elaboration).

6.2.4 Detection of false positives

Just as a large number of anomalies were detected using this mechanism, a considerable proportion of false positives were also detected, that is, normal values that were erroneously detected as outliers.

In the same way that it is important to know how this method detects anomalies, it is also important to know in which cases the proposed model fails; Figure 6.4 shows some cases where the model fails, that is, this figure presents some false positives' examples. Figure 6.4 clearly illustrates that these false positives are generally presented in isolation, that is, one or two values erroneously detected as anomalies continuously, which is a different behavior from true outliers, since these present a detection minimally three outliers continuously.

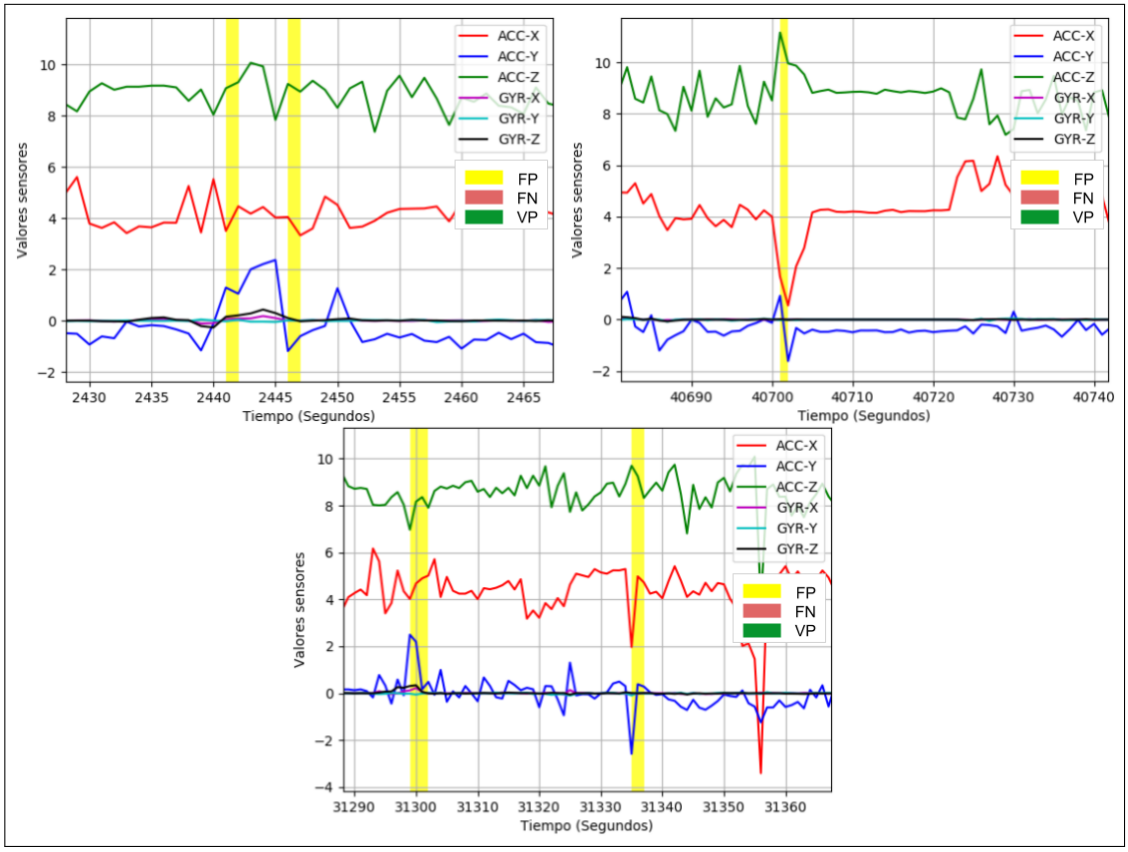


FIGURE 6.4: Results of false positives' detection (Own elaboration).

