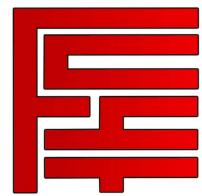




Mayor de San Simón University  
Faculty of Science and Technology  
Systems Engineering Degree



# 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



## *Agradecimientos*

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.*



# Índice general

<b>Agradecimientos</b>	<b>III</b>
<b>Abstract</b>	<b>xv</b>
<b>1. INTRODUCTION</b>	<b>1</b>
1.1. Approach of the problem . . . . .	1
1.2. General objective . . . . .	2
1.3. Specific objectives . . . . .	2
1.4. Justification . . . . .	3
1.4.1. Practical justification . . . . .	3
1.4.2. Methodological justification . . . . .	4
1.5. Limits and scope . . . . .	4
1.6. Research method . . . . .	4
<b>2. FUNDAMENTALS OF THE DETECTION OF ANOMALIES</b>	<b>5</b>
2.1. Anomaly detection . . . . .	5
2.2. Challenges in anomaly detection . . . . .	7
2.2.1. Anomaly detection approaches . . . . .	7
Supervised anomaly detection . . . . .	8
Semi-supervised anomaly detection . . . . .	8
Unsupervised anomaly detection . . . . .	8
2.3. Related work . . . . .	8
2.4. Focus on the problem . . . . .	9
<b>3. MACHINE LEARNING FOR ANOMALY DETECTION</b>	<b>11</b>
3.1. Supervised Learning, Unsupervised Learning and Semi Supervised Learning . . . . .	11
3.1.1. Supervised Learning . . . . .	12
3.1.2. Unsupervised Learning . . . . .	12
3.1.3. Semi Supervised Learning . . . . .	12
3.2. Generative and Discriminative Models . . . . .	13
3.3. Artificial neural networks . . . . .	14
3.3.1. Neurons or nodes . . . . .	14
3.3.2. Types of activation functions . . . . .	16
Sigmoid activation function (logistics function) . . . . .	16
Hyperbolic Tangent Function - Tanh . . . . .	16
Rectified Linear Unit function (ReLU) . . . . .	18
Leaky ReLU (LReLU) . . . . .	19

Exponential Linear Unit Function (ELU) . . . . .	19
3.3.3. Architecture of the Neural Networks . . . . .	19
3.3.4. Learning process of Neural Networks . . . . .	20
Backpropagation . . . . .	21
3.4. Types of Neural Networks . . . . .	21
3.4.1. Autoencoders . . . . .	21
3.4.2. Convolutional neural networks . . . . .	22
3.4.3. Recurrent neural networks . . . . .	22
3.4.4. LSTM . . . . .	24
3.4.5. GRU . . . . .	25
3.5. Técnicas de detección de anomalías . . . . .	26
3.5.1. One-Class SVM . . . . .	26
3.5.2. Isolation Forest . . . . .	27
3.5.3. Autoencoders . . . . .	28
3.6. Métricas de evaluación . . . . .	30
3.6.1. Precisión de clasificación (accuracy) . . . . .	30
3.6.2. Pérdida logarítmica (Logarithmic Loss) . . . . .	30
3.6.3. Matriz de confusión . . . . .	31
3.6.4. Área bajo la curva (AUC) . . . . .	32
Tasa de Verdaderos Positivos (TPR) . . . . .	32
Tasa de Verdaderos Negativos (TNR) . . . . .	32
ROC (Receiver Operating Characteristics) . . . . .	33
3.6.5. F1 Score . . . . .	33
Precisión (Precision) . . . . .	34
Recuperación (Recall) . . . . .	34
<b>4. CAPTURA Y PREPARACIÓN DE DATOS</b>	<b>35</b>
4.1. Captura de datos . . . . .	35
4.2. Preparación de datos . . . . .	37
4.2.1. Selección de datos . . . . .	37
4.3. Pre-procesamiento de datos . . . . .	38
4.3.1. Limpieza de datos . . . . .	39
Técnicas de compensación de registros incompletos . . . . .	39
Eliminar ruido (suavizado) de los datos . . . . .	39
Fusión o integración de datos . . . . .	42
Transformación de datos . . . . .	43
Reducción de datos . . . . .	44
<b>5. GENERACIÓN DEL MECANISMO DE DETECCIÓN DE ANOMALÍAS</b>	<b>53</b>
5.1. Entorno de desarrollo . . . . .	53
5.2. Conjunto de datos normales y anómalos . . . . .	54
5.2.1. Generación de series temporales . . . . .	54
5.3. Modelo de detección de anomalías . . . . .	55
5.3.1. Modelo del comportamiento normal . . . . .	56
Arquitectura del modelo . . . . .	56

5.3.2. Método de detección de anomalías . . . . .	59
Umbralización . . . . .	59
Isolation Forest . . . . .	63
One-Class SVM . . . . .	66
Evaluación del método de detección de anomalías . . . . .	67
<b>6. RESULTADOS Y EVALUACIÓN</b>	<b>71</b>
6.1. Evaluación de desempeño . . . . .	71
6.1.1. Evaluación en términos de rendimiento de detección . . . . .	71
6.2. Resultados . . . . .	72
6.2.1. Detección de anomalías del tipo zig zag . . . . .	73
6.2.2. Detección de anomalías del tipo giros a alta velocidad . . . . .	74
6.2.3. Detección de anomalías del tipo frenos en seco . . . . .	75
6.2.4. Detección de falsos positivos . . . . .	77
<b>7. CONCLUSIONES Y TRABAJOS FUTUROS</b>	<b>79</b>
7.1. Conclusiones . . . . .	79
7.2. Trabajos futuros . . . . .	80
<b>BIBLIOGRAPHY</b>	<b>83</b>
Referencias . . . . .	83
<b>A. Experimentos de diferentes arquitecturas para los autoencoders</b>	<b>89</b>
A.1. Redes densas . . . . .	90
A.1.1. Redes densas para 3 componentes . . . . .	90
A.1.2. Evaluación redes densas . . . . .	91
A.2. Redes convolucionales . . . . .	91
A.2.1. Redes convolucionales para 3 componentes . . . . .	91
A.2.2. Evaluación redes convolucionales . . . . .	92
A.3. Redes recurrentes . . . . .	92
A.3.1. Redes recurrentes para 3 componentes . . . . .	92
A.3.2. Evaluación redes recurrentes . . . . .	93
<b>B. Arquitectura del Sistema de Demostración</b>	<b>95</b>
B.1. Arquitectura Física . . . . .	95
B.2. Arquitectura Lógica . . . . .	96



# Índice de figuras

1.1. Deaths due to traffic accidents by region depending on user's type (de Salud (OMS), s.f.). . . . .	2
1.2. Problem tree (Own elaboration). . . . .	3
2.1. Example of point anomalies in a 2-dimensional data set (Varun y Arindam, 2009). . . . .	6
2.2. Contextual anomaly $t_2$ in a temperature time serie (Varun y Arindam, 2009). . . . .	6
2.3. Collective anomaly corresponding to premature atrial contraction in a human electrocardiogram (Varun y Arindam, 2009). . . . .	7
2.4. Proposed anomaly detection method (Own elaboration). . . . .	10
3.1. Graphic of a biological neuron. Reproduced from (Wikipedia, s.f.). . . . .	15
3.2. Graphic of an artificial neuron. Reproduced from (Jayesh, s.f.). . . . .	15
3.3. Activation functions (Jing y Guanci, 2018). . . . .	17
a. Function Sigmoid . . . . .	17
b. Function Tanh . . . . .	17
c. Function ReLU . . . . .	17
d. Function Leaky ReLu . . . . .	17
e. Function ELU . . . . .	17
3.4. Architecture of an artificial neuron. Reproduced from (Michael, 2015). . . . .	20
3.5. Graphic of an Autoencoder (Own elaboration). . . . .	21
3.6. Architecture of a Convolutional Neural Network (CNN) (Muhammad, s.f.). . . . .	23
3.7. Sequential processing in a recurrent neural network (RNN) (Olah, s.f.). . . . .	24
3.8. Estructura de LSTM. Reproducido desde Yan (Yan, s.f.). . . . .	25
3.9. Estructura de GRU. Reproducido desde (Zhang, Lipton, Li, y Smola, 2019). . . . .	26
3.10. One-Class SVM. Reproducido desde (Alashwal, Bin D., y Othman, 2006) . . . . .	27
3.11. Comparacion del rendimiento entre los algoritmos One-Clas SVM e Isolation Forest. Reproducido desde (0.22, s.f.). . . . .	28
3.12. Detección de anomalías con autoencoder (Elaboración propia). . . . .	29
3.13. Ejemplo de un curva AUC-ROC (Özler, s.f.). . . . .	33
4.1. Recolección de datos, con intervalo de un segundo (Elaboración propia). . . . .	36
4.2. Soporte para celular de parabrisas, posición horizontal (Elaboración propia). . . . .	36
4.3. Fragmento del conjunto de datos obtenido (Elaboración propia). . . . .	37
4.4. Gráfica de los sensores capturados en diferentes posiciones (Elaboración propia). . . . .	38
4.5. Histograma de frecuencias del conjunto de datos (Elaboración propia). . . . .	40
4.6. Tabla de resultados estadísticos del conjunto de datos (Elaboración propia). . . . .	41
4.7. Regla 68-95-99.7 (Galarnyk, s.f.). . . . .	42

4.8. Regla 68-95-99.7 aplicada a los valores de los sensores del acelerómetro en Z (Elaboración propia) . . . . .	42
4.9. División del conjunto de entrenamiento (Elaboración propia) . . . . .	46
4.10. Visualización de los parámetros de conducción capturados (Elaboración propia) . . . . .	47
4.11. Gráfica resultante de aplicar diferentes tipos de normalizaciones a un conjunto de datos (Elaboración propia) . . . . .	48
a. Min max Scaler . . . . .	48
b. Standard Scaler . . . . .	48
c. Max Scaler . . . . .	48
d. Robust Scaler . . . . .	48
4.12. Gráfico de la varianza vs. el número de componentes (Elaboración propia) . . . . .	52
5.1. Gráfica resultante de diferentes tamaños de series de tiempo (Elaboración propia) . . . . .	55
a. Serie de tiempo de 2 pasos. . . . .	55
b. Serie de tiempo de 3 pasos. . . . .	55
c. Serie de tiempo de 4 pasos. . . . .	55
d. Serie de tiempo de 5 pasos. . . . .	55
5.2. Resultados (Elaboración propia) . . . . .	60
a. Resultados de la red <b>NN_33</b> . . . . .	60
b. Resultados de la red <b>CNN_33</b> . . . . .	60
c. Resultados de la red <b>RNN_33</b> . . . . .	60
5.3. Curva de los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia) . . . . .	61
5.4. Resultados de la obtención de codos con diferentes valores de Sensibilidad, para los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia) . . . . .	62
5.5. La figura de la izquierda muestra el aislamiento de una anomalía, que requiere solo tres particiones. A la derecha, el aislamiento de un punto normal requiere seis particiones (Wolpher, s.f.) . . . . .	64
5.6. Representación gráfica del modelo de comportamiento normal o autoencoder (Elaboración propia) . . . . .	64
5.7. Representación gráfica del error de reconstrucción usado para el entrenamiento de los bosques de aislamiento y los SVM de una clase (Elaboración propia) . . . . .	65
5.8. Mecanismo de detección de anomalías (Elaboración propia) . . . . .	68
6.1. Resultados de la detección de anomalías del tipo zig zag (Elaboración propia) . . . . .	73
6.2. Resultados de la detección de anomalías del tipo giros a alta velocidad (Elaboración propia) . . . . .	75
6.3. Resultados de la detección de anomalías del tipo frenos en seco (Elaboración propia) . . . . .	76
6.4. Resultados de la detección de falsos positivos (Elaboración propia) . . . . .	77
B.1. Arquitectura física del Sistema de Demostración (Elaboración propia) . . . . .	95
B.2. Arquitectura Lógica del Sistema de Demostración (Elaboración propia) . . . . .	96

# Índice de cuadros

3.1. Matriz de confusión, para una clasificación binaria (Elaboración propia) . . . . .	31
4.1. Tabla de división del conjunto de datos (Elaboración propia). . . . .	46
4.2. Tabla con estadísticas descriptivas de los datos escalados con diferentes técnicas (Elaboración propia). . . . .	50
5.1. Tabla del conjunto de anomalías (Elaboración propia). . . . .	54
5.2. Tabla de los métodos comparados (Elaboración propia). . . . .	56
5.3. Arquitectura densa para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia). . . . .	57
5.4. Arquitectura convolucional para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia). . . . .	57
5.5. Arquitectura recurrente para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia). . . . .	58
5.6. Evaluación de las redes NN_33, CNN_33 y RNN_33 (Elaboración propia). . . . .	59
5.7. Evaluación de la detección de anomalías para cada codo obtenido con los diferentes valores de sensibilidad (Elaboración propia). . . . .	63
5.8. Evaluación de la detección de anomalías usando Isolation forest para valores compresos (Elaboración propia). . . . .	65
5.9. Evaluación de la detección de anomalías usando Isolation forest para errores de reconstrucción (Elaboración propia). . . . .	66
5.10. Evaluación de la detección de anomalías usando One-Class SVM para valores compresos (Elaboración propia). . . . .	66
5.11. Evaluación de la detección de anomalías usando One-Class SVM para el error de reconstrucción del autoencoder (Elaboración propia). . . . .	66
5.12. Comparación de los mejores métodos de detección de anomalías (Elaboración propia). . . . .	67
6.1. Matriz de confusión, para el mecanismo de detección de anomalías (Elaboración propia). . . . .	72
6.2. Resultados anomalías tipo Zig Zag (Elaboración propia). . . . .	74
6.3. Resultados del tipo Giros a alta Velocidad (Elaboración propia). . . . .	75
6.4. Resultados del tipo Frenos en seco (Elaboración propia). . . . .	76
A.1. Arquitectura densa para 3 componentes principales (Elaboración propia). . . . .	90
A.2. Tabla de evaluación de redes densas (Elaboración propia). . . . .	91
A.3. Arquitectura convolucional para 3 componentes principales (Elaboración propia).	91

A.4. Tabla de evaluación de redes convolucionales (Elaboración propia). . . . .	92
A.5. Arquitectura recurrente para 3 componentes principales (Elaboración propia). . . . .	92
A.6. Tabla de evaluación de redes recurrentes (Elaboración propia). . . . .	93

## *Resumen*

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.



## Capítulo 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.

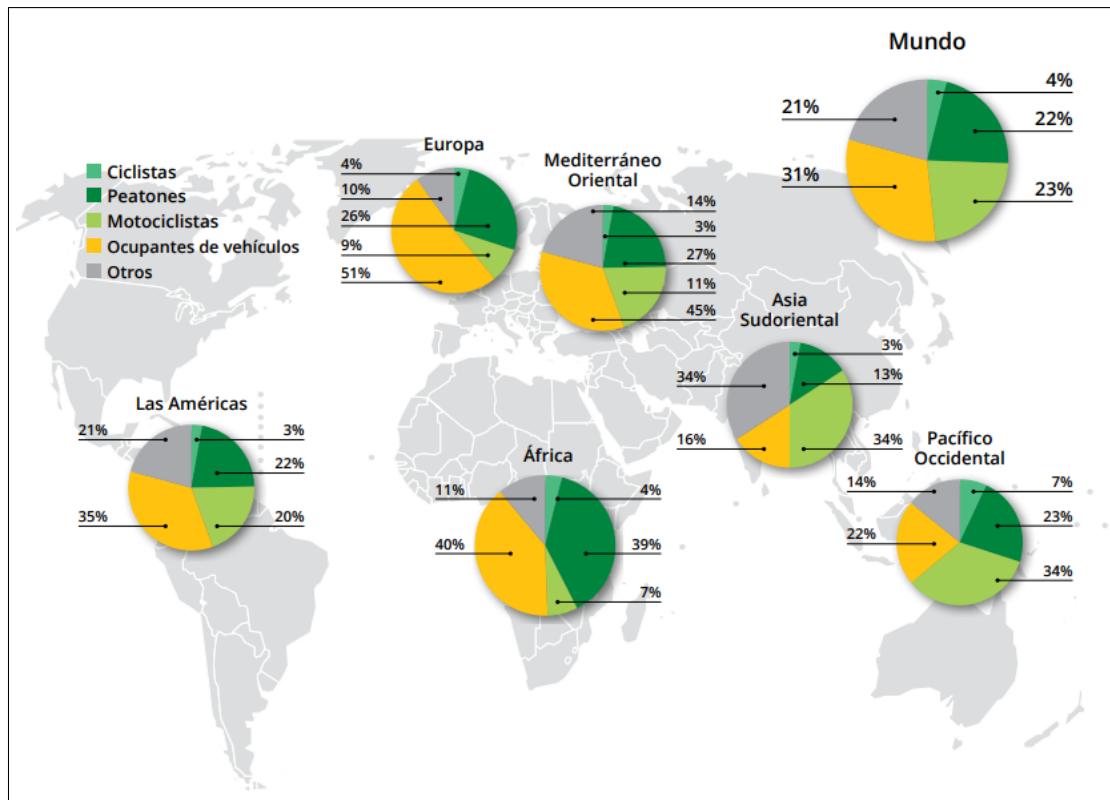


FIGURA 1.1: Deaths due to traffic accidents by region depending on user's type  
(de Salud (OMS), s.f.).

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 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.

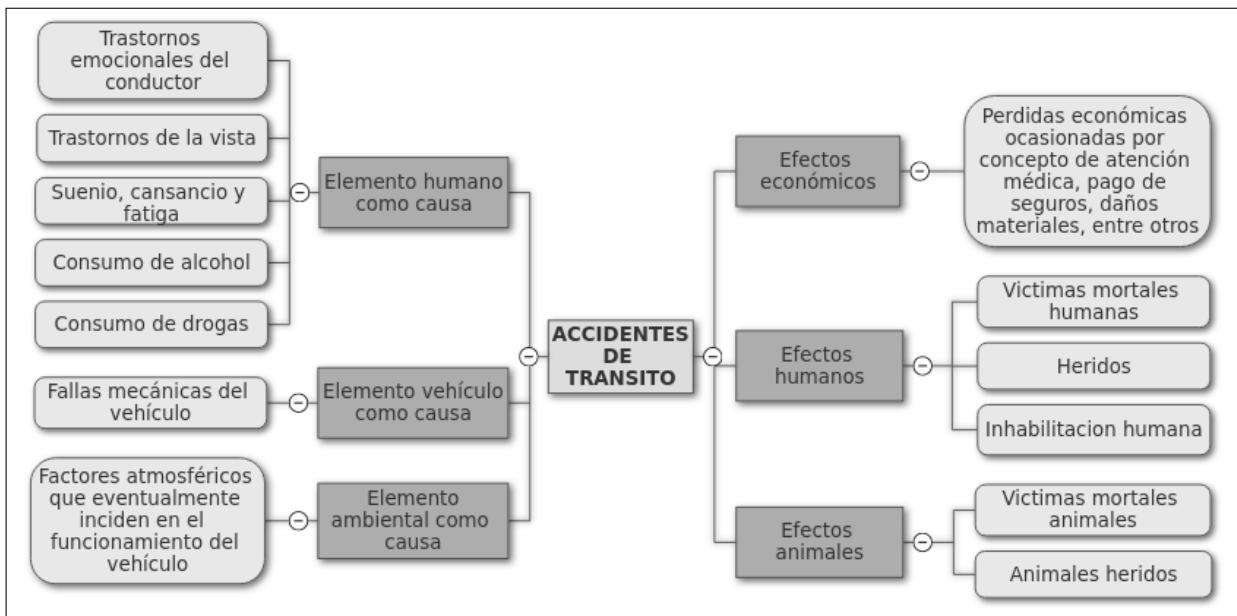


FIGURA 1.2: Problem tree (Own elaboration).

- 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.
- 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 to 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?*

## Capítulo 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.

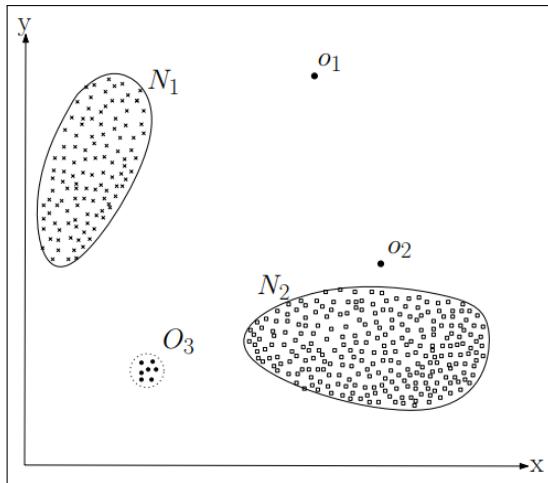


FIGURA 2.1: Example of point anomalies in a 2-dimensional data set (Varun y Arindam, 2009).

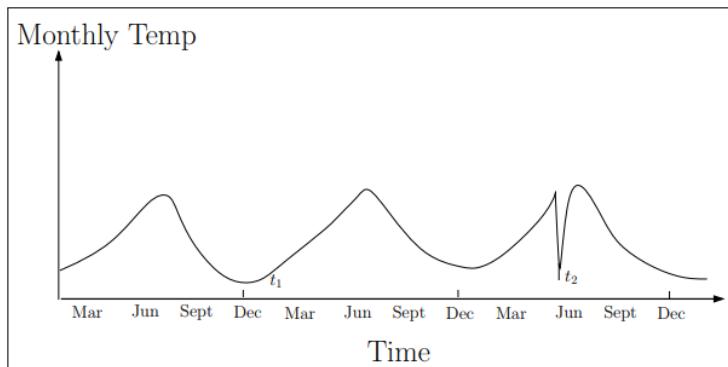


FIGURA 2.2: Contextual anomaly  $t_2$  in a temperature time serie (Varun y Arindam, 2009).

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.