Chapter 7

CONCLUSIONS AND FUTURE WORK

Después de haber realizado el procedimiento descrito en los anteriores capítulos, con el objetivo de comprobar la hipótesis establecida en la presente investigación, se generó un mecanismo (modelo) capaz de detectar anomalías de conducción. De esta forma se puede decir que se ha cumplido a cabalidad con los objetivos propuestos en esta investigación. A continuación se presentará las conclusiones a las que se llegó, así como también aquellas nuevas ideas e inquietudes que surgieron durante el proceso de desarrollo, las cuales podrían mejorar los resultados obtenidos por el presente trabajo.

7.1 Conclusiones

El objetivo fundamental de este trabajo de investigación fue desarrollar un mecanismo capaz de detectar anomalías de conducción, tal que, se aporte con una solución para alertar de forma oportuna el hallazgo de patrones anómalos en la conducción de agentes, ya sean humanos o autónomos, independizando cada modelo según la experiencia y el ambiente por el que recorre cada agente.

Así pues, el principal aporte de este estudio consiste en la implementación de un mecanismo capaz de identificar anomalías a partir de los datos de conducción normal de cada agente, sin intervención humana, es decir, el modelo detector no requiere que un humano intervenga para generarlo, sin embargo, este puede ser optimizado por medio del ajuste del hiperparámetro *Contaminación* con el fin de definir cuán sensible a las anomalías será dicho detector. Por otra parte, se puede decir que el mecanismo de detección de este trabajo de investigación, además de ser novedoso, es uno de los pocos trabajos que se realizaron con un enfoque *"semi-supervisado"*, ya que la mayoría de los trabajos realizados a la fecha fueron realizados mediante un enfoque supervisado.

Las conclusiones que se derivan de este trabajo de investigación se hicieron en base a los diferentes experimentos realizados, dichas conclusiones se exponen a continuación.

- Se comprueba, a partir del análisis de resultados de este estudio, la capacidad con la que cuentan los sensores inerciales de un dispositivo móvil para representar correctamente

el movimiento de un automóvil y de esa manera ser capaz de alimentar, con un previo pre-procesamiento, un mecanismo de detección de anomalías.

- En este trabajo se compararon diferentes arquitecturas de redes neuronales para generar el modelo del comportamiento normal, donde la red más simple logró los mejores resultados tanto en precisión como en el tiempo empleado durante el proceso de predicción; demostrando así, que no siempre las redes más complejas interpretan mejor los conjuntos de datos.
- Por otra parte, se comparó diferentes técnicas para definir un método de detección de anomalías adecuado al contexto de la presente investigación, donde por la simplicidad de su entrenamiento y por su robusto resultado se optó por la elección de la técnica de bosques de aislamiento, con un valor de 0.0075 para el hiperparámetro *Contaminación*.
- Integrando el modelo del comportamiento normal y el método de detección de anomalías, los cuales sólo fueron entrenados con el conjunto de datos ”normal”, se logra la creación de un mecanismo capaz de identificar valores atípicos de la conducción de cada agente.
- Finalmente se evaluó la capacidad del mecanismo de detección, mediante el conjunto de evaluación el cuál presenta muestras anómalas, dando como resultado la correcta detección del 67.68% de las muestras, que presentan anomalías en el conjunto de evaluación, así como también presenta una tasa de tan sólo 0.80% de muestras normales detectadas como anomalías. Siendo un gran avance en el ámbito de la detección de anomalías con un enfoque semi-supervisado.

Este trabajo de investigación antes que presentar una solución final sienta las bases para el desarrollo de sistemas de detección de anomalías de la conducción de los agentes, mediante el uso de técnicas de Inteligencia Artificial, resaltando la capacidad y alcance que conlleva este estudio, ya que no sólo se enfoca en la conducción de agentes humanos, sino que es igual de capaz de ser aplicado en un enfoque de conducción autónomo.

7.2 Trabajos futuros

Una vez concluido el trabajo de investigación, se considera interesante investigar sobre diferentes aspectos de la detección de anomalías y se propone:

- Agregar la velocidad del vehículo como un nuevo parámetro del conjunto de datos, debido a que esto podría brindar un mejor entendimiento del comportamiento normal de conducción, así como también de las anomalías.
- En lugar de trabajar con los datos en crudo, usar la diferencia entre un dato capturado en el tiempo t y un dato capturado en $t-1$ ($dif_t = dato_t - dato_{t-1}$), la aplicación de éste pre-procesamiento de datos podría maximizar la detección de aquellas anomalías que presentan elevadas diferencias entre los datos consecutivos.
- Validar el modelo con nuevos tipos de anomalías como por ejemplo: derrapes, choques, giros en U a alta velocidad, entre otros. Esto debido a que el estudio se limitó al reconocimiento de sólo tres tipos de anomalías por la dificultad y peligro que conlleva su captura.

- Extender el modelo para que sea capaz de determinar no sólo una anomalía sino también el tipo al que dicha anomalía pertenece.
- Migrar el modelo del comportamiento normal de keras a Tensorflow 2.0.
- Implementar un sistema de información para monitorear las anomalías mediante el mecanismo de detección propuesto en el presente trabajo.

BIBLIOGRAFY

References

0.22, S. (n.d.). *Novelty and outlier detection.* https://scikit-learn.org/stable/modules/outlier_detection.html. (Último acceso en 19 de diciembre de 2019)

Alashwal, H., Bin D., S., & Othman, R. (2006, 01). One-class support vector machines for protein protein interactions prediction. *Int J Biomed Sci*, *1*.

Araujo, R., Igreja, A., R., D. C., & Araujo, R. (2012, 6). Driving coach: A smartphone application to evaluate driving efficient patterns. *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV); Alcala de Henares, España.*(3–7), 1005-–1010. Retrieved from https://www.researchgate.net/publication/261309792_Driving_coach_A_smartphone_application_to_evaluate_driving_efficient_patterns

Bellman, R. E. (2003). *Dynamic programming*. Courier Dover Publications ISBN.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw*, *5*, 157–166. Retrieved from https://ieeexplore.ieee.org/document/279181/authors#authors

Bhoyar, V., Lata, P., Katkar, J., Patil, A., & Javale, D. (2013, 3–4). Symbian based rash driving detection system. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, *2*, 124-–126. Retrieved from https://www.ijettcs.org/Volume2Issue2/IJETTCS-2013-03-28-046.pdf

Bishop, M. C. (2006). *Pattern recognition and machine learning*. Springer Science+Business Media, LLC.

Boonmee, S., & Tangamchit, P. (2009, 5). Portable reckless driving detection system. *Proceedings of the 6th IEEE International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology; Pattaya, Chonburi, Tailandia.*(6–9), 412-–415. Retrieved from https://ieeexplore.ieee.org/abstract/document/5137037

Chen, Z., Yu, J., Zhu, Y., Chen, Y., & Li, M. (2015, 6). D3: Abnormal driving behaviors detection and identification using smartphone sensors. *Proceedings of the 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON); Seattle, WA, USA.*, 20–25. Retrieved from http://www.winlab.rutgers.edu/~yychen/papers/D3-Abnormal%20Driving%20Behaviors%20Detection%20and%20Identification%20Using%20Smartphone%20Sensors.pdf

Cho, K., Merriënboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014a). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv*, *1406.1078*. Retrieved from https://www.aclweb.org/anthology/D14-1179.pdf

Dang-Nhac, L., Duc-Nhan, N., Thi-Hau, N., & Ha-Nam, N. (2018, 4). Vehicle mode and driving activity detection based on analyzing sensor data of smartphones. *Sensors*, *18(4)*, 1036. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5948751/