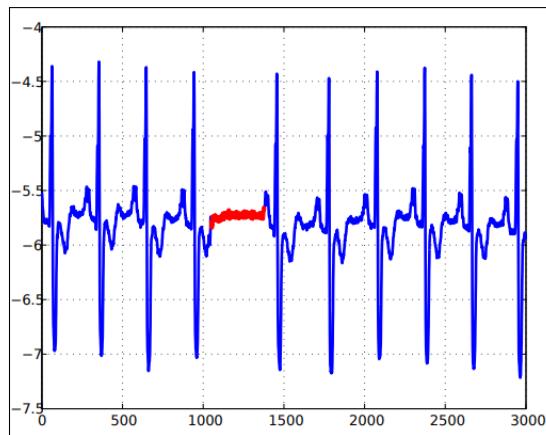


FIGURA 2.3: Collective anomaly corresponding to premature atrial contraction in a human electrocardiogram (Varun y Arindam, 2009).

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 y cols. (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 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 y cols. (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 y 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, y Javale, 2013; Chen, Yu, Zhu, Chen, y Li, 2015; Eren, Makinist, Akin, y Yilmaz, 2012; Boonmee y Tangamchit, 2009; Koh y 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, y 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.

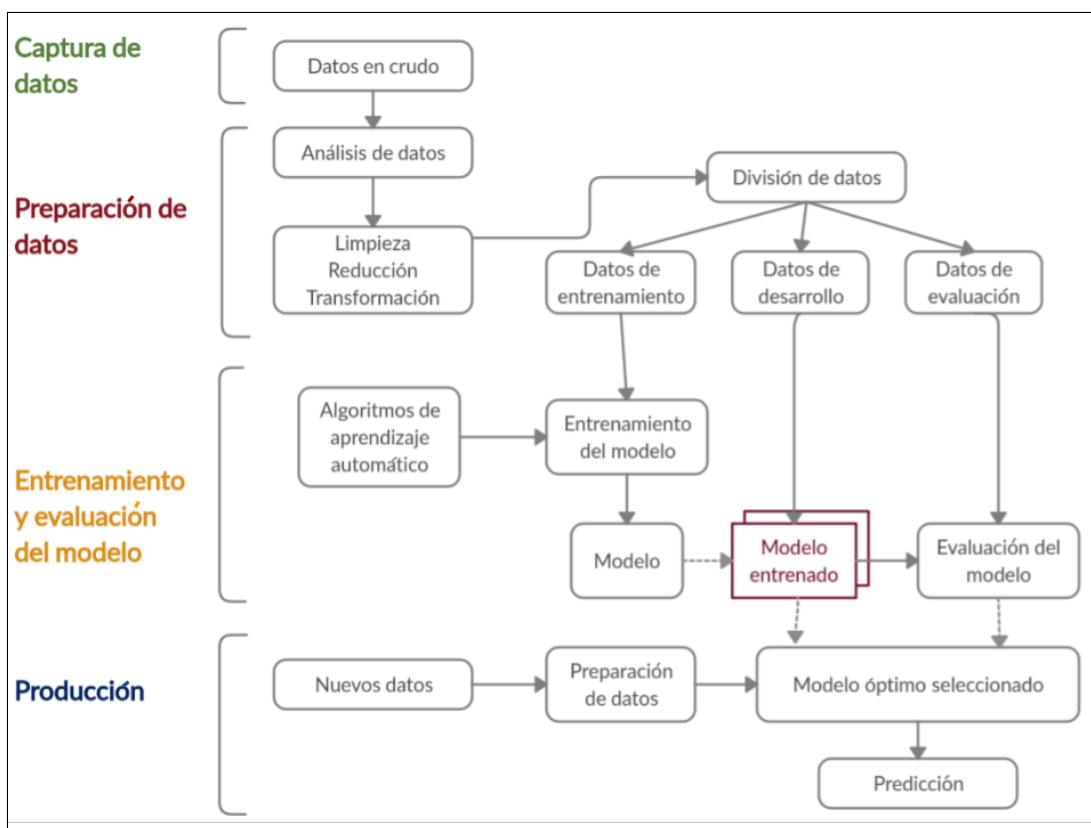


FIGURA 2.4: Proposed anomaly detection method (Own elaboration).

## Capítulo 3

# MACHINE LEARNING FOR ANOMALY DETECTION

The term Machine Learning refers to the automatic detection of significant patterns within a data set (Shai y 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.

$$Y = f(X) \quad (3.1)$$

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, y 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 y 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 Boltzmann Machine, Generative Adversarial 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<sup>1</sup> 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).

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 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  $x(x_1, x_2, \dots, x_n)$  of inputs and produces a unique response or output. The main elements of an artificial neuron are the following:

- **Inputs** that receive data from other neurons, these inputs would be dendrites of a biological neuron.

---

<sup>1</sup>ANN, Artificial Neural Network

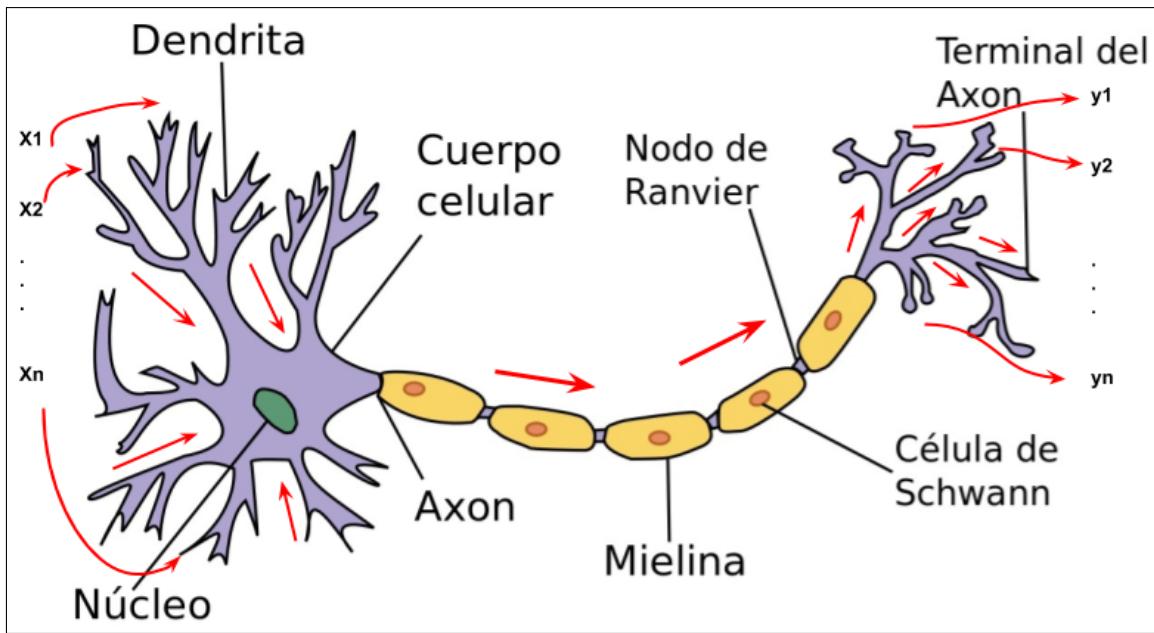


FIGURA 3.1: Graphic of a biological neuron. Reproduced from (Wikipedia, s.f.).

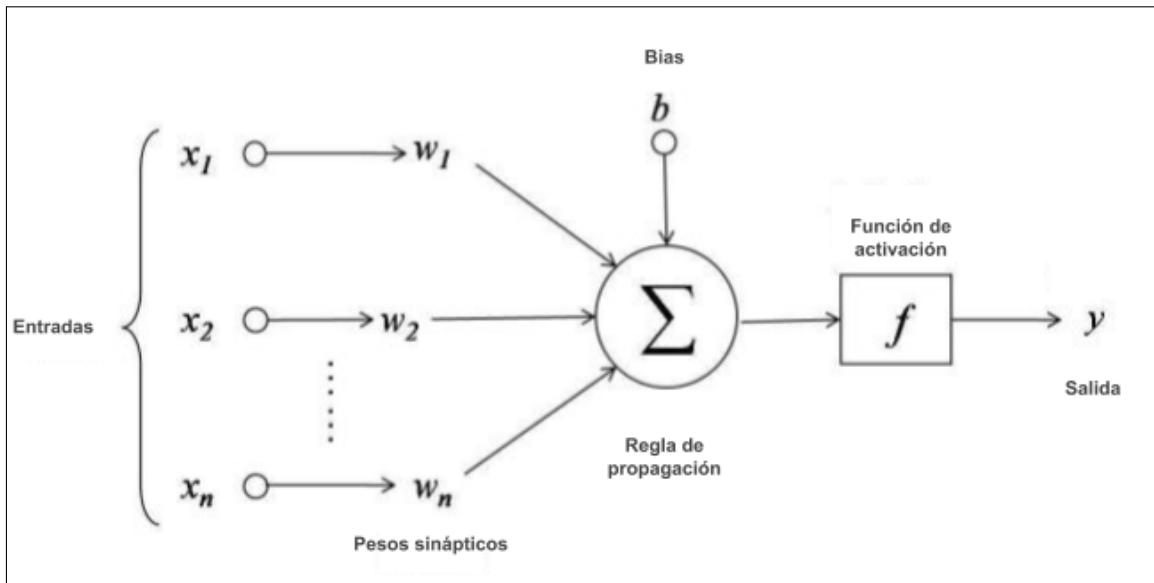


FIGURA 3.2: Graphic of an artificial neuron. Reproduced from (Jayesh, s.f.).

- **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.

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

- **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:

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

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:

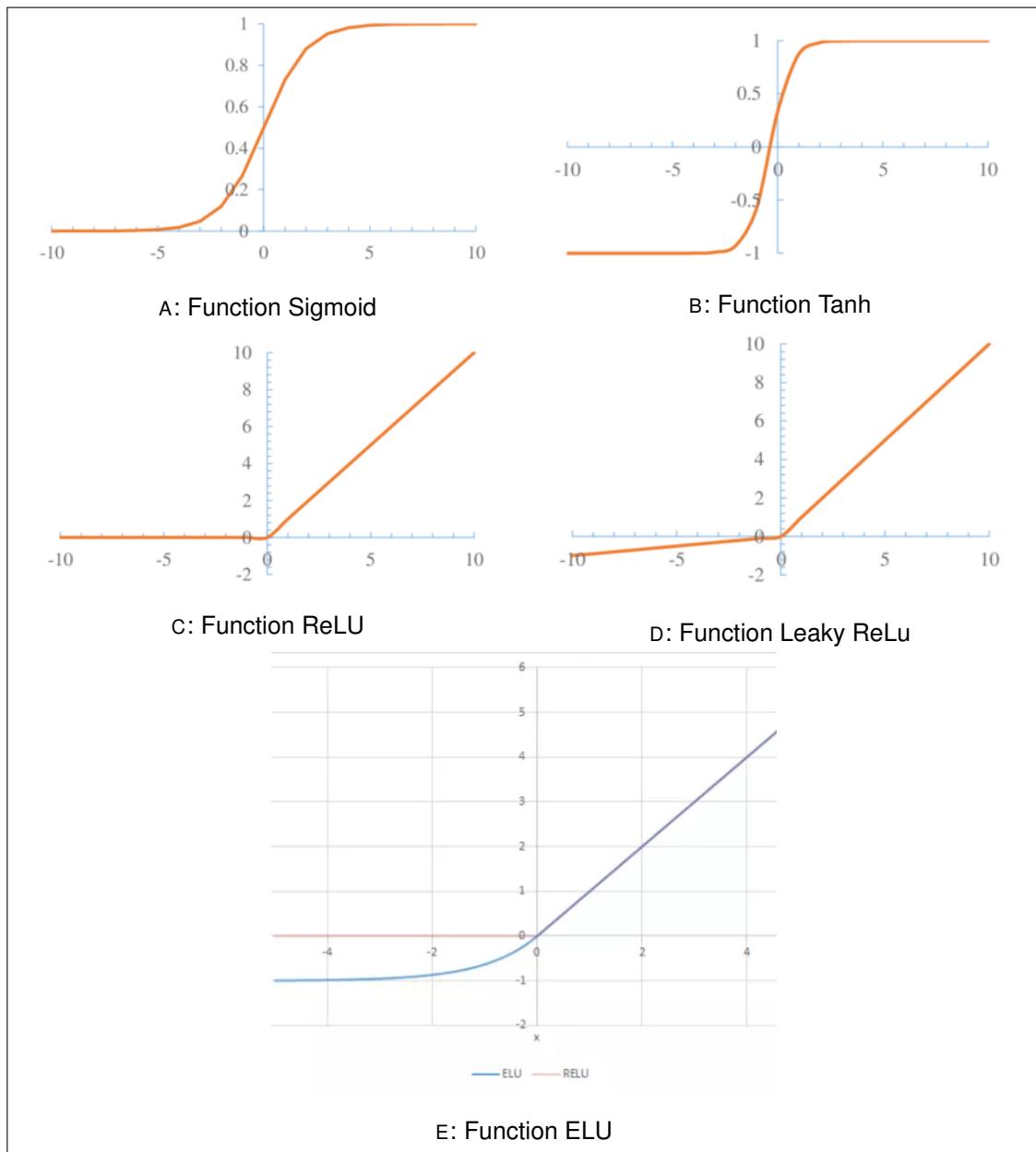


FIGURA 3.3: Activation functions (Jing y Guanci, 2018).

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

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 tangent 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, y Grangiera, 2017) and speech recognition tasks (Mass, Han-nun, y 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. ReLU is a faster learning activation function (Lecun, Bengio, y 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:

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

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 y cols., 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 overfitted compared to sigmoid function, although abandonment technique has been adopted to reduce overfitted 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 y cols., 2013). The parameter  $\alpha$  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  $\alpha$  in the range of 0.01, so LReLU (See Figure 3.3d) is calculated as:

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

### Exponential Linear Unit Function (ELU)

The ELU<sup>2</sup> 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.

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

### 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 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 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.

---

<sup>2</sup>ELU, Exponential Lineal Unit

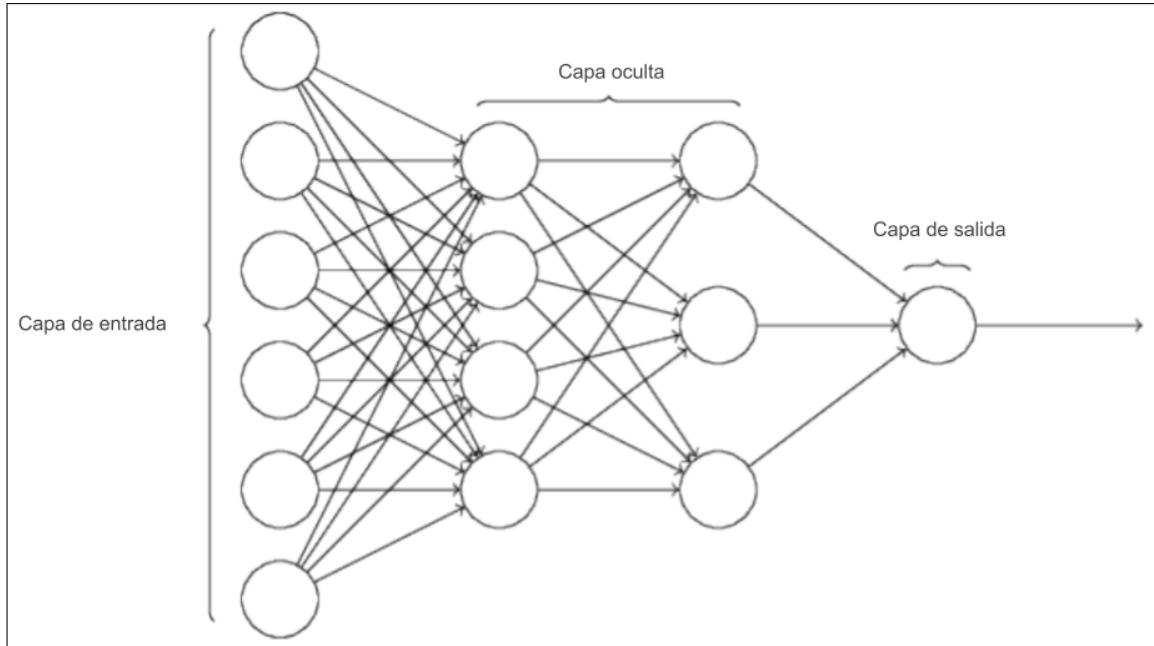


FIGURA 3.4: Architecture of an artificial neuron. Reproduced from (Michael, 2015).

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.

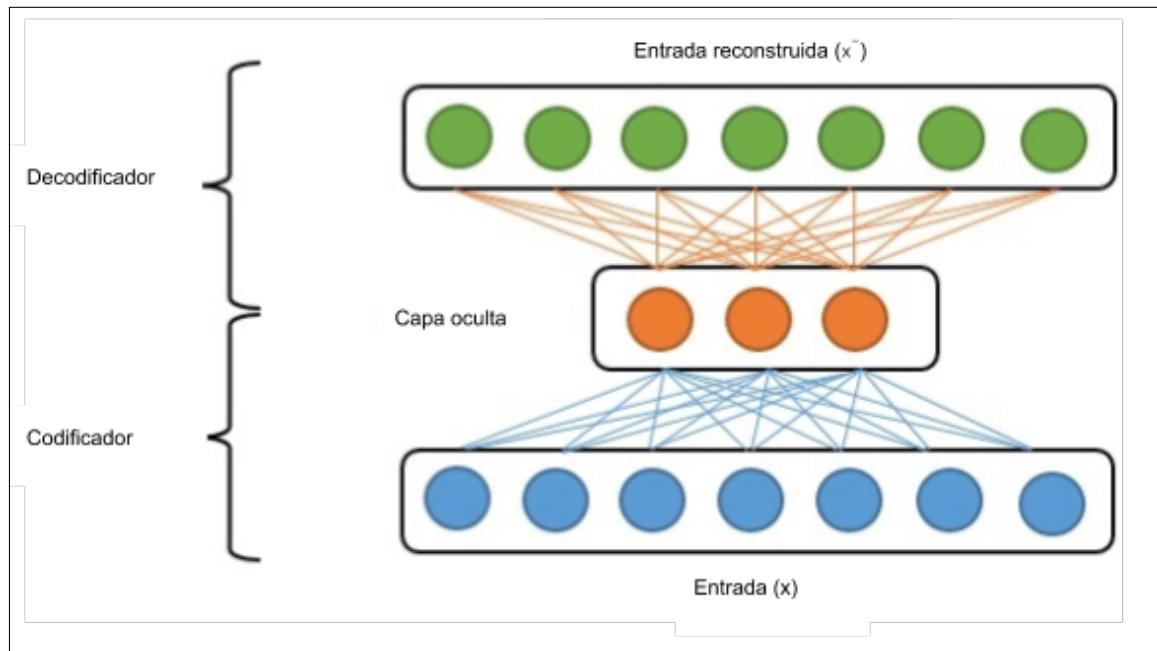


FIGURA 3.5: Graphic of an Autoencoder (Own elaboration).

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  $h = \text{encoder}(x)$ , which is defined by a linear or nonlinear function. If function of encoder is non-linear, autoencoder 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,  $\hat{x} = \text{decoder}(h)$ .

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 y cols. (1998) propose convolutional neural networks (CNN<sup>3</sup>) 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.

### 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.

---

<sup>3</sup>CNN, Convolutional Neural Network

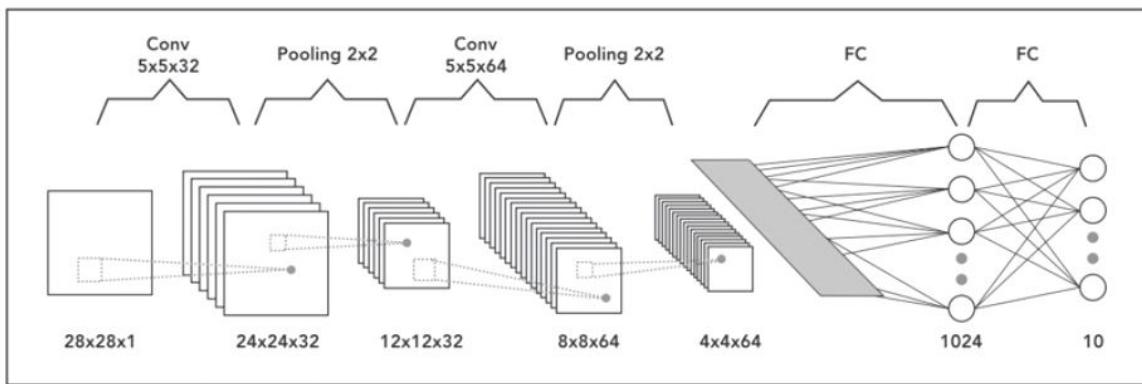


FIGURA 3.6: Architecture of a Convolutional Neural Network (CNN) (Muhammad, s.f.).

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<sup>4</sup>) appear, which are a type of artificial neural network proposed in the 80s (Rumelhart, Hinton, y Williams, 1986; Elman, 1990; Werbos, 1988) designed to 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 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<sup>5</sup>). 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,

<sup>4</sup>RNN, Recurrent Neural Network

<sup>5</sup>BPTT, Backpropagation Through The Time

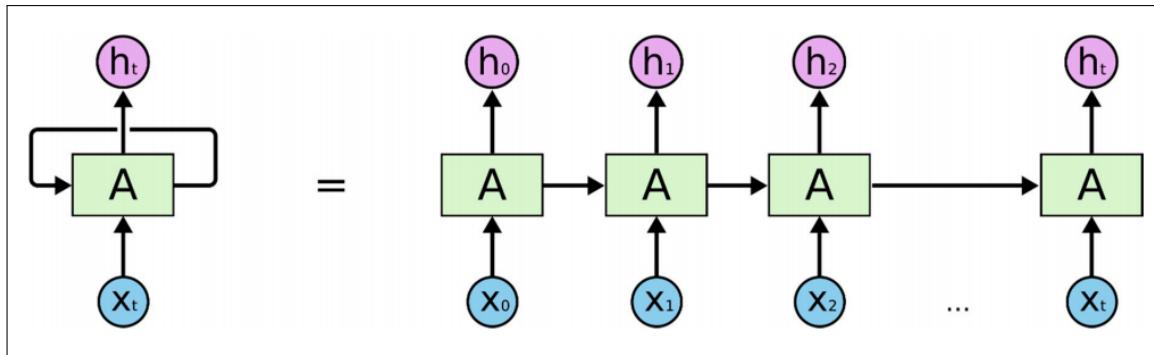


FIGURA 3.7: Sequential processing in a recurrent neural network (RNN) (Olah, s.f.).