Dauphin, Y. N., Fan, A., Auli, M., & Grangiera, D. (2017, 9). Language modeling with gated convolutional networks. *arXiv*, *1612.08083v3*(8). Retrieved from https://arxiv.org/pdf/1612.08083.pdf

de Salud (OMS), O. M. (n.d.). *Informe de la situación mundial de la seguridad vial 2015.* https://www.who.int/violence_injury_prevention/road_safety_status/2015/Summary_GSRRS2015_SPA.pdf?ua=1. (Último acceso en 19 de diciembre de 2019)

Elman, J. (1990). Finding structure in time. *Cognitive Science*, *14(2)*, 179–211. Retrieved from https://crl.ucsd.edu/~elman/Papers/fsit.pdf

Eren, H., Makinist, S., Akin, E., & Yilmaz, A. (2012, 6). Estimating driving behavior by a smartphone. *Proceedings of the Intelligent Vehicles Symposium. Alcalá de Henares, España.*(3–7), 234—239.

Ferreira, J., Carvalho, E., Ferreira, B., De Souza, C., Suhara, Y., Pentland, A., & Pessin, G. (2017). Driver behavior profiling: An investigation with different smartphone sensors and machine learning. *PLoS One*, *12(4)*, e0174959. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5386255/

Galarnyk, M. (n.d.). *Explaining the 689599.7 rule for a normal distribution.* https://towardsdatascience.com/understanding-the-68-95-99-7-rule-for-a-normal-distribution-b7b7cbf760c2. (Último acceso en 19 de diciembre de 2019)

Guo, Y., Liao, W., Wang, Q., Yu, L., Ji, T., & Li, P. (2018). Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. *Proceedings of Machine Learning Research*, *95*, 97–112. Retrieved from http://proceedings.mlr.press/v95/guo18a/guo18a.pdf

Hawkins, S., He, H., Williams, G., & Baxter, R. (2002, 9). Outlier detection using replicator neural networks. *International Conference on Data Warehousing and Knowledge Discovery*, 170-–180. Retrieved from https://togaware.com/papers/dawak02.pdf

Hochreiter, S., & Schmidhuber, J. (1997, 11). Long short-term memory. *Neural Comput*, *9(8)*(15), 1735—1780. Retrieved from https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

Hsu, A., & Griffiths, T. (2010). Effects of generative and discriminative learning on use of category variability. Retrieved from https://cocosci.princeton.edu/tom/papers/discgencat.pdf

Jayesh, B. (n.d.). *The artificial neural networks handbook: Part 4.* https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e. (Último acceso en 19 de diciembre de 2019)

Jing, Y., & Guanci, Y. (2018, 03). Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. *Algorithms*, *11*, 28.

Johnson, D., & Trivedi, M. (2011, 10). Driving style recognition using a smartphone as a sensor platform. *14th International IEEE Conference en Intelligent Transportation Systems (ITSC)*, 1609—1615. Retrieved from http://cvrr.ucsd.edu/publications/2011/Johnson_ITSC2011.pdf

Klos, M., & Waszczyszyn, Z. (2011). Modal analysis and modified cascade neural networks in identification of geometrical parameters of circular arches. *Computers & Structures*, *89*,

581–589. Retrieved from http://cames.ippt.gov.pl/index.php/cames/article/view/110

Koh, D. W., & Kang, H. (2015, 7). Smartphone-based modeling and detection of aggressiveness reactions in senior drivers. *Proceedings of the IEEE Intelligent Vehicles Symposium; Seoul, Korea.*(1), 12–17. Retrieved from https://ieeexplore.ieee.org/abstract/document/7225655

Kridalukmana, R., Yan-Lu, H., & Naderpour, M. (2017). An object oriented bayesian network approach for unsafe driving maneuvers prevention system. *12th International IEEE Conference*. Retrieved from https://opus.lib.uts.edu.au/bitstream/10453/122196/4/633634.pdf

Lecun, Y., Bengio, Y., & Hinton, G. (2015, 5). Deep learning. *Nature*, *521(7553)*(27), 436—444. Retrieved from https://doi.org/10.1038/nature14539