Simard, y Frasconi, 1994). To overcome disappearance and explosion gradient problems that standard RNNs have, LSTM and GRU can be used (Pascanu, Mikolov, y Bengio, 2013).

#### 3.4.4. LSTM

LSTM<sup>6</sup> 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 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.

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.

<sup>6</sup>LSTM, Long Short-Term Memory

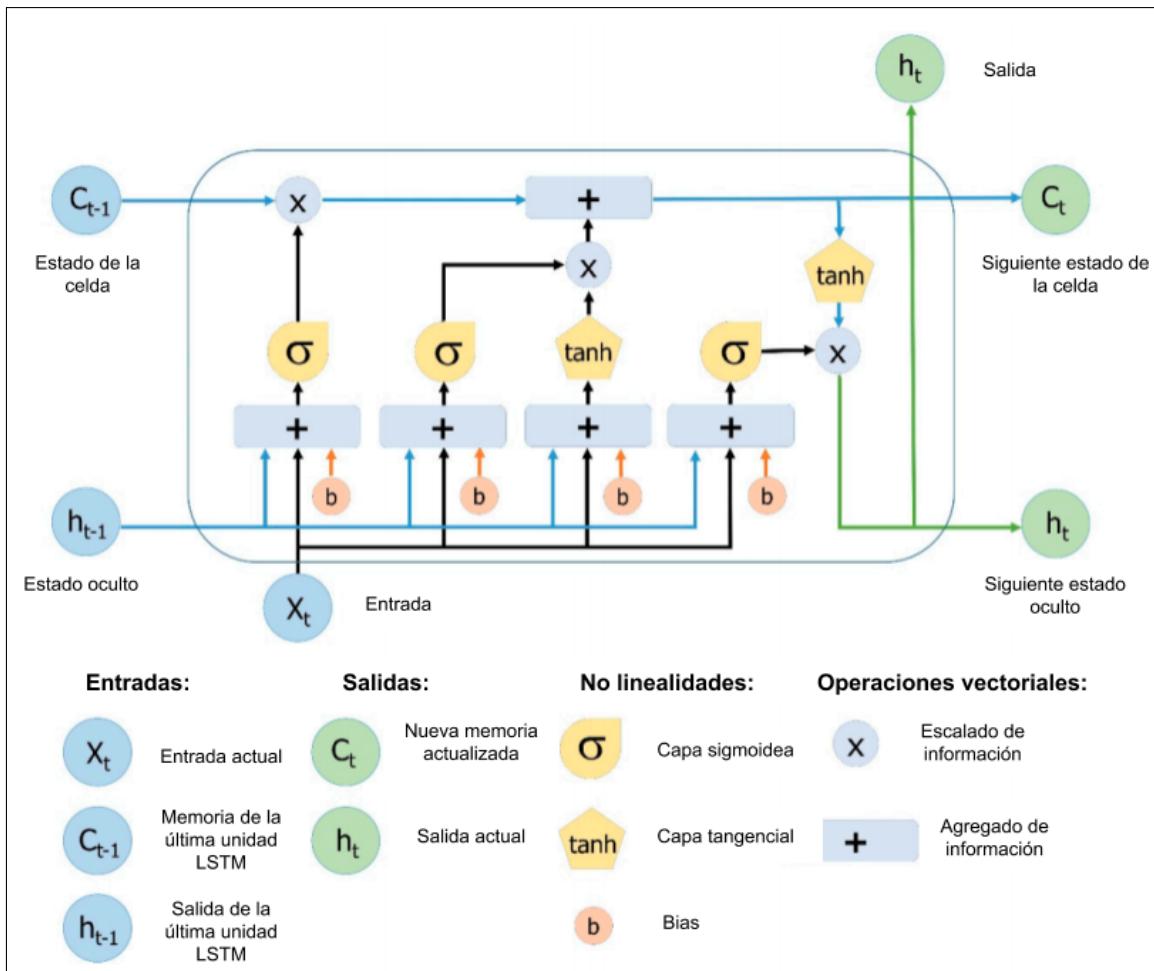


FIGURA 3.8: LSTM structure. Played from Yan (Yan, s.f.).

### 3.4.5. GRU

GRU<sup>7</sup> 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.

<sup>7</sup>GRU, Gated Recurrent Unit

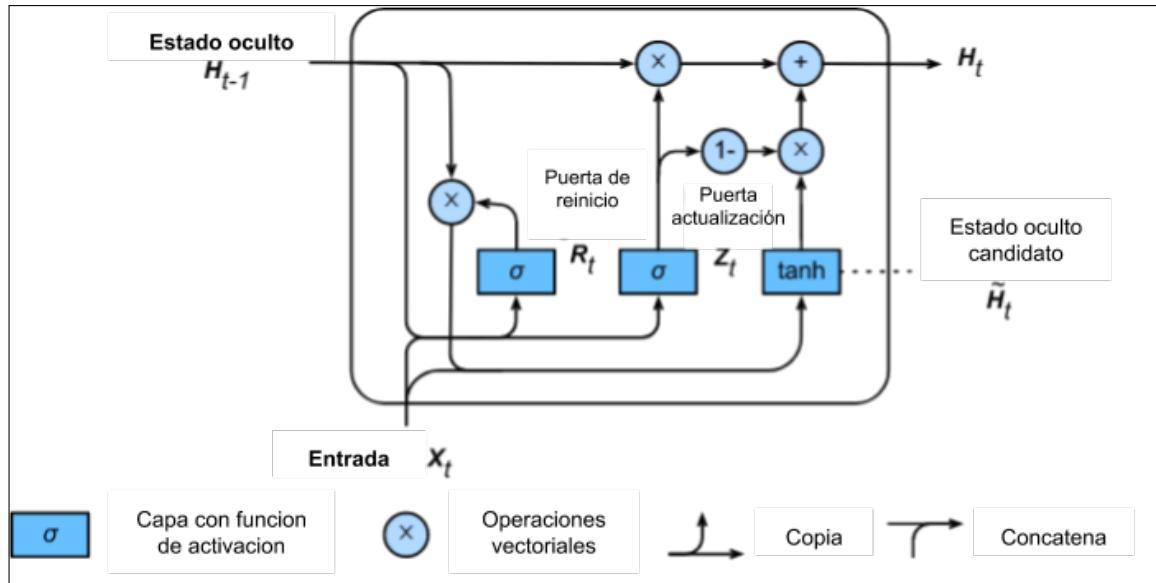


FIGURA 3.9: GRU structure. Reproduced from (Zhang y cols., 2019).

### 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.

#### 3.5.1. One-Class SVM

One-Class SVM<sup>8</sup> (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.

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.

<sup>8</sup>SVM, Support Vector Machine

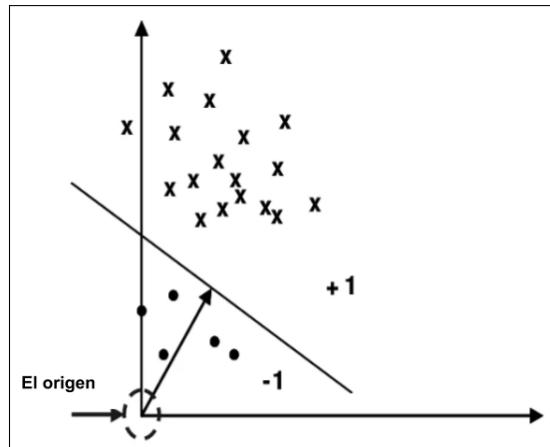


FIGURA 3.10: One-Class SVM. Reproduced from (Alashwal y cols., 2006)

### 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 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.

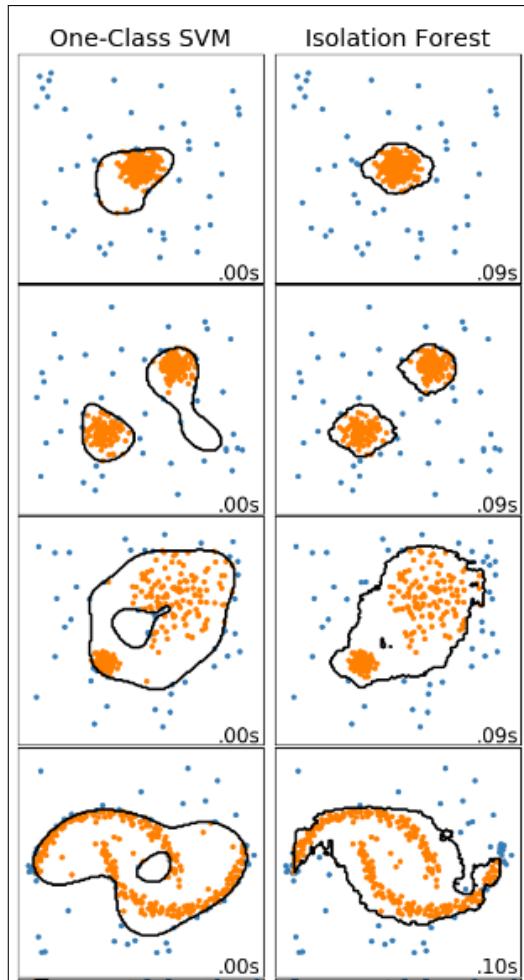


FIGURA 3.11: Performance comparison between the One-Class SVM and Isolation Forest algorithms. Reproduced from (0.22, s.f.).

### 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 y cols. (2002) and Williams y Baxter (2002) were the first to propose auto-encoders 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 y cols., 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.

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

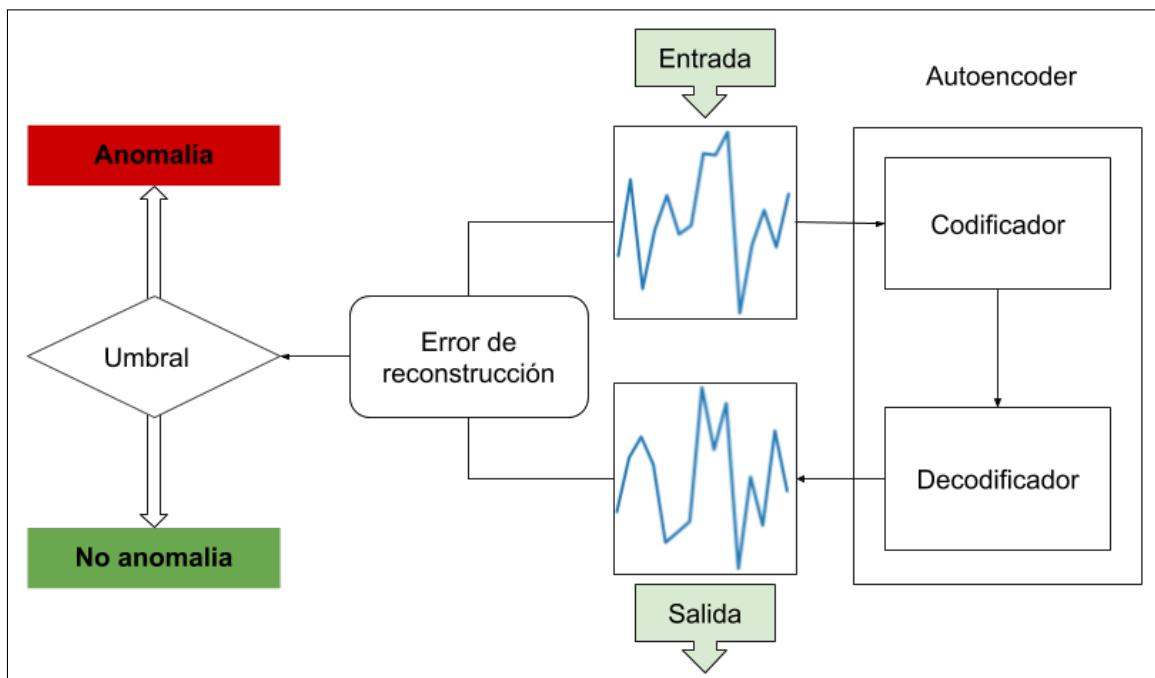


FIGURA 3.12: Detection of anomalies with autoencoder (Own elaboration).

## 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. Precisión de clasificación (accuracy)

La precisión de clasificación, conocida también con el nombre de precisión, es la relación entre el número de predicciones correctas y el número total de muestras de entrada.

$$\text{Precisión} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones realizadas}} \quad (3.9)$$

Esta métrica funciona bien si hay el mismo número de muestras que pertenecen a cada clase; por ejemplo, si se considera que se tiene un conjunto de datos que contiene 98 % de muestras que pertenecen a la clase A y 2 % que pertenecen a la clase B, el modelo podría obtener fácilmente un 98 % de precisión de entrenamiento simplemente prediciendo cada muestra de entrenamiento como clase A.

Esto conlleva que la precisión puede dar la falsa sensación de lograr una alta precisión, lo cual se convierte en un verdadero problema si se trata problemas que conllevan la detección de cosas de alto riesgo; por ejemplo, de una enfermedad rara pero mortal ya que el costo de no diagnosticar la enfermedad de una persona enferma es mucho mayor que el costo de enviar a una persona sana a realizarce más análisis.

### 3.6.2. Pérdida logarítmica (Logarithmic Loss)

La pérdida logarítmica, conocida también como *Log Loss*, funciona penalizando las clasificaciones falsas, además tiene un buen rendimiento para la clasificación de varias clases. Cuando se trabaja con Log Loss, el clasificador debe asignar una probabilidad a cada clase de todas las muestras; suponiendo que hay  $N$  muestras que pertenecen a clases  $M$ , Log Loss se calcula según la siguiente ecuación:

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

donde:

$y_{ij}$ , indica si la muestra  $i$  pertenece a la clase  $j$  o no.

$p_{ij}$ , indica la probabilidad de que la muestra  $i$  pertenezca a la clase  $j$ .

Log Loss no tiene límite superior y existe en el rango  $[0, \infty)$ . Cuando se obtiene un Log Loss cercano a 0 indica una mayor precisión, mientras que si está lejos de 0 indica una menor precisión. En general se puede decir que minimizar Log Loss proporciona una mayor precisión para el clasificador.

### 3.6.3. Matriz de confusión

La matriz de confusión, también llamada matriz de error, es el método más común para evaluar la exactitud de un resultado de clasificación (Smits, Dellepiane, y Schowengerdt, 1999). Esta matriz es una tabulación cruzada de los datos esperados y los resultados de la clasificación del modelo. El número de columnas y filas es igual al número de categorías de la clasificación y de ellas se derivan diferentes medidas estadísticas.

		Predicción	
		Clase Positiva	Clase Negativa
Reales	Clase Positiva	Verdadero Positivo (VP)	Falso Negativo (FN)
	Clase Negativa	Falso Positivo (FP)	Verdadero Negativo (VN)

CUADRO 3.1: Matriz de confusión, para una clasificación binaria (Elaboración propia).

En el Cuadro 3.1 las filas de la matriz representan los valores reales, mientras que las columnas están asociadas con los datos clasificados por el modelo (predicciones). La diagonal principal, que se presenta de color verde, indica los aciertos ó Verdaderos Positivos (VP) y Verdaderos Negativos (VN), que son todos aquellos datos donde el modelo obtiene el mismo resultado que se esperaba obtener. En cuanto a todos los demás valores de la matriz pertenecen a aquellos datos que fueron clasificados de forma errónea, estos se clasifican en dos clases: Falsos Positivos (FP), que en la matriz se presentan de color rojo y Falsos Negativos (FN) que en la matriz fueron representados de color naranja.

La precisión global de la matriz se calcula dividiendo la suma de muestras correctamente clasificadas por el número total de muestras tomadas 3.11. Este valor es una medida de clasificación como un todo, ya que indica la probabilidad de que una muestra se clasifique correctamente.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN} \quad (3.11)$$

La exactitud no es una medida adecuada para la evaluación de algoritmos de detección de valores atípicos, debido a que la mayoría de las veces un falso negativo es mucho más costoso que un falso positivo, además que los conjuntos de entrenamiento presentan una gran cantidad de datos normales comparado a la cantidad de datos anómalos. Existen dos métricas comunes para evaluar algoritmos de detección de valores atípicos, AUC y F1 Score; a continuación se profundizará detalladamente estas dos métricas.

### 3.6.4. Área bajo la curva (AUC)

Área bajo la curva (AUC<sup>9</sup>) es una de las métricas más utilizadas para la evaluación. Se utiliza para problemas de clasificación binaria.

El AUC de un clasificador es igual a la probabilidad de que el clasificador clasifique un ejemplo positivo elegido al azar más alto que un ejemplo negativo elegido al azar. Antes de definir AUC, se debe comprender los siguientes términos:

#### Tasa de Verdaderos Positivos (TPR)

La tasa de verdaderos positivos (TPR<sup>10</sup>), también conocida como sensibilidad, corresponde a la proporción de puntos de datos positivos que se consideran correctamente como positivos, con respecto a todos los puntos de datos positivos. Se define según la ecuación 3.12.

$$\text{Sensibilidad} = \frac{VP}{FN + VP} \quad (3.12)$$

#### Tasa de Verdaderos Negativos (TNR)

La tasa de verdaderos negativos (TNR<sup>11</sup>), también conocida como especificidad, corresponde a la proporción de puntos de datos negativos que se consideran correctamente como negativos, con respecto a todos los puntos de datos negativos. Se define según la ecuación 3.13.

$$\text{Especificidad} = \frac{VN}{FP + VN} \quad (3.13)$$

#### ROC (Receiver Operating Characteristics)

ROC es la curva dibujada conectando los puntos del eje X = FPR (Tasa de falsos positivos) y el eje y = TPR (Tasa de verdaderos positivos) para diferentes valores de un umbral de discriminación (límite de decisión para determinar si un valor corresponde a una clase o no) para un modelo, es decir, se elige diferentes umbrales para un modelo, se calcula el TPR y FPR para cada umbral, luego dibuja la curva ROC y finalmente se calcula el AUC, que es el área bajo la curva ROC. En la Figura 3.13 se muestra un ejemplo de la curva AUC-ROC.

Existen dos razones principales por lo que se necesita esta curva, la primera es que refleja qué tan bueno es el modelo para separar dos clases y la segunda es que ayuda a elegir el mejor

<sup>9</sup>AUC, Area Under Curve

<sup>10</sup>TPR, de sus siglas en inglés True Positive Rate, también conocido como Sensitivity.

<sup>11</sup>TNR, de sus siglas en inglés True Negative Rate, también conocido como Specificity.

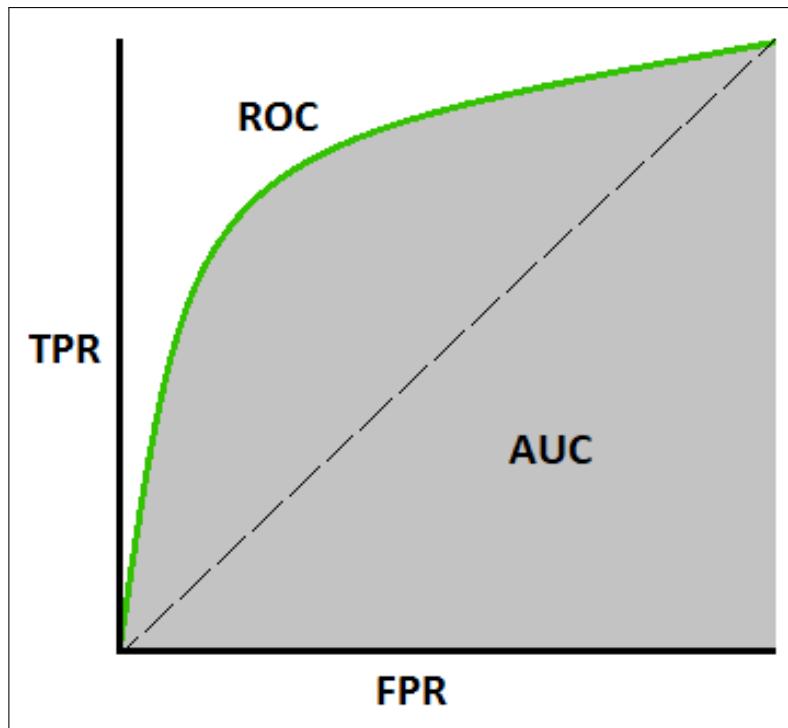


FIGURA 3.13: Ejemplo de un curva AUC-ROC (Özler, s.f.).

umbral; por ejemplo, un AUC igual a 0,5 significa que el modelo separa dos posibles resultados al azar y un AUC de 1 (valor máximo) implica una separación perfecta; por lo tanto se puede decir que mientras mayor sea el valor de AUC mejor es el rendimiento del modelo que se esté evaluando.

### 3.6.5. F1 Score

F1 Score define que tan preciso es un modelo, es decir, cuántas instancias clasifica correctamente, así como también indica que tan robusto es el modelo. Esta métrica es necesaria cuando se desea buscar un equilibrio entre la precisión y la recuperación, ya que da una evaluación justa incluso cuando el conjunto de datos se encuentra desequilibrado.

#### Precisión (Precision)

La precisión es el número de resultados positivos correctos dividido por el número de resultados positivos predichos por el clasificador.

$$\text{Precision} = \frac{VP}{VP + FP} \quad (3.14)$$

**Recuperación (Recall)**

Es el número de resultados positivos correctos dividido por el número de todas las muestras que deberían haber sido clasificadas como positivas.

$$Recall = TPR = \frac{VP}{VP + FN} \quad (3.15)$$

Por lo tanto, F1 Score se expresa matemáticamente como:

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (3.16)$$

El siguiente capítulo detalla el proceso de captura y preparación del conjunto de datos, una etapa importante, debido a que este conjunto será aquel con el que se entrenará el mecanismo de detección de anomalías propuesto.