Lecun, Y., Jackel, L., Boser, B., Denker, J., Graf, H., Guyon, I., ... Hubbard, W. (1998). Handwritten digit recognition : Applications of neural networks chips and automatic learning. *Proceedings of the IEEE*, *86(11)*, 2278-–2324. Retrieved from http://yann.lecun.com/exdb/publis/pdf/lecun-89c.pdf

Mass, A., Hannun, A., & Ng, A. (2013). Rectifier nonlinearities improve neural network acoustic models. *International Conference on Machine Learning (icml)*. Retrieved from https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf

Michael, A. (2015). *Neural networks and deep learning*. Determination Press. Retrieved from http://neuralnetworksanddeeplearning.com/index.html

Moindrot, O., & Genthial, G. (2018, 1). *Splitting into train, dev and test sets.* https://cs230-stanford.github.io/train-dev-test-split.html. (Último acceso en 16 de octubre de 2019)

Muhammad, R. (n.d.). *Convolutional neural network. in a nut shell.* https://engmrk.com/convolutional-neural-network-3/. (Último acceso en 19 de diciembre de 2019)

Olah, C. (n.d.). *Understanding lstm networks.* http://colah.github.io/posts/2015-08-Understanding-LSTMs/. (Último acceso en 11 de octubre de 2019)

Özler, H. (n.d.). *Accuracy trap! pay attention to recall, precision, f-score, auc.* https://medium.com/datadriveninvestor/accuracy-trap-pay-attention-to-recall-precision-f-score-auc-d02f28d3299c. (Último acceso en 16 de octubre de 2019)

Pascanu, R., Mikolov, T., & Bengio, Y. (2013, 6). On the difficulty of training recurrent neural networks. *In Proceedings of the International Conference on Machine Learning; Atlanta, GA, USA*(16–21), 1310-–1318. Retrieved from http://proceedings.mlr.press/v28/pascanu13.pdf

Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann Publishers, Inc. Retrieved from https://pdfs.semanticscholar.org/470a/828d5e3962f2917a0092cc6ba46ccfe41a2a.pdf

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by backpropagating errors. *Nature*, *323(6088)*, 533-–536. Retrieved from https://psycnet.apa.org/record/1987-33645-001

Russakovsky, O. e. a. (2014). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*. Retrieved from http://link.springer.com/article/10.1007/s11263-015-0816-y#

Schölkopf, B., & Smola, A. J. (2002). *Support vector machines, regularization, optimization, and beyond*. MIT Press.

Shai, S., & Shai, B. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press. Retrieved from https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

Smits, P., Dellepiane, S., & Schowengerdt, R. (1999). Quality assessment of image classification algorithms for land-cover mapping: a review and a proposal for a cost-based approach. *International journal of remote sensing 20*, *8*, 1461—1486.

Suad, A., & Wesam, S. (2017). Review of data preprocessing techniques in data mining. *Review of Scientific Instruments*, *12(16)*, 4102–4107. Retrieved from http://docsdrive.com/pdfs/medwelljournals/jeasci/2017/4102-4107.pdf

Varun, C., & Arindam, K., B.and Vipin. (2009). *Anomaly detection: A survey*. Universidad de Minnesota. Retrieved from https://www.researchgate.net/publication/220565847_Anomaly_Detection_A_Survey

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, *1(4)*, 339—356. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/089360808890007X

Who-Lee, K., Sik-Yoon, H., Min-Song, J., & Ryoung-Park, K. (2018, 4). Convolutional neural network-based classification of driver's emotion during aggressive and smooth driving using multi-modal camera sensor. *Sensors*, *18(4)*, 957. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5948584/

Wikipedia. (n.d.). *Neuron.* https://simple.wikipedia.org/wiki/Neuron. (Último acceso en 19 de diciembre de 2019)

Williams, G., & Baxter, R. (2002, 12). A comparative study of rnn for outlier detection in data mining. *IEEE International Conference on Data Mining*, 1–16. Retrieved from https://togaware.com/papers/tr02102.pdf

Wolpher, M. (n.d.). *Anomaly detection in unstructured time series data using an lstm autoencoder.* http://www.diva-portal.org/smash/get/diva2:1225367/FULLTEXT01.pdf. (Último acceso en 11 de octubre de 2019)

Xue, Z., Shang, Y., & Feng, A. (2010, 5). Semi-supervised outlier detection based on fuzzy rough c-means clustering. *Mathematics and Computers in Simulation*, *80(9)*, 1911-–1921. Retrieved from https://www.researchgate.net/publication/220348246_Semi-supervised_outlier_detection_based_on_fuzzy_rough_C-means_clustering

Yan, S. (n.d.). *Understanding lstm and its diagrams.* https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714. (Último acceso en 11 de octubre de 2019)

Zaldivar, J., Calafate, C., Cano, J., & Manzoni, P. (2011, 10). Providing accident detection in vehicular networks through obd-ii devices and android-based smartphones. *Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks; Bonn, Alemania.*(4–7), 813-–819.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., & Hinton, G. E. (2013). On rectified linear units for speech processing. *International Conference on Acoustics, Speech and Signal Processing. IEEE*, 3517—3521. Retrieved from https://static.googleusercontent.com/media/research.google.com/es//pubs/archive/40811.pdf

Zenon, W. (2011). Artificial neural networks in civil engineering: another five years of research in poland. *Computer Assisted Mechanics and Engineering Sciences*, *18*, 131–146.

Retrieved from http://cames.ippt.gov.pl/index.php/cames/article/view/110

Zhang, A., Lipton, Z., Li, M., & Smola, A. (2019, 9). *Dive into deep learning.* https://en.d2l.ai/d2l-en.pdf. (Último acceso en 11 de octubre de 2019)

Appendix A

Experimentos de diferentes arquitecturas para los autoencoders

A.1 Redes densas

A.1.1 Redes densas para 3 componentes

		Arquitecturas Densas			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Neuronales - 3 componentes principales	**NN_33**	Input	(3,3)		0
		Flatten	9		0
		Dense	8	elu	80
		Dense	5	elu	45
		Dense	8	elu	48
		Dense	9	tanh	81
		Reshape	(3,3)		0
	NN_43	Input	(4,3)		0
		Flatten	12		0
		Dense	10	elu	130
		Dense	5	elu	55
		Dense	8	elu	48
		Dense	12	tanh	108
		Reshape	(4,3)		0
	NN_53	Input	(5,3)		0
		Flatten	15		0
		Dense	10	elu	160
		Dense	6	elu	66
		Dense	11	elu	77
		Dense	15	tanh	180
		Reshape	(5,3)		0

TABLE A.1: Arquitectura densa para 3 componentes principales (Elaboración propia).

A.1.2 Evaluación redes densas

Red	val_loss	val_acc	val_f1score	loss	acc	f1score
NN_33	0.003956	0.899894	0.287709	0.003898	0.900735	174173.640890
NN_43	0.006572	0.869167	0.233547	0.006104	0.869548	87092.835609
NN_53	0.006400	0.846857	101587.687459	0.006226	0.850568	0.283059

TABLE A.2: Tabla de evaluación de redes densas (Elaboración propia).