## Capítulo 4

# CAPTURA Y PREPARACIÓN DE DATOS

Contar con una gran cantidad de datos en cualquier problema de detección de anomalías es lo que permite generar modelos más precisos, debido a que nunca se sabe qué características pueden dar indicio de una anomalía, contar con múltiples tipos de datos es lo que permite ir más allá de una mera detección de anomalías puntuales y ser capaz de identificar anomalías contextuales o colectivas más sofisticadas. Sin embargo la obtención de estos datos no siempre es una tarea sencilla, por lo que muchas veces se debe encontrar una manera de generar los mismos.

En este capítulo se detallará el método de recolección de datos que se realizó para la presente investigación y las diferentes técnicas de análisis de datos que se aplicó.

### 4.1. Captura de datos

Actualmente existen varios enfoques para acceder a la información del conductor (agente) y del vehículo. En el primer enfoque, un conjunto de sensores y hardware adicional se implementan previamente en el vehículo, por ejemplo, cajas telemáticas (cajas negras provistas por compañías de seguros de automóviles), adaptadores de diagnóstico a bordo (OBD-II) enchufados en el controlador del vehículo red de área (CAN) (Zaldivar, Calafate, Cano, y Manzoni, 2011; Araujo, Igreja, R., y Araujo, 2012), la información registrada por estos dispositivos se puede recuperar o enviar a través de Internet. Sin embargo, esta estrategia requiere que los vehículos instalen dispositivos adicionales, lo que implica un mayor costo. Para superar estos inconvenientes, existe un enfoque alternativo el cual es usar teléfonos inteligentes para recopilar datos a través de un conjunto de sensores integrados, tales como sensores iniciales (acelerómetros y giroscopios), sistemas de posicionamiento global (GPS), magnetómetros, micrófonos, sensores de imagen (cámaras), sensores de luz , sensores de proximidad, sensores de dirección (brújula), entre otros.

Para el presente trabajo de investigación se eligió el uso de teléfonos inteligentes para acceder a la información del tipo de conducción, por las razones que se presentaron anteriormente, con este enfoque se desarrolló una aplicación móvil basada en Android para recopilar datos de los sensores: acelerómetro y giroscopio, en intervalos de 1 segundo, los cuales en una primera instancia serán almacenados de manera interna en el dispositivo móvil. (Ver Figura 4.1)

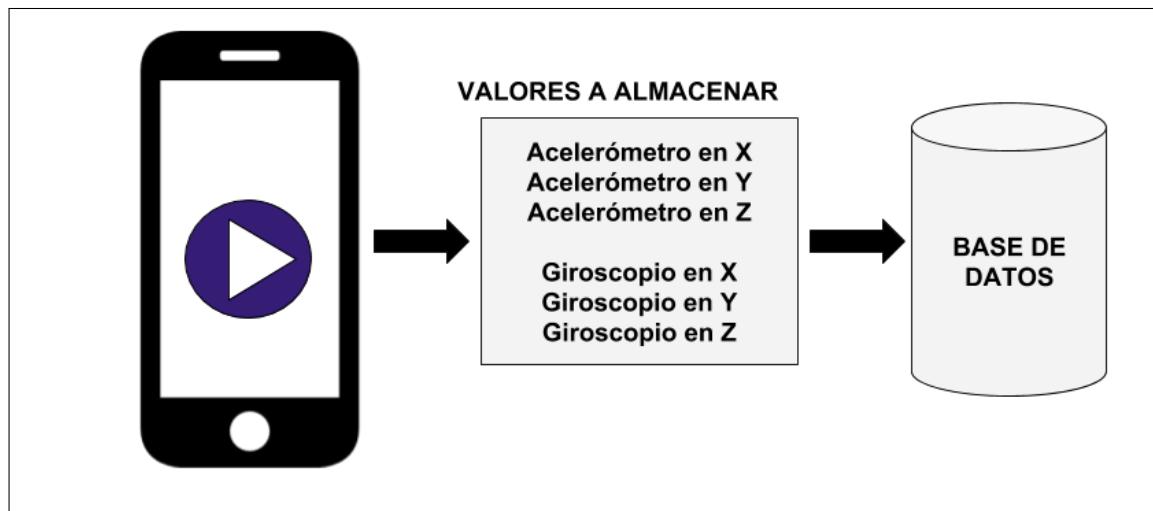


FIGURA 4.1: Recolección de datos, con intervalo de un segundo (Elaboración propia).

Para la captura de datos se usó un soporte para celular de parabrisas como se ve en la Figura 4.2; se realizó la captura en dos posiciones distintas (vertical y horizontal).



FIGURA 4.2: Soporte para celular de parabrisas, posición horizontal (Elaboración propia).

Cada captura, independientemente de la posición en la que se realizó, dió como resultado un

conjunto de datos (dataset), donde por cada tiempo T (1 seg.) se tiene seis variables: acelerómetro en X (acc x), acelerómetro en Y (acc y), acelerómetro en Z (acc z), giroscopio en X (gyr x), giroscopio en Y (gyr y) y giroscopio en Z (gyr z). En la Figura 4.3 se aprecia un fragmento del conjunto de datos que se obtuvo en una captura.

_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

FIGURA 4.3: Fragmento del conjunto de datos obtenido (Elaboración propia).

## 4.2. Preparación de datos

El Aprendizaje Automático depende en gran medida de los datos. Son el aspecto más crucial que hace posible el entrenamiento de algoritmos y explica porque el aprendizaje automático se hizo tan popular en los últimos años. El principal problema es que todos los conjuntos de datos tienen fallas, lo que hace a la preparación de datos un paso muy importante en el proceso de Aprendizaje Automático.

El propósito principal de la preparación de datos es manipular y transformar los datos en crudo, tal que, los datos puedan ser expuestos o hacerse accesibles más facilmente (Pyle, 1999), para lograr este propósito se debe seguir un proceso que implica la selección, el pre-procesamiento y la transformación de datos.

### 4.2.1. Selección de datos

La selección de datos implica los siguientes pasos:

- Seleccionar solo un subconjunto de los datos disponibles.
- Derivar o simular algunos datos a partir de los datos disponibles, en caso de ser necesario.
- Excluir aquellos datos que no son relevantes para el problema.

Para el presente trabajo sólo se hará incapié en el primer paso de esta fase, debido a que los datos con los que se cuenta son limitados ya que éstos fueron capturados para la investigación y como se indicó en la sección 4.1, esta captura se realizó, tanto de forma vertical como horizontal, a continuación se analizará las diferencias entre ellos.

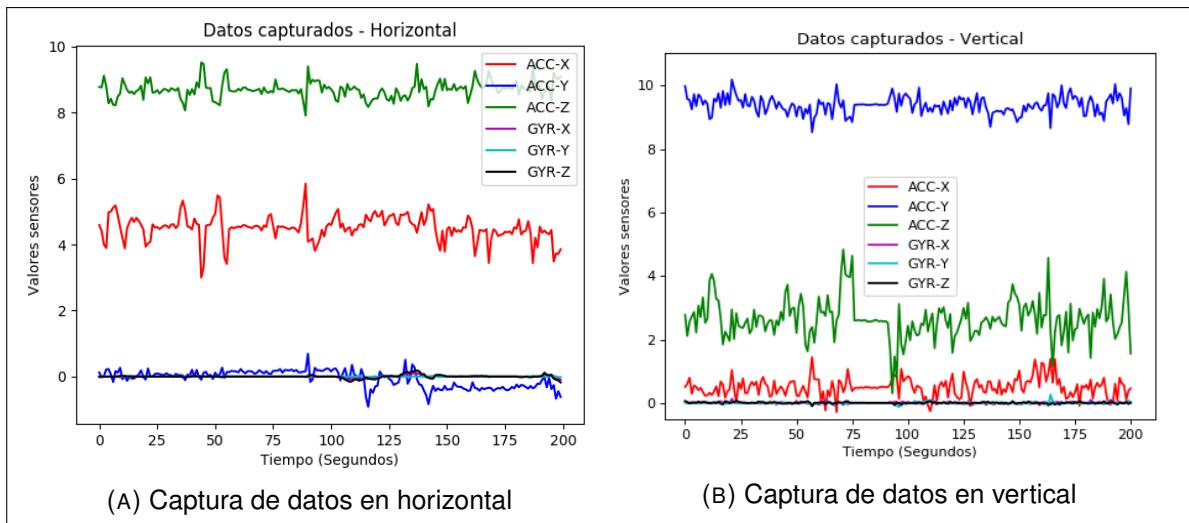


FIGURA 4.4: Gráfica de los sensores capturados en diferentes posiciones (Ela-  
boración propia).

En la Figura 4.4 se muestra fragmentos de las capturas obtenidas por el dispositivo móvil de la conducción de un mismo usuario (agente) desde diferentes posiciones.

Si bien los valores capturados, son muy similares entre sí, los datos que fueron capturados con el dispositivo móvil en posición horizontal, presentan menos ruido, esto debido a que esta posición favorece la inercia del dispositivo cuando el vehículo está en movimiento, lo cual es una gran ventaja frente a los datos que fueron capturados de forma vertical, ya que estos fueron más susceptibles a sacudirse mientras el vehículo se desplazaba haciendo que los valores capturados en esta posición presenten valores de movimiento no sólo del vehículo sino también del dispositivo móvil, lo cual no es lo que se busca en el presente trabajo.

Por las razones presentadas en el anterior párrafo se decidió trabajar con los datos capturados con el dispositivo móvil en posición horizontal, descartando así aquellos datos capturados en vertical.

### 4.3. Pre-procesamiento de datos

Una vez que se seleccionaron los datos con los que se trabajará, se debe proceder a pre-procesarlos, entrando así a la fase de Pre-procesamiento de datos, el objetivo de esta fase es reducir la cantidad de los datos, encontrar las relaciones entre ellos, normalizarlos, remover los valores atípicos y extraer las características de los datos. Esta fase incluye varias técnicas como la limpieza, integración, transformación y reducción de los datos (Suad y Wesam, 2017).

### 4.3.1. Limpieza de datos

Los datos de las filas pueden tener registros incompletos, valores ruidosos, valores atípicos y datos inconsistentes. La limpieza de los datos en la mayoría de los casos es el primer paso del pre-procesamiento de datos. Esta técnica se usa para compensar valores ausentes, suavizar el ruido en los datos, reconocer valores atípicos y corregir inconsistencias.

#### Técnicas de compensación de registros incompletos

Muchos conjuntos de datos del mundo real pueden contener valores faltantes por distintas razones; lo cual puede afectar drásticamente la calidad del modelo de aprendizaje automático.

A continuación se presentan cuatro técnicas de compensación para conjuntos de datos, con el objetivo de sobrelevar el problema de los valores ausentes.

- **Ignorar / Eliminar:** En algunos casos es mejor ignorar o eliminar la tupla que contiene valores faltantes en lugar de llenar esta. Generalmente esta técnica se practica en conjuntos de datos que son muy grandes, donde el eliminar algunos datos no afectará la información que transmite el conjunto de datos. Sin embargo cuando se trabaja con un conjunto de datos pequeño el eliminar las tuplas que contienen valores ausentes podría hacer perder información importante.
- **Llenar manualmente los valores ausentes:** Otra opción también es completar los valores faltantes si se comprende la naturaleza de los mismos, esto generalmente se realiza en conjunto de datos pequeños, ya que en los conjuntos grandes ésta tarea requeriría bastante tiempo.
- **Llenar los valores ausentes con valores centrales (Media/Mediana):** Esta técnica es mucho mejor que las presentadas anteriormente. En esta técnica se inserta la media o la mediana del atributo respectivo a los valores faltantes.
- **Interpolación:** Es una forma confiable, precisa y científica de completar valores perdidos. Para usar esta técnica primero se debe desarrollar una relación entre los atributos y luego se predice el valor más probable y preciso para los lugares faltantes, esto se puede lograr mediante regresión, formulación bayesiana e inducción por árboles de decisión.

#### Eliminar ruido (suavizado) de los datos

Para comprender esta técnica primero se debe definir qué es el ruido en los datos. El ruido en los datos es cualquier tipo de error aleatorio o variación en los atributos medidos; por otra parte los valores atípicos presentes en los datos también pueden considerarse como ruido.

Es importante destacar que el ruido presente en el conjunto de datos puede afectar en gran medida el resultado de los algoritmos de Aprendizaje Automático, por lo tanto aquellos datos que contengan ruido no son considerados buenos datos y deben ser eliminados en lo posible.

Sin embargo antes de eliminar estos, se debe ser capaz de detectarlos; para lograr este objetivo existen muchas técnicas que se pueden utilizar, una de estas técnicas es la **Visualización de datos**, la cual consiste en obtener una representación visual de los datos, para poder evidenciar si existe ruido en los datos y/o valores atípicos.

En el presente trabajo se graficó histogramas de las frecuencias de cada característica del conjunto de datos y así visualizar de mejor manera si existe ruido o valores atípicos en el mismo. En la Figura 4.5 se puede evidenciar que los valores de cada sensor presentan asimetrías negativas y positivas, además de tener muchos valores bastante alejados de la media, lo que da un indicio de posibles valores atípicos.

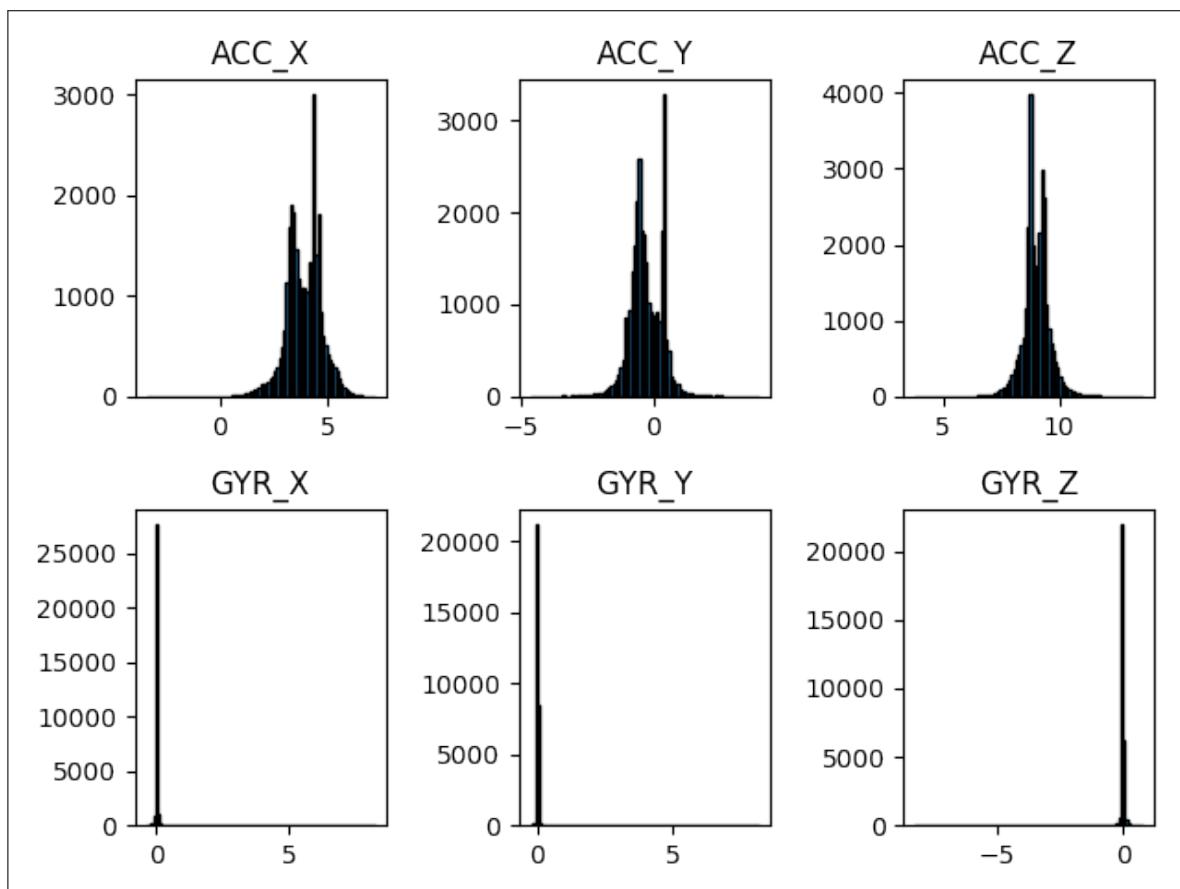


FIGURA 4.5: Histograma de frecuencias del conjunto de datos (Elaboración propia).

En la Figura 4.6 se presenta una tabla con estadísticas descriptivas del conjunto de datos, esta tabla presenta: la cantidad de datos, su media, su desviación estándar, el valor mínimo y máximo del conjunto de datos y los percentiles 50, 25 y 75, cabe recalcar que el percentil 50 es el mismo que la mediana.

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

FIGURA 4.6: Tabla de resultados estadísticos del conjunto de datos (Elaboración propia).

Con la información proporcionada en la Figura 4.6 ahora es más sencillo realizar un análisis del conjunto de datos, en primer lugar se evidencia que los valores obtenidos durante la captura, presentan desviaciones estándar muy pequeñas entre 0.05 y 0.81 y sin embargo la diferencia entre los valores mínimo y máximo son realmente grandes, por ejemplo para el Acelerómetro en X, la diferencia es 10 aproximadamente (Valor mínimo: -3.40 y valor máximo: 7.17), esto quiere decir que el conjunto de datos con el que se trabaja presenta mucho ruido, considerando como ruido o valores atípicos, aquellos valores muy alejados de la media.

Para eliminar el ruido que presenta el conjunto de datos se aplicó la **Regla 68-95-99.7**, conocida también como la **Regla empírica**, donde suponiendo que el conjunto de datos con el que se trabaja tiene una distribución normal, la desviación estándar se puede usar para determinar la proporción de valores que se encuentran dentro de un rango particular del valor medio. Para tales distribuciones, siempre ocurre que el 68 % de los valores están a menos de una desviación estándar (1SD) del valor medio, que el 95 % de los valores están a menos de dos desviaciones estándar (2SD) de la media y que el 99 % de valores están a menos de tres desviaciones estándar (3SD) de la media. En la Figura 4.7 se muestra este concepto de forma esquemática.

En la Figura 4.8, se puede observar como se comporta la regla 68-95-99.7 sobre los valores del sensor del acelerómetro en Z, por lo que para eliminar el ruido del conjunto de datos se cuenta con dos opciones, eliminar los valores después de tres desviaciones estándar de la media, en caso de considerar que el conjunto de datos presenta pocas anomalías o valores atípicos, o eliminar los valores después de dos desviaciones estándar en caso de estar seguro que el conjunto de datos presenta una gran cantidad de ruido en los datos, en este caso la mayoría de los datos pertenecen a comportamientos normales de conducción por lo que sólo se eliminará los valores que se encuentran después de tres desviaciones estándar de la media.

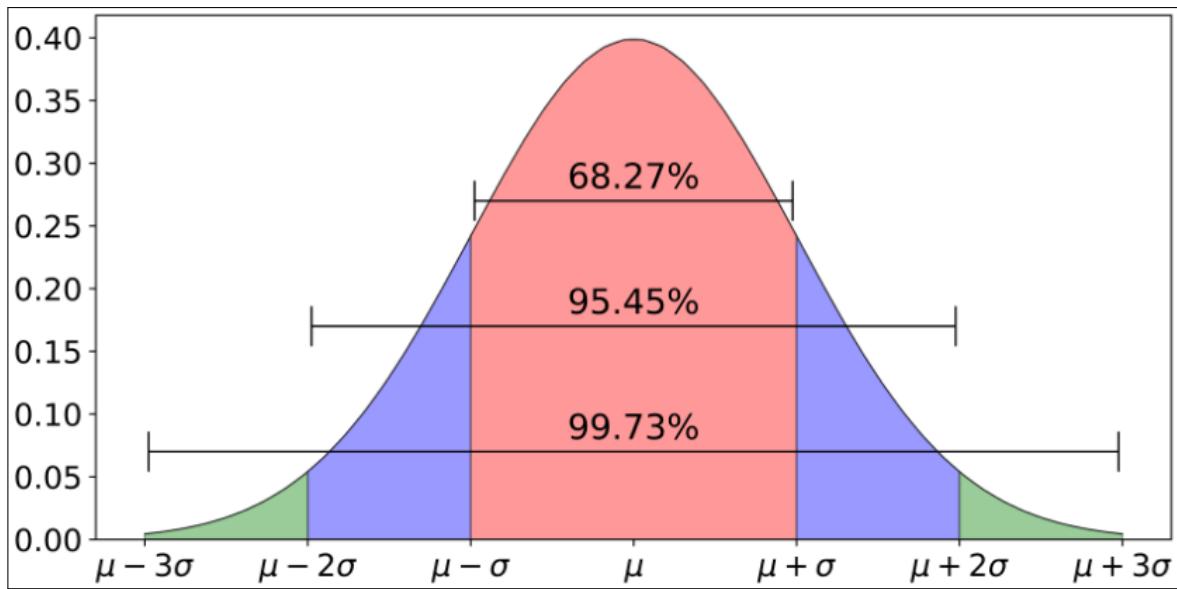


FIGURA 4.7: Regla 68-95-99.7 (Galarnyk, s.f.).

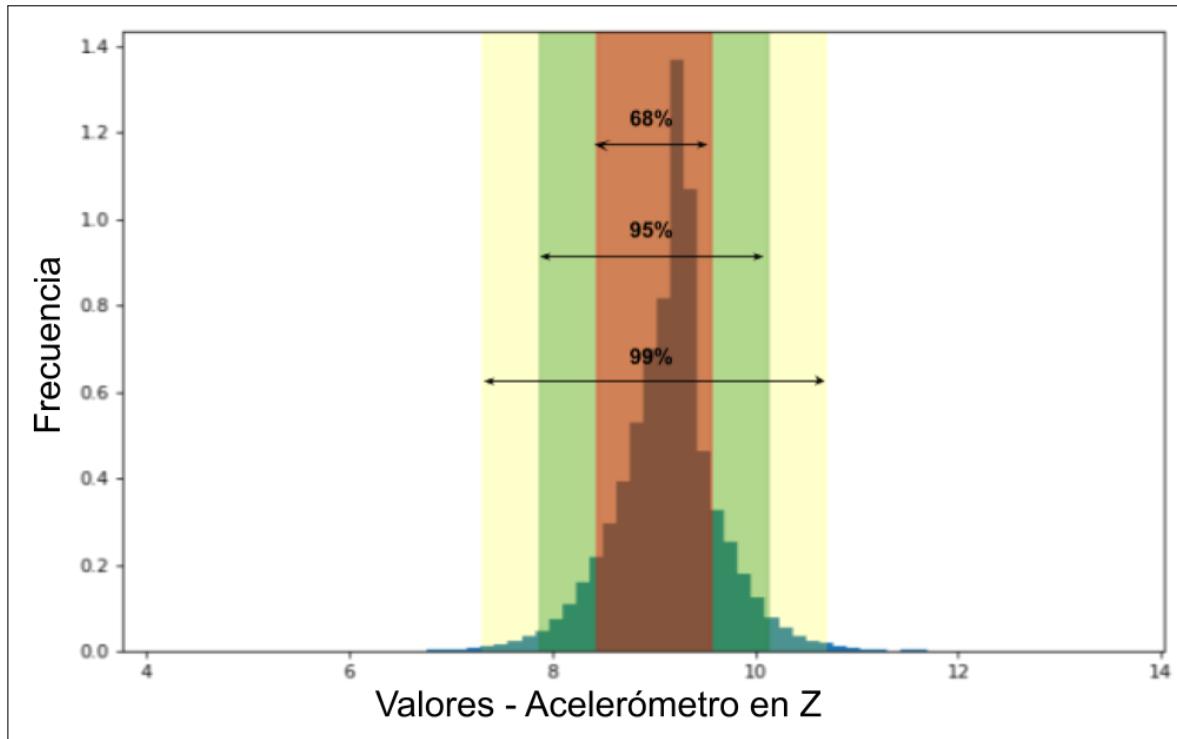


FIGURA 4.8: Regla 68-95-99.7 aplicada a los valores de los sensores del acelerómetro en Z (Elaboración propia).

### Fusión o integración de datos

Cuando se trabaja con datos del mundo real, es posible que los datos que se requiere no se encuentren en un mismo conjunto de datos, en éstos casos, se necesita recopilar datos de

diferentes fuentes y fusionarlos en un solo conjunto de datos; este proceso recibe el nombre de Fusión o Integración de datos. Uno de los problemas más comunes de este proceso es la redundancia.

Este proceso no se aplicó en la investigación debido a que sólo se capturó los datos por medio del dispositivo móvil, por lo cual no se tiene el problema de tener más de una fuente de datos.

### Transformación de datos

La transformación de los datos es el proceso donde se cambia la naturaleza de los datos, dicho proceso usa algunas estrategias para poder extraer información importante del conjunto de datos; algunas de las estrategias para la transformación de datos son:

- **Agregación:** En esta técnica, la operación de resumen o agregación se aplica sobre los datos. Por ejemplo: los datos de ventas diarias se pueden usar para calcular el monto mensual y anual en ventas, para posteriormente agregar estos datos calculados al conjunto de datos.
- **Discretización:** Con esta técnica se construye y reemplaza valores en bruto de un atributo numérico por valores de intervalo.
- **Construcción de atributos / Ingeniería de características:** Esta técnica es útil para generar información adicional a partir de aquellos datos que no son lo suficientemente representativos por sí mismos, además puede ser adecuada cuando se tiene menos características pero aún contienen información oculta para extraer.
- **Normalización / Estandarización:** La normalización o estandarización se define como el proceso de reescalar datos originales sin cambiar su comportamiento o naturaleza. Se define un nuevo límite (generalmente entre 0 y 1) y se convierte los datos en consecuencia. Esta técnica es útil en algoritmos de clasificación que involucran redes neuronales o algoritmos basados en la distancia (por ejemplo, KNN<sup>1</sup>, K-Means<sup>2</sup>, que se utiliza para la agrupación. Este algoritmo es capaz de aprender gradualmente cómo agrupar valores no etiquetados en grupos mediante un análisis de la distancia media de dichos valores.). Algunas técnicas de normalización son:
  - **Normalización Min-Max:** En este método, cada entrada se normaliza entre unos límites definidos:

$$x_{normalizado} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4.1)$$

---

<sup>1</sup>KNN, es un algoritmo de clasificación (o regresión) que, para determinar la clasificación de un punto, combina la clasificación de los  $K$  puntos más cercanos.

<sup>2</sup>K-Means, es un algoritmo de agrupamiento que intenta dividir un conjunto de puntos en  $K$  grupos; de modo que los puntos en cada grupo tienden a estar cerca uno del otro.

Presenta el problema de que comprime los datos de entrada entre unos límites fijos, que por lo general son 0 y 1. Esto quiere decir que si existe ruido, éste va a ser ampliado, lo que hace que este método no sea adecuado para señales estables.

- *Escalado estándar (Standard Scaler)*: Es un método alternativo al escalado de variables, consiste en restar a cada dato la media de la variable y dividirlo por la desviación típica.

$$x_{\text{normalizado}} = \frac{x - x_{\text{media}}}{x_{\text{desvSt}}} \quad (4.2)$$

Este método es adecuado para normalizar señales estables, no obstante, tanto la media como la desviación estándar son muy sensibles a valores anómalos. Una alternativa de solución de ésto, es la eliminación de anomalías antes de realizar la normalización.

- *Escalado sobre el valor máximo*: Este método, presenta la idea de escalar los datos dividiendo éstos entre su máximo valor.
- *Escalado robusto (Robust scaler)*: El escalado robusto consiste en eliminar la mediana y escala los datos de acuerdo con el rango de intercuartil (IQR). Este método es robusto para valores atípicos.

## Reducción de datos

Este proceso se basa en la adopción de algunas estrategias, tal que, el análisis de datos reducidos produce la misma información producida por los datos originales. Algunas de las estrategias incluyen: análisis de componentes principales (PCA), selección de un subconjunto de atributos, agrupación y muestreo entre otros.

### Análisis de componentes principales (PCA)

El Análisis de Componentes Principales (PCA - Principal Component Analysis) es una técnica que se usa ampliamente para aplicaciones como reducción de dimensionalidad, compresión de datos con pérdida, extracción de características y visualización de datos (Bishop, 2006).

PCA es una técnica estadística no supervisada y no paramétrica, que se utiliza para la reducción de dimensionalidad. Esta técnica es un paso importante debido a que la alta dimensionalidad en el campo de aprendizaje automático puede llevar al sobreajuste del modelo, reduciendo así su capacidad de generalización, Bellman (2003) describe este fenómeno como la "Maldición de la Dimensionalidad". Además, el uso de esta técnica puede mejorar directamente el rendimiento de los modelos de Aprendizaje Automático.

PCA combina las variables de entrada de una manera específica, luego se deshace de las variables "menos importantes" y al mismo tiempo conserva las partes más valiosas (o componentes principales<sup>3</sup>) de todas las variables.

Cuando se usa PCA enfocado al aprendizaje automático se sigue los siguientes pasos:

1. Dividir el conjunto de datos  $d$ -dimensional en conjunto de entrenamiento, desarrollo y prueba.
2. Estandarizar / Normalizar el conjunto de datos según el conjunto de entrenamiento.
3. Construir la matriz de covarianza.
4. Descomponer la matriz de covarianza en sus vectores y valores propios.
5. Ordenar los valores propios disminuyendo el orden para clasificar los vectores propios correspondientes.
6. Seleccionar  $k$  vectores propios que corresponden a los  $k$  valores propios más grandes, donde  $k$  es la dimensionalidad del nuevo subespacio de entidad ( $k \leq d$ ).
7. Construir una matriz de proyección  $\mathbf{W}$  a partir de los "top" $k$  vectores propios.
8. Transformar el conjunto de entrenamiento de entrada  $d$ -dimensional  $\mathbf{X}$  utilizando la matriz de proyección  $\mathbf{W}$  para obtener el nuevo subespacio de característica  $k$ -dimensional.

### ***División del conjunto de datos***

Dado que se cuenta con un conjunto de datos para generar el modelo de Aprendizaje Automático, este se divide en tres partes: conjunto de entrenamiento<sup>4</sup>, desarrollo<sup>5</sup> y prueba<sup>6</sup>, sin embargo esto no es una tarea trivial; ya que si no se hace correctamente el resultado puede ser desastroso.

En el trabajo de Moindrot y Genthal (2018) se dice que la división del conjunto de datos tiene un gran impacto en la productividad, por lo cual es importante que al elegir los subconjuntos estos deben tener la **misma distribución** y deben ser elegidos aleatoriamente del conjunto de datos.

<sup>3</sup>**Componente principal**, es una combinación lineal normalizada de las características originales en el conjunto de datos.

<sup>4</sup>**Conjunto de entrenamiento**, es la muestra de datos utilizada para ajustar el modelo de Aprendizaje Automático.

<sup>5</sup>**Conjunto de desarrollo**, es la muestra de datos utilizada para proporcionar una evaluación imparcial de un modelo ajustado con el conjunto de entrenamiento mientras se ajustan los hiperparámetros del modelo.

<sup>6</sup>**Conjunto de prueba**, es la muestra de datos utilizada para proporcionar una evaluación imparcial de un ajuste final del modelo en el conjunto de entrenamiento.

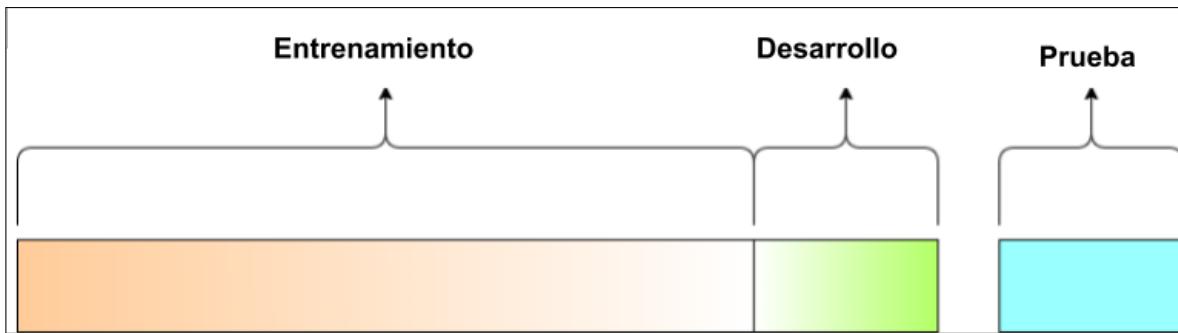


FIGURA 4.9: División del conjunto de entrenamiento (Elaboración propia).

Conjunto de entrenamiento	70 %	21000
Conjunto de desarrollo	15 %	4500
Conjunto de prueba	15 %	4500
Conjunto de datos	100 %	30000

CUADRO 4.1: Tabla de división del conjunto de datos (Elaboración propia).

Por otra parte el tamaño del conjunto de desarrollo y prueba deben ser lo suficientemente grandes como para que los resultados del desarrollo y prueba sean representativos para el rendimiento del modelo. Para conjuntos de datos grandes (mayores a un millón), el conjunto de desarrollo y prueba puede tener alrededor de 10000 ejemplos cada uno, es decir, el 1 % del total de datos.

Otras consideraciones que deben tomarse en cuenta en la práctica son:

- La división del conjunto de entrenamiento/desarrollo/prueba siempre debe ser la misma para todos los experimentos, por lo tanto se debe contar con script reproducible para crear la división entrenamiento/desarrollo/prueba.
- Se debe probar que los conjuntos de desarrollo y prueba provengan de la misma distribución.

En la presente investigación el conjunto de datos cuenta con 30000 ejemplos, por lo que la división de el conjunto de datos será la que se observa en el Cuadro 4.1.

### ***Estandarizar / Normalizar el conjunto de datos***

En la sección 4.3.1 ya se describió lo que es la estandarización o normalización de datos y algunos de los tipos de escalado que existen; por lo tanto esta sección sólo se limitará a la elaboración de un análisis para decidir que técnica de escalado es la más adecuada para el conjunto de datos, en la Figura 4.10 se puede apreciar una fracción del conjunto de datos capturado, la cual es la base con la que se realizará el análisis comparativo con los distintos tipos de escalado.

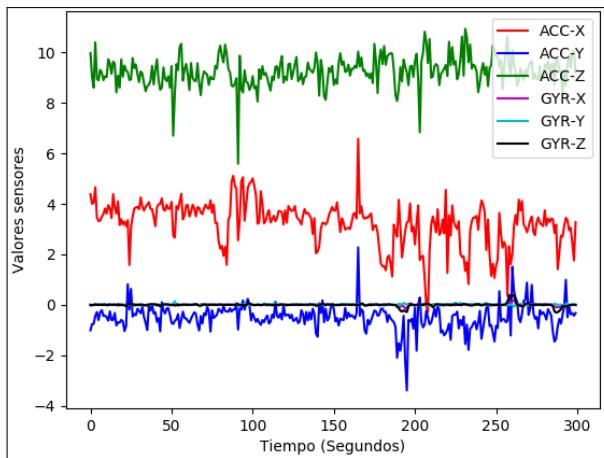


FIGURA 4.10: Visualización de los parámetros de conducción capturados (Elaboración propia).

Para probar los diferentes tipos de escalado, se los ajustó con los datos que ya no presentan ruido o valores atípicos del conjunto de entrenamiento.

El primer tipo de escalado que se realizó sobre el conjunto de datos capturados fue la **Normalización Min-Max**, el cual se realizó entre los límites 0 y 1, los resultados obtenidos se muestran en la figura 4.11a, donde se puede apreciar que los valores del acelerómetro, en sus tres ejes, no se ven deformados después de haber sido escalados con ésta técnica y los valores del giroscopio, los cuales son más estables se tornan deformados, considerando como datos estables aquellos datos que se presentan como una línea en cero con pocas fluctuaciones; esta deformación puede ser ventajosa al hacer más visible pequeñas curvas que anteriormente eran imperceptibles, sin embargo conlleva el peligro de que pueda ampliar ruido existente en los datos que no pudimos eliminar en el paso previo a esta tarea.

La segunda técnica de normalización que se aplicó sobre los datos fue el **escalado estándar**, el resultado se puede apreciar en la figura 4.11b, observando detalladamente los resultados se puede evidenciar que son muy similares a los obtenidos con la normalización Min-Max, las únicas diferencias que se pueden evidenciar son que el nuevo rango de los datos es más amplio con media en cero y valores que oscilan principalmente entre 2 y -2, y que se amplia un poco más algunas de las fluctuaciones que presentan los giroscopios.

Para la tercera normalización de datos se aplicó la técnica de **escalado sobre el valor máximo**, los resultados se presentan totalmente diferentes a los que se obtuvo anteriormente, sin embargo se observa claramente que los valores del acelerómetro no presentan mucho cambio, con la diferencia de que la media de estos valores son distintas para cada uno, y los valores de los giroscopios se comportan como en los anteriores, como se puede ver en la figura 4.11c. Esta técnica presenta los peores resultados, ya que no deja el conjunto de datos en un mismo rango lo cual complica el trabajo con dichos datos.

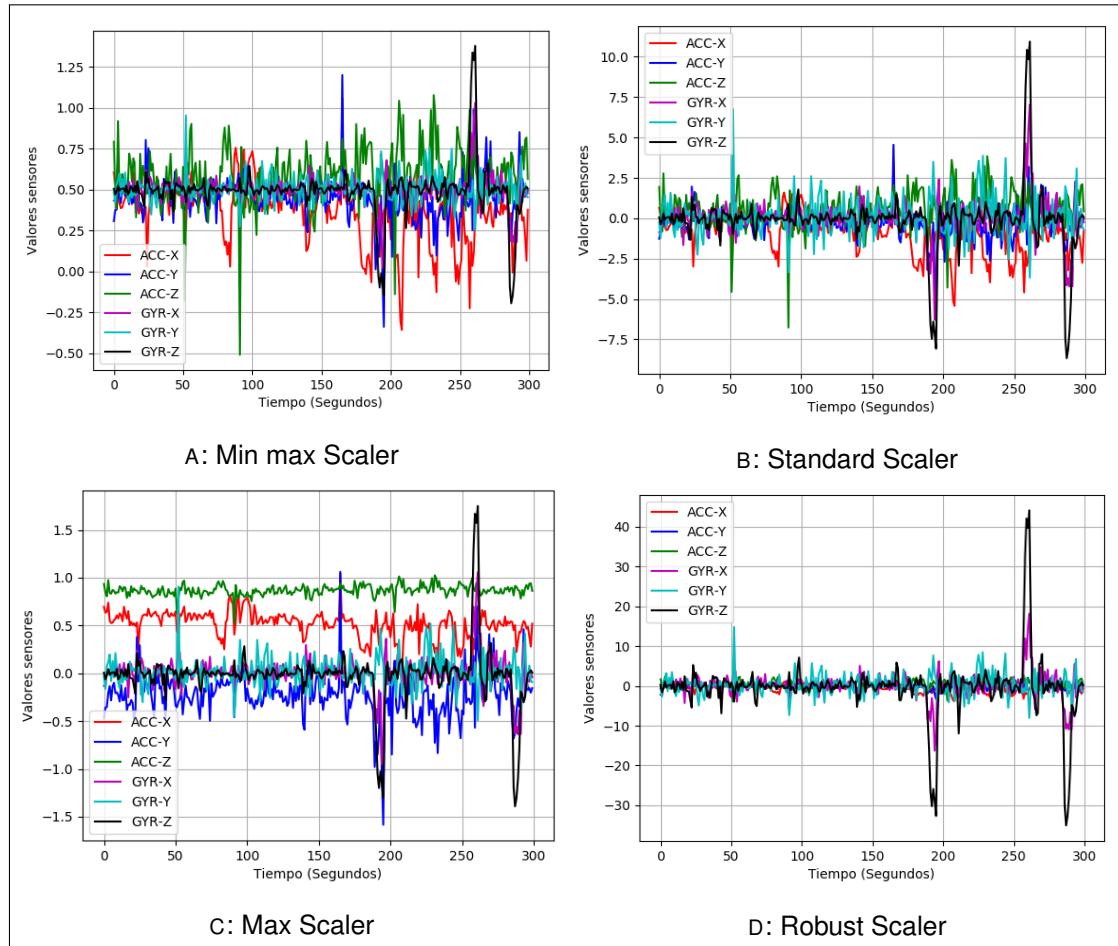


FIGURA 4.11: Gráfica resultante de aplicar diferentes tipos de normalizaciones a un conjunto de datos (Elaboración propia).

La última técnica aplicada es el **escalado robusto**, su resultado puede ser observado en la figura 4.11d, el cual puede ser considerado bastante similar a las dos primeras técnicas, con la diferencia de que este escalado pronuncia mucho más las fluctuaciones de los valores de los giroscopios, convirtiéndolos incluso a una escala superior que el de los valores de los acelerómetros.

Para decidir el mejor método de normalización que se puede aplicar a los datos capturados, primero se descartará completamente el **método de escalado sobre el valor máximo** debido a que los resultados que presentó fueron totalmente desalentadores ya que los valores después del escalado no compartían los mismos límites, por otra parte los restantes tres tipos de escalado son muy similares, lo cual complica la elección de escalado correcto, sin embargo cuando se usa PCA se debe ser muy cuidadoso, debido a que si se tiene una variable con una desviación estándar alta, esta tendrá un mayor peso en el cálculo del eje que una variable con una desviación estándar baja, por lo cual este puede ser un parámetro de decisión importante para elegir el tipo de escalado más adecuado.

En la tabla 4.2 se muestra la media y la desviación estándar de cada variable después de haber aplicado los diferentes tipos de escalado sobre los datos. Como se puede evidenciar en los escalados sobre el valor máximo y estándar, las desviaciones estándar son muy diferentes, en cuanto al escalado Min-Max las desviaciones estándar de cada variable son casi las mismas ya que todas oscilan entre 0.166814 y 0.169386, lo cual se adecúa muy bien para el análisis de componentes principales, mencionando además que esta técnica es la que mejor conserva la información relevante del conjunto de datos, por lo cual esta técnica fue la seleccionada debido a que es la más conveniente para esta etapa.

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

CUADRO 4.2: Tabla con estadísticas descriptivas de los datos escalados con diferentes técnicas (Elaboración propia).

### **Aplicar PCA al conjunto de datos**

Aunque al inicio de esta subsección se explica en detalle algunos de los pasos del funcionamiento interno de PCA, existe una clase llamada PCA implementada en SCIKIT-LEARN, la cual automatiza todos los pasos siguientes; sin embargo se necesita determinar cuántas características (componentes principales) mantener y cuántas eliminar; para ésta tarea existen tres métodos comúnmente utilizados, los cuales serán descritos a continuación.

- Seleccionar arbitrariamente cuántas dimensiones se desea mantener, dependiendo del caso de uso. Por ejemplo, para un enfoque de visualización se podría elegir 2 o 3 características.
- Calcular la proporción de la varianza para cada característica, elegir un umbral y agregar características hasta llegar o superar el umbral elegido.
- Este método está estrechamente relacionado con el anterior, debido a que se calcula la proporción de varianza para cada característica, se ordena las características por la proporción de varianza y se traza la proporción de varianza acumulada explicada a medida que mantiene las características (este diagrama es conocido como *diagrama de pantalla*). Se puede elegir cuántas características incluir al identificar el punto en el que se agrega una nueva característica tiene una caída significativa en la variación en relación con la característica anterior, y elegir la cantidad de características que existen hasta ese punto. Este método suele ser conocido como "*Encontrar el codo*".

Para el desarrollo de la presente investigación se descartó tanto el primer y último método de los listados anteriormente, esto debido a que el primer método no tiene la certeza de que la cantidad de características que se elige sea lo suficientemente descriptiva para el conjunto de datos; mientras que el último método no cuenta con una definición matemáticamente precisa, debido a que sólo encuentra el "codo", por lo que también quita el control de la cantidad de variabilidad total en los datos que se obtiene finalmente.

Una vez descartados los métodos que no se ajustan al requerimiento del presente trabajo, la única opción restante es el segundo método, siendo así este el que se aplicará para el presente trabajo. Por lo tanto primero se definió como 90 % el umbral de variabilidad total que se desea preservar en el conjunto de datos, posteriormente haciendo uso de la clase PCA de SCIKIT-LEARN se calcula la varianza de cada característica y posteriormente se grafica los resultados (Ver Figura 4.12).