A.2 Redes convolucionales

A.2.1 Redes convolucionales para 3 componentes

		Arquitecturas Convolucionales			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Conv - 3 componentes principales	**CNN_33**	Input	(3,3)		0
		Conv1D	(3,2)	elu	20
		MaxPooling1D	(2,2)		0
		Conv1D	(2,4)	elu	28
		MaxPooling1D	(1,4)		0
		Conv1D	(1,6)	elu	54
		UpSampling1D	(3,6)		0
		Conv1D	(3,3)	tanh	57
	CNN_43	Input	(4,3)		0
		Conv1D	(4,2)	elu	20
		MaxPooling1D	(2,2)		0
		Conv1D	(2,4)	elu	26
		MaxPooling1D	(1,4)		0
		Conv1D	(1,6)	elu	54
		UpSampling1D	(4,6)		0
		Conv1D	(4,3)	tanh	57
	CNN_53	Input	(5,3)		0
		Conv1D	(5,2)	elu	20
		MaxPooling1D	(3,2)		0
		Conv1D	(3,4)	elu	28
		MaxPooling1D	(2,4)		0
		Conv1D	(2,5)	elu	45
		Reshape	(5,2)		0
		Conv1D	(5,3)	tanh	15

TABLE A.3: Arquitectura convolucional para 3 componentes principales (Elaboración propia).

A.2.2 Evaluación redes convolucionales

Red	val_loss	val_acc	loss	acc
CNN_33	0.0070	0.8453	0.0067	0.8434
CNN_43	0.0100	0.8136	0.0097	0.8152
CNN_53	0.0102	0.7951	0.0108	0.7907

TABLE A.4: Tabla de evaluación de redes convolucionales (Elaboración propia).

A.3 Redes recurrentes

A.3.1 Redes recurrentes para 3 componentes

		Arquitecturas Recurrentes			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Rec - 3 componentes principales	**RNN_33**	Input	(3,3)		0
		LSTM	(3,9)	elu	468
		LSTM	6	elu	384
		Reshape	(3,2)		0
		LSTM	(3,3)	elu	72
		LSTM	(3,9)	elu	468
		TimeDistributed(Dense(3))	(3,3)	tanh	30
	RNN_43	Input	(4,3)		468
		LSTM	(4,9)	elu	384
		LSTM	6	elu	0
		Reshape	(3,2)		216
		LSTM	(3,6)	elu	576
		LSTM	(3,9)	elu	40
		TimeDistributed(Dense(3))	(3,4)	tanh	0
		Reshape	(4,3)		0
	RNN_53	Input	(5,3)		0
		LSTM	(5,9)	elu	468
		LSTM	6	elu	384
		Reshape	(3,2)		0
		LSTM	(3,3)	elu	72
		LSTM	(3,9)	elu	468
		TimeDistributed(Dense(3))	(3,5)		50
		Reshape	(5,3)	tanh	0

TABLE A.5: Arquitectura recurrente para 3 componentes principales (Elaboración propia).

A.3.2 Evaluación redes recurrentes

Red	val_loss	val_acc	loss	acc
RNN_33	0.0039	0.8855	0.0037	0.8900
RNN_43	0.0108	0.7871	0.0102	0.7884
RNN_53	0.0295	0.2986	0.0290	0.3370

TABLE A.6: Tabla de evaluación de redes recurrentes (Elaboración propia).

Appendix B

Arquitectura del Sistema de Demostración

Si bien el trabajo presentado no promete un sistema que implemente el mecanisno propuesto, es necesario tener un prototipo para demostrar el correcto funcionamiento del detector de anomalías, por lo que este anexo se centra en presentar la arquitectura tanto física como lógica del prototipo.