Y por último se debe definir que cantidad de componentes principales conservan el 90 % de la varianza de los datos como mínimo. La Gráfica 4.12 indica que seleccionando 4 componentes se puede preservar alrededor del 97.7 % de la varianza total de los datos, seleccionando 3 componentes se conserva alrededor del 92.3 % y seleccionando 2 se conserva el 85 % de la varianza; por lo tanto se decidió el uso de 3 características con lo cual se conservará el 92.3 % de la varianza total del conjunto de datos.

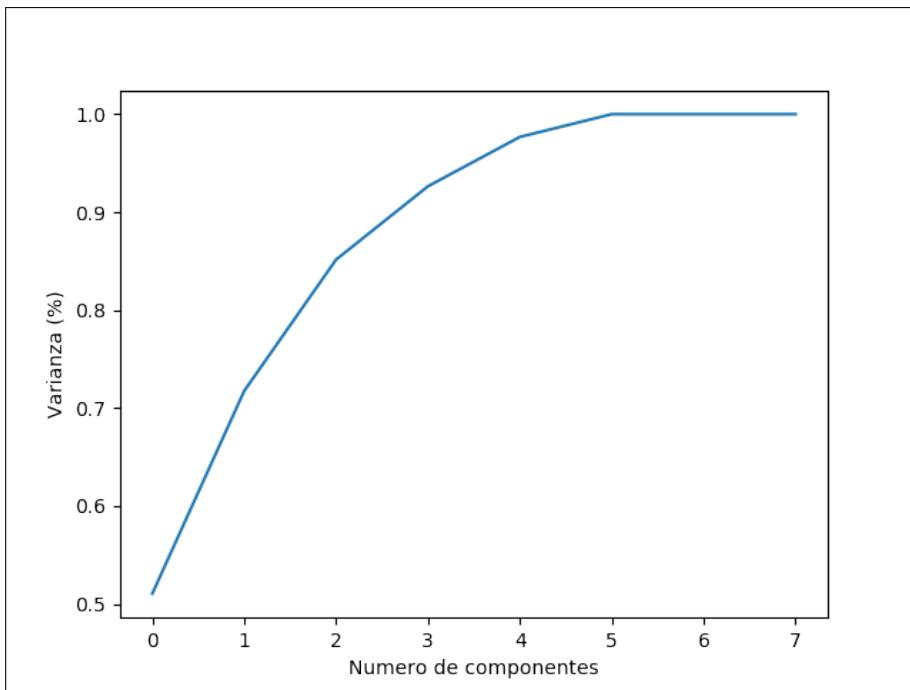


FIGURA 4.12: Gráfico de la varianza vs. el número de componentes (Elaboración propia).

En diversos estudios (Zenon, 2011; Klos y Waszczyszyn, 2011) se probó que la aplicación de PCA como pre-procesamiento del conjunto de datos es una etapa importante, debido a que no sólo sirve para la compresión de los datos de entrada, sino que también proporciona una mejora satisfactoria de la precisión de los modelos de Aprendizaje Automático; razón por la cual esta técnica forma parte de la etapa de pre-procesamiento de este trabajo.

## Capítulo 5

# GENERACIÓN DEL MECANISMO DE DETECCIÓN DE ANOMALÍAS

Este capítulo describe el proceso que se siguió para la generación del mecanismo de detección de anomalías de conducción. Según lo repasado en el capítulo 3, en el presente trabajo, se propone un detector de valores atípicos con un enfoque semi-supervisado; sin embargo, antes de profundizar en el método propuesto se debe realizar una descripción del entorno de desarrollo con el se que desarrolló el experimento, además de realizar un repaso del conjunto de datos con el que se cuenta en este estudio.

### 5.1. Entorno de desarrollo

El experimento de este estudio fue desarrollado en una computador portátil con las siguientes características:

- Procesador Intel Core i5-5200U 2.2GHz (c/TB 2.7 GHz).
- 8 GB de memoria RAM.
- Sistema Operativo ArchLinux version 4.15.15-1-ARCH (64 bits).

Es importante aclarar que el tipo de métodos utilizado en este estudio suelen ser mucho más eficientes en computadores que cuenten con una unidad de procesamiento gráfico (GPU), dado que esta unidad permite el procesamiento en paralelo. El código desarrollado en este trabajo fue escrito en PYTHON, un lenguaje de programación interpretado que se enfatiza en la simplicidad y legibilidad de código, además de que se potencia con el apoyo de poderosas librerías científicas tales como NUMPY, SCIPY, OPENCV, KERAS, MATPLOTLIB, SEABORN, etc. En cuanto al desarrollo de experimentos y la generación del mecanismo de detección esta desarrollado en Jupyter Notebook que es una aplicación web local basado en Python que permite visualizar y ejecutar documentos que contienen código fuente y ecuaciones. Las versiones de las herramientas utilizadas son detalladas a continuación:

- **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. Conjunto de datos normales y anómalos

En el Capítulo 4 se describió el proceso de captura y preparación del conjunto de datos, así como también su división en conjunto de entrenamiento/desarrollo/prueba; sin embargo cabe aclarar que aquel capítulo sólo se enfocó en el **conjunto de datos normales**.

A pesar de que se cuenta con una gran cantidad de datos normales, es necesario recolectar muestras que corresponden a anomalías, con el objetivo de poder validar el método que se propone en este proyecto. Por lo tanto, se realizó la captura de un conjunto de **datos anómalos**, el cual está conformado según el Cuadro 5.1.

Tipo de anomalía	No. anomalías	No. datos
Giros en Zig Zag	5	105
Giros a la derecha e izquierda a alta velocidad	7	35
Frenos en seco	6	24

CUADRO 5.1: Tabla del conjunto de anomalías (Elaboración propia).

Como se mencionó en el párrafo anterior el conjunto de anomalías fue capturado para validar el método propuesto, por lo tanto este conjunto se etiquetó como positivo (con la etiqueta 1) y el conjunto de datos normales como negativo (con la etiqueta 0).

### 5.2.1. Generación de series temporales

Para la generación del modelo detector de anomalías se decidió ir más allá de una simple detección de anomalías puntuales y así poder detectar anomalías contextuales o colectivas; debido a ello se requiere el uso de datos en series de tiempo.

Los datos capturados por el dispositivo móvil, son dependientes del tiempo cronométrico en el que fueron capturados (un dato por segundo); por lo cual el primer paso a realizar es la generación de pequeñas fracciones de series temporales. En la Figura 5.1 se presenta los resultados de diferentes tamaños de series de tiempo, observando estos resultados en primera instancia se descarta la serie de tiempo que cuenta con dos pasos; debido a que no es lo

suficientemente descriptiva. En cuanto a las series de tiempo restantes no es posible definir aún cual es la cantidad correcta de pasos, por lo cual, será un parámetro a optimizar en los diferentes experimentos que se realizará en las siguientes secciones. Cabe recalcar que el dominio de ésta variable está entre 3 y 5 pasos.

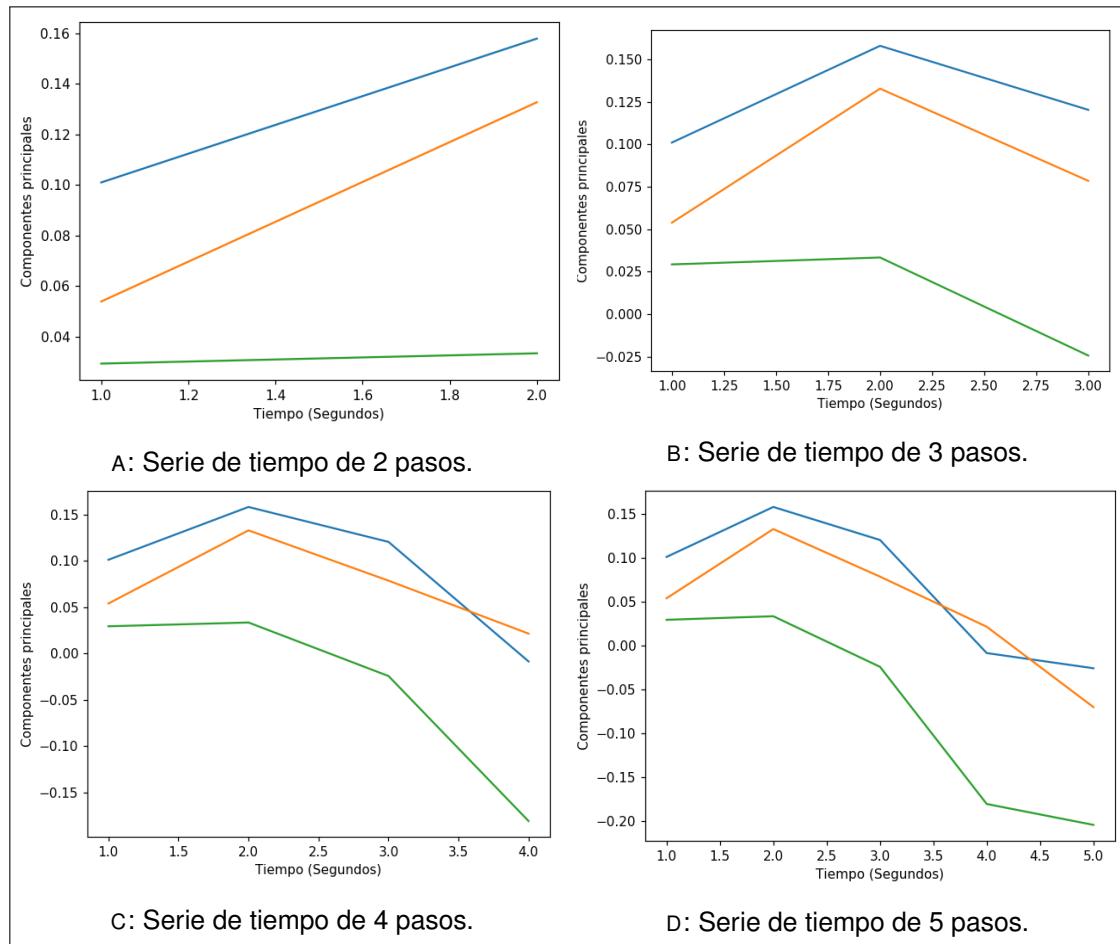


FIGURA 5.1: Gráfica resultante de diferentes tamaños de series de tiempo (Ela-  
boración propia).

### 5.3. **Modelo de detección de anomalías**

La presente investigación propone un método de detección de anomalías de conducción siguiendo un enfoque semi-supervisado, el cual consta de dos componentes: un **modelo del comportamiento normal** y un **método para la detección de valores atípicos**.

Por lo tanto, se realizó la comparación entre 3 diferentes métodos de detección, y según el rendimiento de cada uno se eligió la mejor opción. En el Cuadro 5.2 se presenta los tres diferentes métodos que fueron comparados; donde se puede observar que en todos los casos se usa un autoencoder como modelo del comportamiento normal, de esta manera, las siguientes secciones describirán la elección del autoencoder y la elección de uno de los tres diferentes métodos de detección de anomalías propuestos.

Método	Descripción
AE_T	Método de detección basado en autoencoders y umbralización (Thresholding).
AE_IF	Método de detección basado en autoencoders y aplicación de Isolation Forest.
AE_OC-SVM	Método de detección basado en autoencoders y aplicación de One-Class SVM.

CUADRO 5.2: Tabla de los métodos comparados (Elaboración propia).

### 5.3.1. Modelo del comportamiento normal

Esta etapa es una de las partes más importantes de éste trabajo, debido a que el rendimiento del modelo de detección de anomalías depende en gran parte de la precisión de esta etapa.

#### Arquitectura del modelo

Como se mencionó en la anterior sección en esta etapa se utilizará un autoencoder como modelo ajustado al comportamiento normal de conducción. Por lo cual el autoencoder se entrenó con el conjunto de datos normales, de manera que el modelo aprenda a generar sólo las clases que se consideran normales y, con suerte, tendrá problemas para reconstruir anomalías, debido a que estas muestras no fueron presentadas durante el entrenamiento.

Para ello se probó con diferentes arquitecturas, primero la forma más simple que sólo se basa el uso de capas densas (completamente conectadas), luego se hizo pruebas con redes convolucionales y por último con redes recurrentes haciendo uso específico de capas LSTM. Por cada tipo de red se hizo la prueba con 3 diferentes tipos de entrada, es decir, se probó una diferente cantidad de pasos (entre 3 y 5) en las series temporales. Por lo tanto se realizaron 9 diferentes experimentos, de los cuales por cada tipo de red sobresalió una (usando la precisión de las redes como tipo de evaluación para desarrollar las comparaciones).

En el Cuadro 5.3 se presenta la red que obtuvo el mejor resultado de todas las Redes Densas que se probaron, esta red corresponde a la red que fue alimentada con secuencias de 3 pasos. Esta red cuenta con una capa de entrada (Input), una capa de aplanamiento (Flatten) esto debido a que la capa de entrada recibe una entrada bidimensional, un conjunto de capas densas

(Dense) que van comprimiendo la información de los datos de entrada para posteriormente reconstruirlos, y por último la capa de salida es solo una capa para modificar la forma de la salida (Reshape); otro punto importante a resaltar es que las capas internas usan *elu* como función de activación y la última capa densa utilizan una función de activación tangencial (*tanh*), esto se debe a que el conjunto de datos, posterior a la obtención de componentes principales, se encuentra en el rango  $(1, -1)$ .

Arquitectura Densa				
NN_33				
	Tipo	Salida	Activación	# Parámetros
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

CUADRO 5.3: Arquitectura densa para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia).

Por otra parte el Cuadro 5.4 se presenta la red que obtiene la mejor precisión de todas las Redes Convolucionales que fueron probadas, esta red al igual que la anterior corresponde a la red que fue alimentada con secuencias de 3 pasos. Su arquitectura consta de una capa de entrada (Input), una combinación de capas de convolución de una dimensión (Conv1D) y agrupación (MaxPooling1D) hasta comprimir los datos a una dimensión de (2,4), luego un conjunto de capas convolucionales y de muestra ascendente (Upsampling1D) para decodificar la información compresa. Cabe recalcar que esta red también usa la función de activación tangencial en su última capa por las razones que se explicaron en el párrafo anterior.

Arquitectura Convolucional				
CNN_33				
	Tipo	Salida	Activación	# Parámetros
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

CUADRO 5.4: Arquitectura convolucional para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia).

En el Cuadro 5.5 se muestra la red que obtuvo la mejor precisión de todas las Redes Recurrentes probadas, como en los anteriores casos ésta red es alimentada con secuencias de 3 pasos. Dicha red cuenta con una capa de entrada (Input), dos capas LSTM una que retorna sus secuencias y una que no, luego viene una capa de redimensionado, posteriormente dos capas LSTM, y finalmente un contenedor (TimeDistributed) de una capa densa.

Arquitectura Recurrente				
RNN_33				
	Tipo	Salida	Activación	# Parámetros
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

CUADRO 5.5: Arquitectura recurrente para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia).

Es importante recalcar que las capas de redimensionamiento, agrupación, muestra ascendente y contenedores sólo fueron usadas para controlar la correcta compresión y descompresión de los autoencoders, es por ello que no se detalla a profundidad su funcionamiento.

### Evaluación de autoencoders

Anteriormente se presentó los mejores representantes por tipo de red; ahora se procederá a la evaluación y comparación de estos 3 tipos de autoencoders, con el objetivo de elegir la arquitectura que se ajusta mejor al comportamiento normal de conducción.

En el Capítulo 3 se presentó los diversos tipos de evaluación que existen, en esta etapa el tipo de evaluación más apropiado es la **precisión** del modelo, debido a que se tiene un gran conjunto de datos balanceado (debido a que sólo se cuenta con comportamientos normales de conducción que corresponden a una sola clase, la "Normal"). Los resultados de la evaluación de los tres tipos de redes son mostrados en el Cuadro 5.6; dicho cuadro presenta la precisión, pérdida logarítmica y tiempo de ejecución de cada autoencoder según el conjunto de prueba; observando estos resultados se puede apreciar que las dos mejores redes son la red densa **NN\_33** y la red recurrente **RNN\_33** con precisiones de 90 % aproximadamente, además de presentar un valor de pérdida relativamente bajo en comparación a la red **CNN\_33**.

<b>Red</b>	<b>Precisión</b>	<b>Loss</b>	<b>Tiempo ejecución</b>
NN_33	0.9000740711953905	0.003956934471097257	26us/step
CNN_33	0.843777761353387	0.006740666443275081	31us/step
RNN_33	0.8899259290695191	0.003611267575787173	101us/step

CUADRO 5.6: Evaluación de las redes NN\_33, CNN\_33 y RNN\_33 (Elaboración propia).

Por otra parte la diferencia más grande entre las dos mejores redes (**NN\_33** y **RNN\_33**) es el tiempo de ejecución ya que de la primera es de tan solo 26 segundos/paso y de la segunda es de 101 segundos/paso, debido a estas similitudes entre ambas redes es necesario verificar visualmente los resultados de reconstrucción de cada tipo de red, de tal manera que se pueda elegir la red más adecuada para este problema.

En la Figura 5.2 se muestra los resultados de los autoencoders de siete secuencias tomadas aleatoriamente del conjunto de prueba, en la parte superior de cada figura se encuentra la secuencia de entrada y en la inferior la reconstrucción del modelo, como ya se podía esperar la red **CNN\_33** presenta los peores resultados, lo cual hace que dicha red sea descartada; en cuanto a las dos redes restantes, la red **NN\_33** presenta reconstrucciones muy similares a las secuencias de entrada, con algunos pequeños errores; por otra parte **RNN\_33** presenta errores un poco más notorios que los obtenidos por **NN\_33**. Por lo tanto se llegó a la conclusión de que la red **NN\_33** se ajusta mejor al comportamiento normal de conducción, además de tener la gran ventaja de tener un tiempo de ejecución mucho menor que el de **RNN\_33**, lo cual es realmente importante para los sistemas en tiempo real así como también de aquellos que cuentan con recursos de ejecución limitados, como es el caso del presente trabajo.

Una vez definido como está constituido el modelo del comportamiento normal se puede proceder con la elección del método de detección de valores atípicos.

### 5.3.2. Método de detección de anomalías

Al inicio de esta sección se definió tres diferentes enfoques para la detección de anomalías: la umbralización, la aplicación de bosques de aislamiento y finalmente la aplicación de SVM para una clase.

#### Umbralización

Esta técnica se basa en la definición de un umbral para determinar si el error de reconstrucción que obtiene el autoencoder (modelo del comportamiento normal) es lo suficientemente alto como para considerarse un valor atípico. Por lo tanto primero se debe definir la ecuación del error de reconstrucción para el modelo. En el presente trabajo el error de reconstrucción se

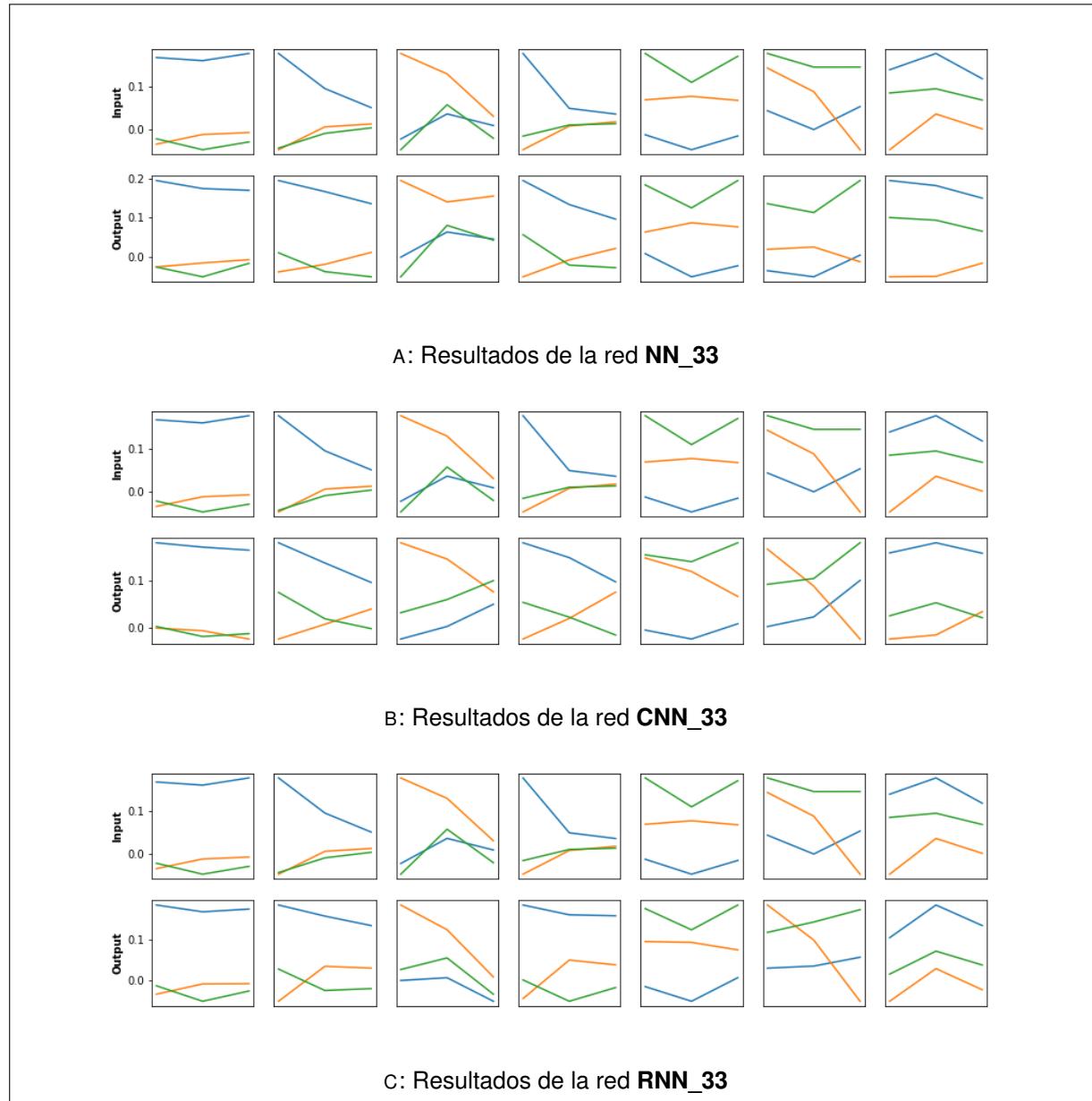


FIGURA 5.2: Resultados (Elaboración propia).

define según la ecuación 5.1, donde  $x_i$  representa el valor real (entrada del autoencoder) y  $\hat{x}_i$  representa el valor obtenido por el autoencoder (salida del autoencoder).

$$\text{Error de reconstrucción} = S_z = |x_i - \hat{x}_i|^2 \quad (5.1)$$

En la Figura 5.3 se muestra la curva de los errores de reconstrucción obtenidos con el modelo del comportamiento normal para el conjunto de muestras normales.

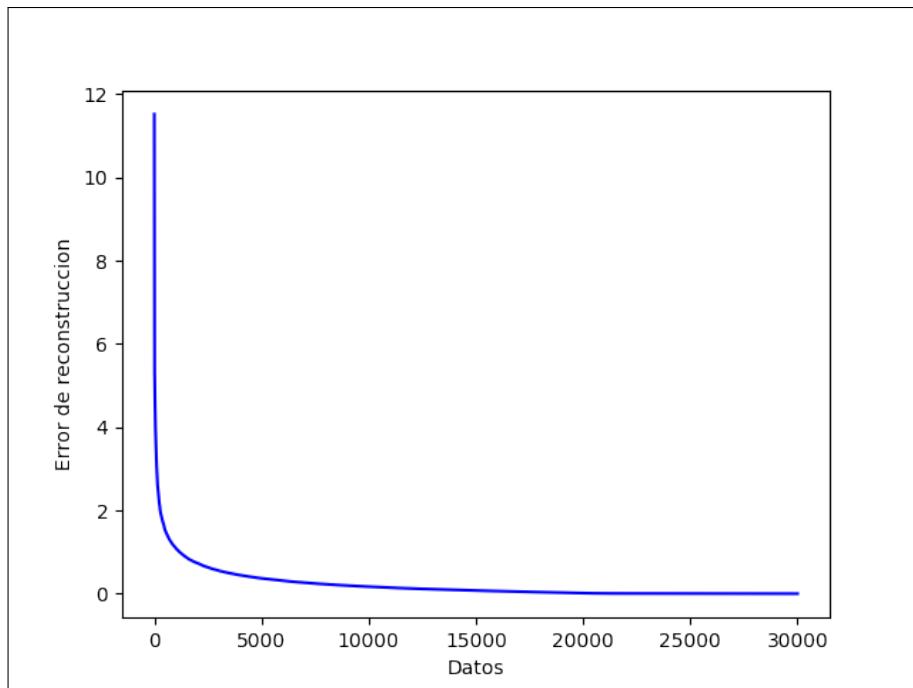


FIGURA 5.3: Curva de los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia).

Una vez definido la ecuación de reconstrucción, se debe definir un umbral capaz de poder detectar la mayor cantidad de anomalías posibles. Esta tarea puede tornarse simple en un entorno de aprendizaje supervisado, sin embargo automatizar esta tarea en un contexto de aprendizaje no supervisado es un desafío que puede ser difícil de sobreponer. En el presente trabajo se usó una técnica basada en encontrar un *Punto de codo* de una curva, que en este caso la curva está construida en base a los errores de reconstrucción del autoencoder.

Existen diferentes formas de hallar el punto de codo, sin embargo en este trabajo se utilizó una herramienta de Python, que automatiza esta tarea, llamada Kneedle. Esta herramienta devuelve el punto de inflexión de la función de la curva obtenida por el conjunto de valores proporcionado  $x$  y  $y$ , cabe recalcar que el punto de codo es el punto de máxima curvatura, por otra parte esta herramienta cuenta con un parámetro de sensibilidad ( $S$ ), este parámetro permite ajustar qué tan agresivo se desea ser al detectar codos, los valores más pequeños

para  $S$  detectan los codos más rápido, mientras que los más grandes son más conservadores, es decir,  $S$  es una medida de cuántos puntos "planos" se espera ver en la curva de datos sin modificar antes de declarar un codo.

De esta manera en el presente proyecto se realizó experimentos con diferentes valores de sensibilidad para encontrar el codo más adecuado para el conjunto de datos con el que se trabaja. En la Figura 5.4, se muestra los diferentes codos hallados para los valores de sensibilidad proporcionados (valores entre 0 y 2).

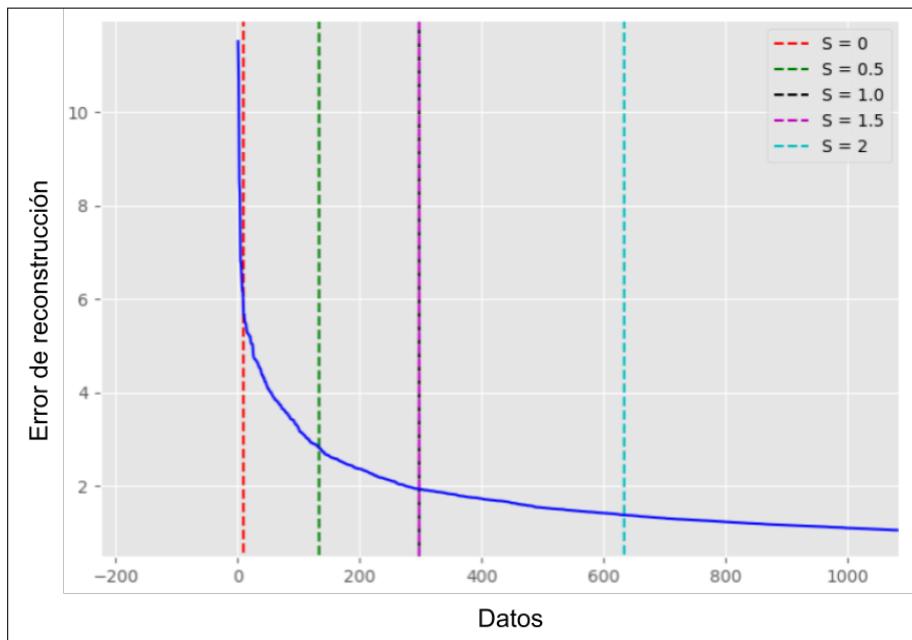


FIGURA 5.4: Resultados de la obtención de codos con diferentes valores de Sensibilidad, para los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia).

Una vez obtenidos los codos se realizó la evaluación de cada uno de ellos, en el Cuadro 5.7 se presentan el umbral, los valores de la matriz de confusión, la sensibilidad y especificidad para cada codo. Los valores de la matriz de confusión son el resultado de aplicar el umbral de cada codo a los errores de reconstrucción obtenidos del conjunto de datos total (conjunto de datos normal y anormal equivalente a 44204 datos).

S	Umbral	VP	VN	FN	FP	Sensibilidad	Especificidad
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

CUADRO 5.7: Evaluación de la detección de anomalías para cada codo obtenido con los diferentes valores de sensibilidad (Elaboración propia).

Según los resultados que se muestran en el Cuadro 5.7 se puede decir que mientras más pequeño es el umbral la sensibilidad (proporción de anomalías detectadas correctamente como anomalías) incrementa, sin embargo, a su vez reduce la especificidad (proporción de valores normales correctamente detectados como valores normales). Por lo tanto se debe hallar un punto intermedio, donde se pueda detectar la mayor cantidad de anomalías posibles y reducir en lo posible la cantidad de falsos positivos (datos normales que son detectados como anomalías). De esta forma el umbral más adecuado para el objetivo planteado fue 2.806, ya que con este umbral se detecta 111 anomalías de 164 y los falsos positivos son aproximadamente el doble de los valores atípicos detectados.

De ello se deduce que, para detectar automáticamente el umbral el uso de 0.5 como parámetro S es el más adecuado, sin embargo si se desea incrementar el porcentaje de detección de anomalías a costa de incrementar el número de falsos positivos se puede usar un valor mayor a 0.5 para S y en caso de querer la menor cantidad de falsos positivos posibles se debe usar un valor menor a 0.5.

### Isolation Forest

Antes de presentar como se llevará a cabo los experimentos con este algoritmo, es necesario ilustrar más detalladamente el funcionamiento del mismo. Por lo tanto la Figura 5.5 representa cómo se espera que un punto de datos anómalo se aísle rápidamente con el uso de este algoritmo, mientras que un punto de datos normal necesita más particiones para poder ser aislado.

Una vez detallado resumidamente el funcionamiento de los bosques de aislamiento se puede proseguir con los diferentes enfoques de los experimentos que se realizará con Isolation Forest.

Existen dos enfoques que pueden realizarse con esta técnica; el primero entrena el modelo con los valores compresos del codificador del autoencoder y el segundo se entrena con los errores de reconstrucción del autoencoder. A continuación se presenta una gráfica (Ver Figura 5.6) del autoencoder (modelo del comportamiento normal) con el fin de tener un mejor entendimiento de cómo se realizarán los experimentos tanto para los bosques de aislamiento como para los SVM de una clase.

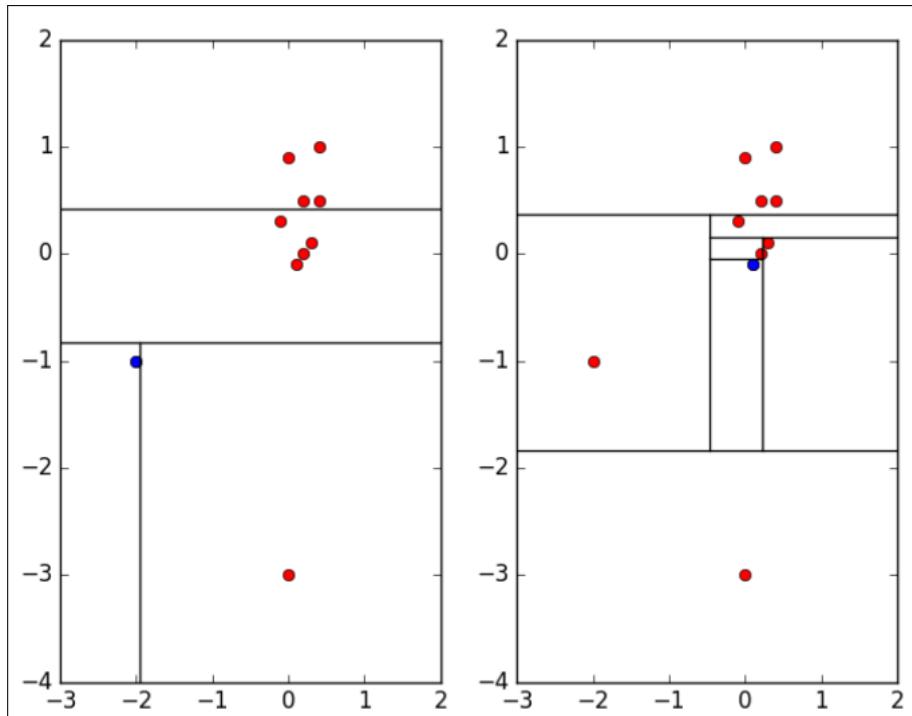


FIGURA 5.5: La figura de la izquierda muestra el aislamiento de una anomalía, que requiere solo tres particiones. A la derecha, el aislamiento de un punto normal requiere seis particiones (Wolpher, s.f.).

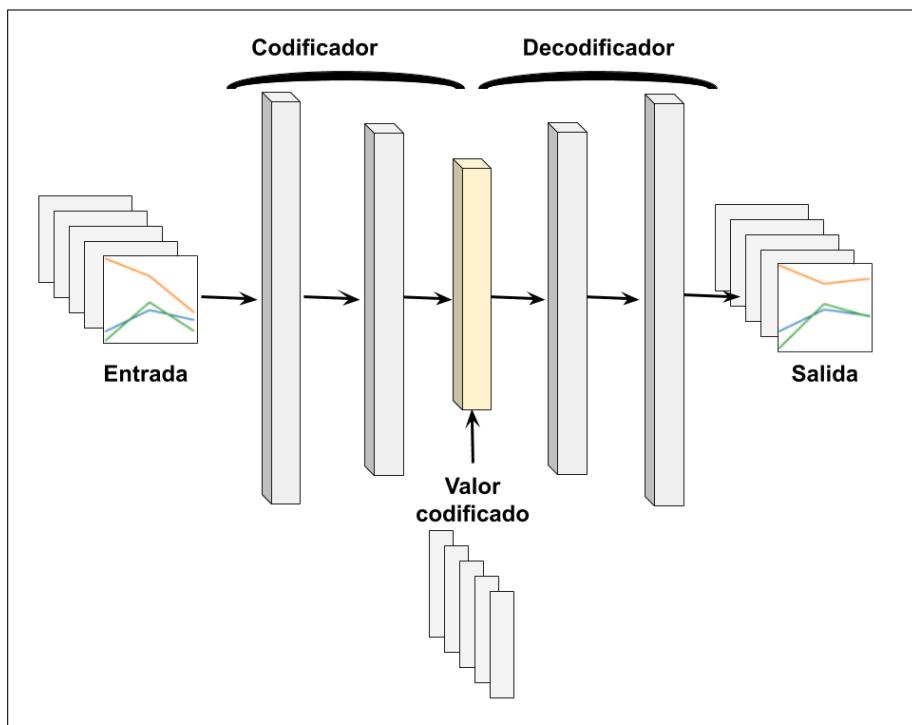


FIGURA 5.6: Representación gráfica del modelo de comportamiento normal o autoencoder (Elaboración propia).

- **Isolation forest para valores codificados:** Esta técnica entrena un modelo de bosque de aislamiento con los valores codificados (mediante el codificador del autoencoder, ver Figura 5.6) del conjunto de entrenamiento normal. Para los experimentos se utilizó la clase IsolationForest de SCIKIT-LEARN, esta clase tiene un parámetro llamado CONTAMINACIÓN el cual sirve para definir que cantidad del conjunto de datos esta contaminado, es decir, define que cantidad de los datos de entrenamiento pueden ser valores atípicos; en la presente investigación se realizó varias pruebas con diferentes valores para el parámetro contaminación. En el Cuadro 5.8 se presenta los resultados, donde se evidencia que ninguno de los resultados es alentador, ya que la cantidad de anomalías detectadas es muy baja para los tres casos con los que se experimento.

Contaminación (C)	VP	VN	FN	FP	Sensibilidad	Especificidad
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

CUADRO 5.8: Evaluación de la detección de anomalías usando Isolation forest para valores compresos (Elaboración propia).

- **Isolation forest para errores de reconstrucción:** Para esta técnica se realizó el entrenamiento del bosque de aislamiento con la diferencia de los valores de entrada con los valores obtenidos por el autoencoder (Ver Figura 5.7), cabe aclarar que la diferencia mencionada anteriormente también será llamada *Error de reconstrucción* tanto en esta como en la siguiente subsección. Los resultados de esta técnica para diferentes valores de contaminación se presentan en el Cuadro 5.9, donde estos resultados se pueden considerar como óptimos, debido a que oscilan entre 62 y 67 % de detecciones correctas de anomalías, además de presentar una especificidad realmente alta, del 99 % aproximadamente, lo cual quiere decir que estos modelos presentan una baja tasa de falsos positivos.

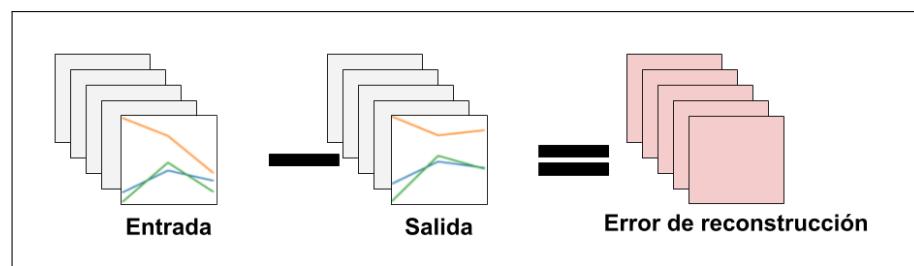


FIGURA 5.7: Representación gráfica del error de reconstrucción usado para el entrenamiento de los bosques de aislamiento y los SVM de una clase (Elaboración propia).

Contaminación (C)	VP	VN	FN	FP	Sensibilidad	Especificidad
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

CUADRO 5.9: Evaluación de la detección de anomalías usando Isolation forest para errores de reconstrucción (Elaboración propia).

## One-Class SVM

De la misma forma que en los bosques de aislamiento, se realizó dos diferentes tipos de experimentos con One-Class SVM, a continuación se detalla cada uno de ellos.

- **One-Class SVM para valores codificados:** Se debe entrenar un modelo SVM de una clase para los valores compresos obtenidos por el autoencoder; los experimentos fueron realizados usando la clase OneClassSVM de SCIKIT-LEARN, donde se tiene diferentes parámetros que pueden ser personalizados, para la presente investigación se probó diferentes kernels, obteniendo así los resultados que se muestran en el Cuadro 5.10, donde claramente ninguno de los resultados obtenidos podría ser tomado en cuenta para ser el método de detección de anomalías de conducción ya que la sensibilidad en ninguno de los casos es superior a 50 %.

Kernel	VP	VN	FN	FP	Sensibilidad	Especificidad
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

CUADRO 5.10: Evaluación de la detección de anomalías usando One-Class SVM para valores compresos (Elaboración propia).

- **One-Class SVM para los errores de reconstrucción:** Al igual que uno de los experimentos que se realizó con Isolation Forest, en esta técnica se usa los errores de reconstrucción (Ver Figura 5.7) para realizar el entrenamiento del modelo SVM de una clase. Como en los experimentos realizados en la anterior técnica se realizó diferentes pruebas con distintos tipos de kernel, a continuación en el Cuadro 5.11 se presenta los resultados obtenidos en los experimentos.

Kernel	VP	VN	FN	FP	Sensibilidad	Especificidad
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

CUADRO 5.11: Evaluación de la detección de anomalías usando One-Class SVM para el error de reconstrucción del autoencoder (Elaboración propia).

Observando los resultados del Cuadro 5.11 se puede notar que se aumentó notablemente la sensibilidad, o cantidad de anomalías detectadas correctamente, sin embargo, redujo drásticamente la especificidad ya que en algunos casos tan solo llega a un 3.5 %, lo cual es muy alejado al objetivo que se persigue en el presente trabajo.

### Evaluación del método de detección de anomalías

Una vez realizado los diferentes tipos de experimentos, se evaluó el mejor exponente de cada tipo, con el fin de elegir el más adecuado para la investigación. A continuación se presenta un Cuadro 5.12 con los resultados de los mejores representantes por cada tipo de técnica.

<b>Nombre método</b>	<b>VP</b>	<b>VN</b>	<b>FN</b>	<b>FP</b>	<b>Sensibilidad</b>	<b>Especificidad</b>
Umbralización con S=0.5	111	43814	53	226	0.6768	0.9949
Isolation Forest para errores de reconstrucción con C=0.0075	111	43688	53	352	0.6768	0.9920
One-Class SVM para errores de reconstrucción con kernel RBF	134	41887	30	2153	0.8170	0.9511

CUADRO 5.12: Comparación de los mejores métodos de detección de anomalías  
(Elaboración propia).

Evidentemente el mejor resultado de detección de anomalías es el que se obtuvo por el modelo SVM para una clase con una sensibilidad del 81.7 %, sin embargo, este método presenta la desventaja de tener una alta cantidad de falsos positivos, es decir, por cada anomalía detectada se tendrá aproximadamente 13 alertas por falsos positivos, lo cual es una valor muy alto; y es la principal razón por la que se descarta este método.

Debido a esto sólo quedan dos métodos a comparar, donde ambos resultados son muy similares; ya que estos cuentan con una sensibilidad de 67.68 %, por otra parte la especificidad tiene una pequeña variación entre ambas técnicas dando un resultado levemente mejor para la técnica de umbralización con 99.49 % frente a un 99.20 %.

En este punto se puede elegir cualquiera de estos dos métodos debido a las similitudes que ambos presentan. Por razones de simplificación en este estudio se eligió el método de bosque de aislamiento ya que este método hace más sencilla la detección de anomalías debido a que uno puede especificar la cantidad de contaminación que se espera del conjunto de datos, esto es mucho más ventajoso que la búsqueda de codos con el método de umbralización ya que presenta la desventaja de que es realmente complejo definir el umbral cuando se trata esta técnica en un enfoque no supervisado, como es el caso de esta etapa, además que la

definición del umbral depende mucho de cuán limpio o contaminado se encuentra el conjunto de datos, haciendo más complejo el correcto tratamiento al aplicar este método.

Una vez elegido los mejores métodos para conformar el mecanismo de detección de valores atípicos, se procede a formalizar este mecanismo por medio de una gráfica (Ver Figura 5.8), la cual proporciona una representación visual del flujo del detector de anomalías propuesto; ya que es importante conocer como se compone y como funciona, especialmente antes de realizar su respectiva evaluación.