B.1 Arquitectura Física

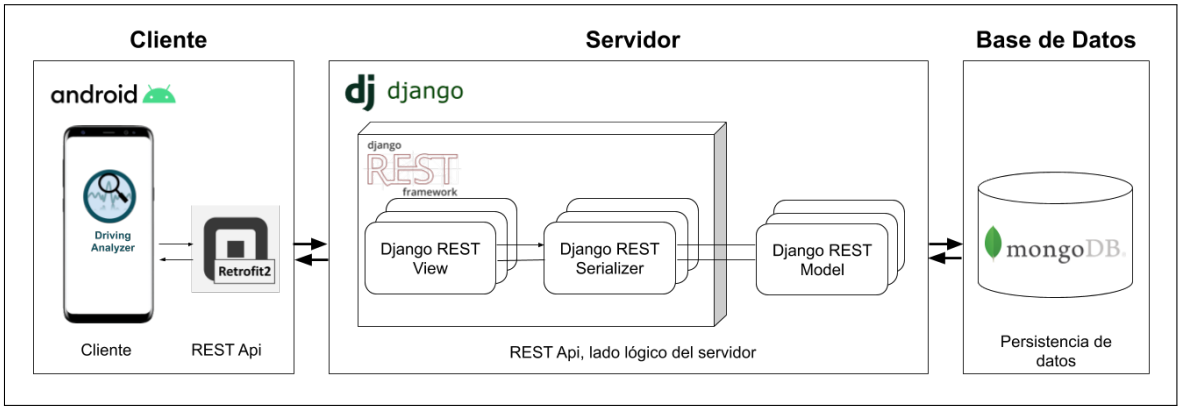


FIGURE B.1: Arquitectura física del Sistema de Demostración (Elaboración propia).

La arquitectura física del prototipo cuenta con tres partes principales: el cliente, el servidor y la base de datos, como se puede observar en la Figura B.1. En primer lugar el cliente esta compuesto por una aplicación móvil para Android encargada de capturar los datos de los sensores inerciales, para posteriormente enviarlas al servidor usando Retrofit 2, posteriormente los datos ingresan al servidor realizado en Django, una vez el servidor recibe estos datos, se encarga de preprocesarlos para posteriormente predecir sí los datos enviados son anomalías o no lo son, y por último los datos recibidos son almacenados en la Base de Datos de MongoDB con su respectiva predicción con el objetivo de hacer un monitoreo de la conducción del usuario.

B.2 Arquitectura Lógica

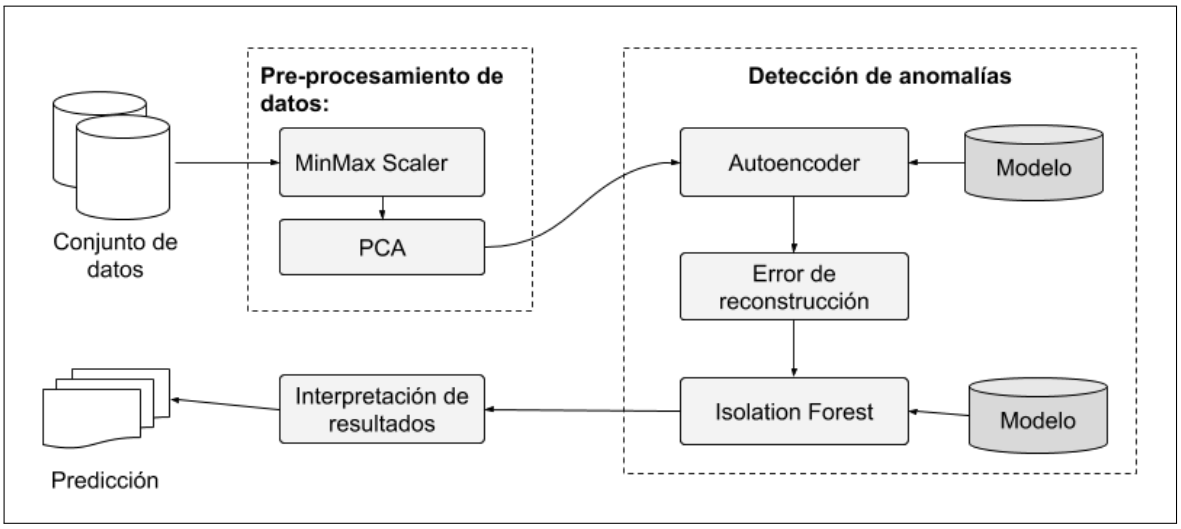


FIGURE B.2: Arquitectura Lógica del Sistema de Demostración (Elaboración propia).

En cuanto a la arquitectura lógica se observa las dos principales partes del mecanismo de detección propuesto: el pre-procesamiento y la detección de anomalías, como se puede observar en la Figura B.2.