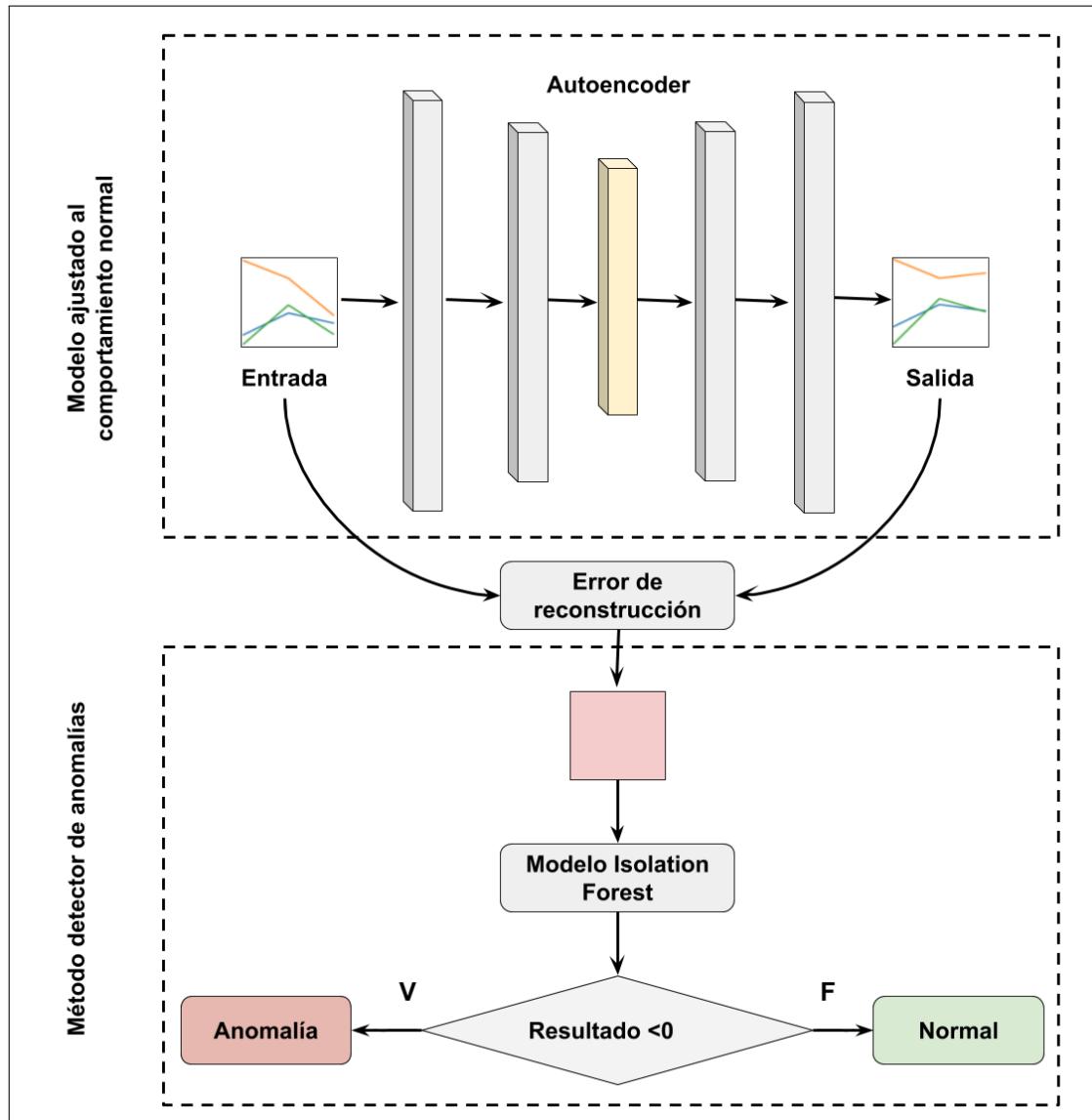


FIGURA 5.8: Mecanismo de detección de anomalías (Elaboración propia).

Observando la Figura 5.8, se puede notar claramente los componentes que conforman el mecanismo de detección de anomalías: el **modelo del comportamiento normal** y el **método detector de anomalías**. Este mecanismo funciona de una forma muy sencilla, en primer lugar se proporciona al autoencoder una secuencia de entrada con 3 pasos para 3 componentes principales (cabe aclarar que los datos de entrada han sido previamente pre-procesados), este autoencoder devuelve como salida la reconstrucción de la entrada, con la cual se obtiene el error de reconstrucción (diferencia entre la entrada real y el valor reconstruido), dicho valor es a su vez la entrada del modelo de bosque de aislamiento, el cual puede retornar dos tipos de valores (-1, 1); cuando este modelo retorna el valor 1 quiere decir que la entrada proporcionada corresponde a un valor considerado como normal y en caso de retornar -1 significa que dicha entrada es una anomalía, terminando así el flujo del mecanismo de detección propuesto en el presente trabajo.



## Capítulo 6

# RESULTADOS Y EVALUACIÓN

El propósito de este capítulo es presentar los resultados de la evaluación del mecanismo de detección de anomalías propuesto, para posteriormente mostrar algunos de los resultados obtenidos con el mismo.

### 6.1. Evaluación de desempeño

En este estudio se evaluará la efectividad de las técnicas de detección de anomalías desde las dos siguientes perspectivas:

- La capacidad del enfoque para distinguir entre datos normales y anómalos.
- La eficiencia del método de acuerdo con el tiempo requerido para entrenar el modelo y el tiempo empleado durante el proceso de detección.

#### 6.1.1. Evaluación en términos de rendimiento de detección

Antes de evaluar el mecanismo propuesto en este estudio es importante destacar qué:

- El **modelo de comportamiento normal** fue entrenado con 21000 muestras, durante 50 iteraciones, con 4500 muestras que se usaron para validar el modelo durante la etapa de entrenamiento, y por último el conjunto de prueba con el que se realizó la evaluación final de este modelo esta conformado por 4500 muestras.
- Por otra parte el **método detector de anomalías** fue entrenado con la totalidad de los datos que se usaron en el desarrollo de la generación del modelo del comportamiento normal, es decir, con 30000 muestras.

Para evaluar el mecanismo de detección de anomalías propuesto en el estudio, se utilizó los siguientes criterios: la tasa de detección y la tasa de falsos positivos. La tasa de detección se define como el número de anomalías detectadas dividido por el número total de anomalías. La tasa de falsos positivos se define como el número de series "normales" que se clasifican como anomalías divididos por el número total de series "normales". Es importante aclarar que

el conjunto de valores atípicos, con el que se cuenta en esta investigación, no fue usado para el entrenamiento del método propuesto; sin embargo este conjunto sí se usó para validar su precisión, por lo tanto el conjunto de datos con el que se valida este mecanismo cuenta con 44204 datos.

En la Tabla 6.1 se presenta la matriz de confusión obtenida por el mecanismo propuesto, de donde se pueden obtener las siguientes afirmaciones:

- La entrada superior izquierda de la matriz muestra que 111 anomalías de 164 fueron correctamente etiquetadas, es decir, que el 67.68 % de las muestras de anomalías se reconocieron correctamente.
- En la fila inferior se muestra que 43688 de 44040 datos fueron etiquetadas correctamente como valores normales, es decir, el 99.20 %. Por lo tanto la tasa de falsos positivos para la clase normal es  $100 - 99,20\% = 0,80\%$ .

		Predicción	
		Anomalía	Clase Normal
Reales	Anomalía	111	53
	Clase Normal	352	43688

CUADRO 6.1: Matriz de confusión, para el mecanismo de detección de anomalías  
(Elaboración propia).

Estos resultados son un gran avance para la detección de anomalías de conducción con un enfoque semi supervisado, ya que al no contar con muestras de valores atípicos en el entrenamiento es difícil tener una precisión más alta; considerando además, que uno de los valores agregados más importantes que presenta este trabajo de investigación, es el poder generar un modelo personalizado por cada tipo de agente, lo cual es realmente sobresaliente, debido a que el trabajo relacionado que se revisó, previamente a la elaboración de esta investigación, no cuenta con un ejemplar que contemple un enfoque semi-supervisado y mucho menos con modelos que se ajusten y personalicen para cada agente.

## 6.2. Resultados

Los resultados de este estudio proporcionan una contribución esencial en el campo de la automatización de detección temprana de conductas anómalas en la conducción de automóviles; sin embargo, éstos presentan una visión general del comportamiento del modelo propuesto, por lo cual es necesario realizar un análisis más específico de dicho comportamiento con cada tipo de anomalía presentada en el conjunto de datos, así como también el análisis sobre aquellos valores que fueron detectados erróneamente como anomalías (falsos positivos). A continuación se presentan los resultados de los análisis previamente mencionados.

### 6.2.1. Detección de anomalías del tipo zig zag

Esta anomalía corresponde a un comportamiento común que suelen realizar agentes que conducen bajo los efectos del alcohol; consiste en una conducción que presenta movimientos en zig zag de forma brusca, es decir, cambios de dirección constante y a una velocidad relativamente alta. A continuación se presentan algunos de los resultados que se obtuvo con el mecanismo de detección de anomalías propuesto con este trabajo de investigación.

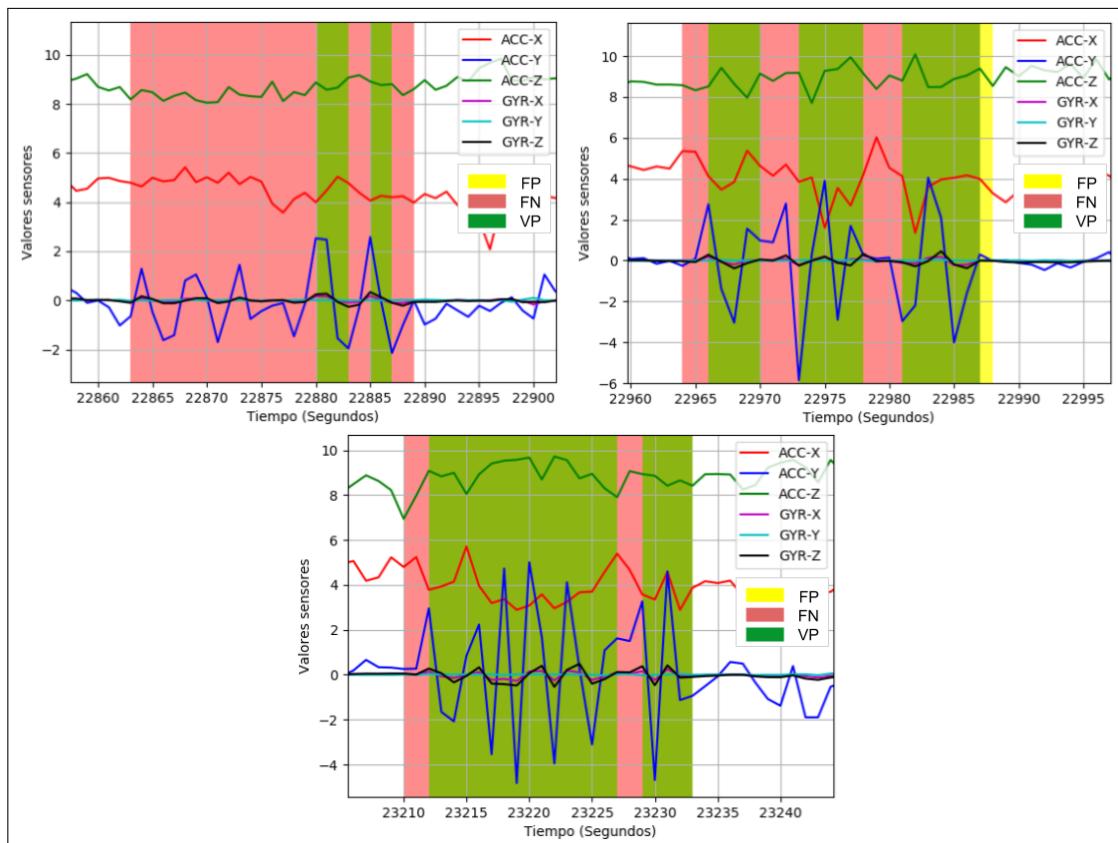


FIGURA 6.1: Resultados de la detección de anomalías del tipo zig zag (Elaboración propia).

Antes de realizar el análisis de los resultados obtenidos se debe aclarar que aquellas secciones de las siguientes gráficas que se presentan en color rojo son los valores que pertenecen al conjunto de anomalías que no fueron correctamente detectados (Falsos negativos), las secciones en amarillo corresponden a los falsos positivos y por último las secciones verdes son los verdaderos positivos, es decir, aquellos valores que fueron detectados correctamente como anomalías.

Como se observa en la Gráfica 6.1, la imagen superior izquierda presenta una gran cantidad de falsos negativos, esto se debe a que las oscilaciones de los movimientos en Zig Zag no fueron

lo suficientemente bruscos, en comparación a los demás, por otro lado la imagen superior derecha presenta una cantidad moderada de falsos negativos y un ejemplar de falso positivo, aunque el resultado no parezca del todo bueno realmente si lo es, ya que muchos de los falsos negativos se encuentran entre valores detectados correctamente, lo cual conllevaría a una correcta generación de alarma de anomalías a pesar de no detectar como valor atípico la totalidad de los datos anómalos, en la imagen inferior se presenta un ejemplo similar, aunque en este caso no se detectan falsos positivos.

Con el fin de formalizar los resultados para las anomalías del tipo Zig Zag, en la tabla 6.2 se puede observar que 69 anomalías de 105 fueron correctamente detectadas, es decir el 65.71 %.

Giros en Zig Zag		
VP	FN	Total
69	36	105

CUADRO 6.2: Resultados anomalías tipo Zig Zag (Elaboración propia).

### 6.2.2. Detección de anomalías del tipo giros a alta velocidad

Este tipo de anomalías suelen ser comunes en agentes que conducen bajo los efectos del alcohol, drogas o con un estado emocional alterado, dichos datos se consideran anomalías ya que los giros normalmente se realizan bajando la velocidad del vehículo, y al realizar este tipo de actos un agente es propenso a ser el causante de un accidente de tránsito.

La Figura 6.2 muestra los resultados obtenidos para las anomalías del tipo giros a alta velocidad, las tres imágenes presentan resultados muy similares, todas tienen una sección en la parte inicial que se presenta como falso negativo, es decir tienen una proporción de datos que no son detectadas correctamente, posteriormente cuentan con un bloque de verdaderos positivos, y por último, dos de las tres imágenes cuentan con un ejemplar de falso positivo posterior a la anomalía. A pesar de que este tipo de anomalías no son detectadas completamente, todas presentan una sección que sí es detectado como anomalía, lo cual es suficiente para generar una alarma oportunamente.

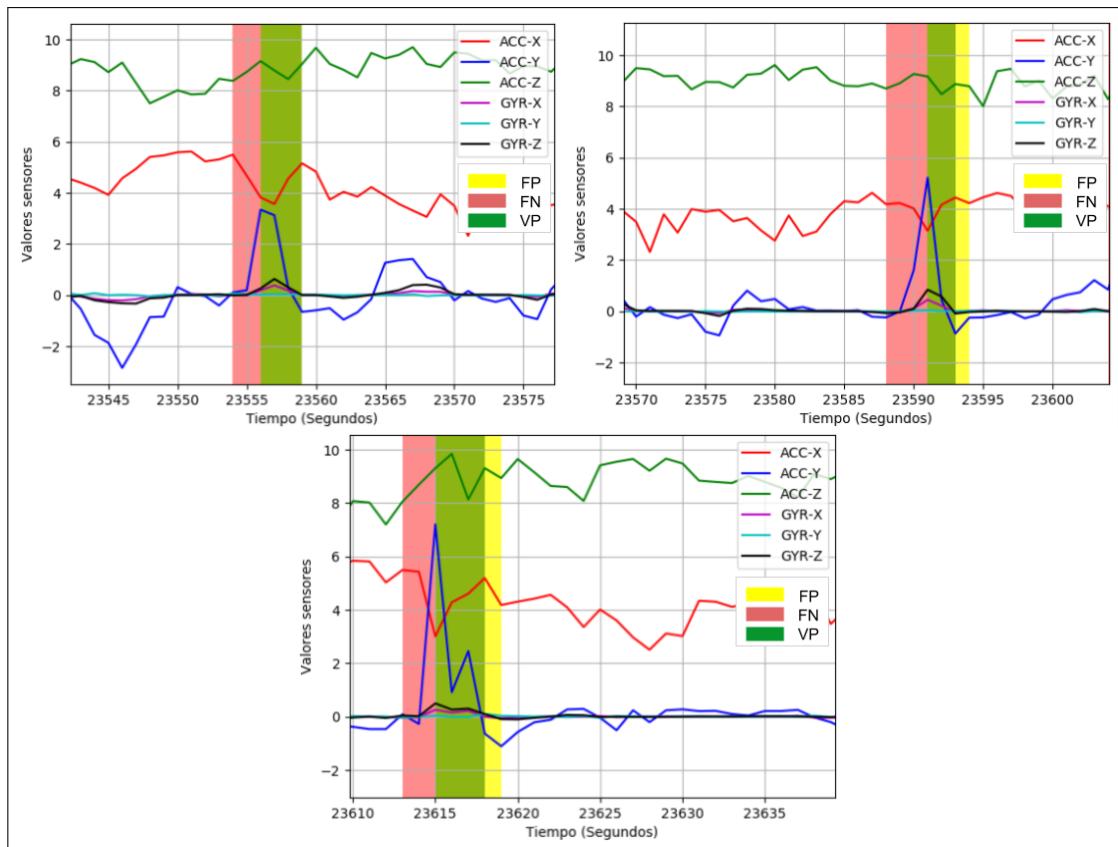


FIGURA 6.2: Resultados de la detección de anomalías del tipo giros a alta velocidad (Elaboración propia).

En el Cuadro 6.3 se presenta los resultados generales obtenidos para las anomalías del tipo Giros a alta velocidad, donde de 35 anomalías 23 fueron correctamente detectadas , es decir, el 65.71 %.

Giros a alta velocidad		
VP	FN	Total
23	12	35

CUADRO 6.3: Resultados del tipo Giros a alta Velocidad (Elaboración propia).

### 6.2.3. Detección de anomalías del tipo frenos en seco

Este tipo de anomalía suele ser uno de los valores atípicos más comunes que existen, ya que no sólo se presentan bajo los efectos del alcohol, drogas o fallas mecánicas, sino que también se presentan en contextos de distracción del conductor ya sea por el uso del celular u otro tipo de distracción, ante la aparición de un peatón o mascota que se presenta de manera repentina en la carril que conduce el agente, entre otros casos.

Los resultados de la detección de este tipo de anomalía se presentan en la Figura 6.3, donde al igual que el caso anterior este tipo de anomalía presenta una sección de falsos negativos, posteriormente un grupo de anomalías correctamente detectadas y finalmente falsos positivos; con lo cual es suficiente para generar alertas de manera oportuna y de esa manera poder evitar en lo posible algún accidente de tránsito.

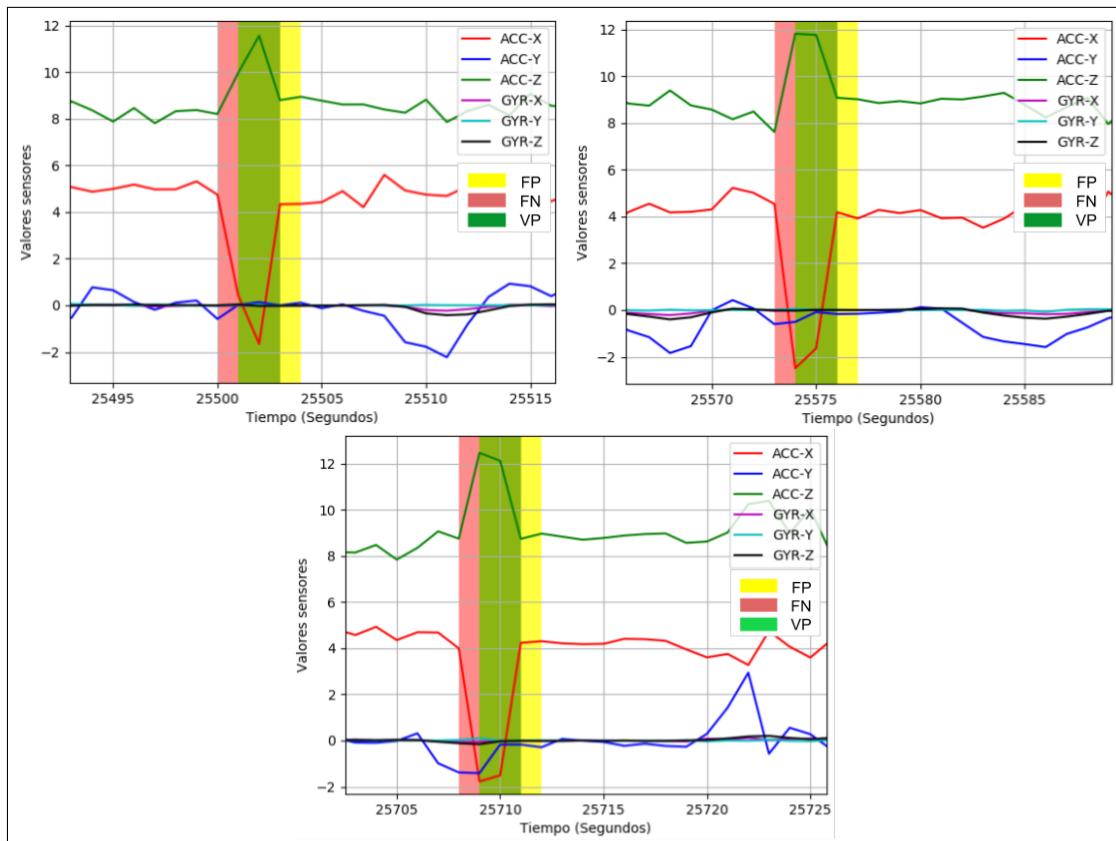


FIGURA 6.3: Resultados de la detección de anomalías del tipo frenos en seco  
(Elaboración propia).

A continuación en el Cuadro 6.4 se puede ver que 19 anomalías de 24 fueron correctamente detectadas (79.17 %), siendo así el tipo de anomalía que tiene el porcentaje de detección más elevado.

Frenos en seco		
VP	FN	Total
19	5	24

CUADRO 6.4: Resultados del tipo Frenos en seco (Elaboración propia).

### 6.2.4. Detección de falsos positivos

Así como se detectó una gran cantidad de anomalías mediante este mecanismo, también se detectó una proporción considerable de falsos positivos, es decir, valores normales que fueron detectados erróneamente como valores atípicos.

De la misma forma que es importante conocer como este método detecta anomalías, también es importante saber en que casos el modelo propuesto falla; en la Figura 6.4 se presenta algunos casos donde el modelo falla, es decir, esta figura presenta algunos ejemplos de falsos positivos. La figura 6.4 ilustra claramente que estos falsos positivos se presentan generalmente de forma aislada, es decir, uno o dos valores detectados erróneamente como anomalías de forma continua, lo cual es un comportamiento diferente al de los verdaderos valores atípicos, ya que estos presentan una detección de tres valores atípicos de forma continua mínimamente.

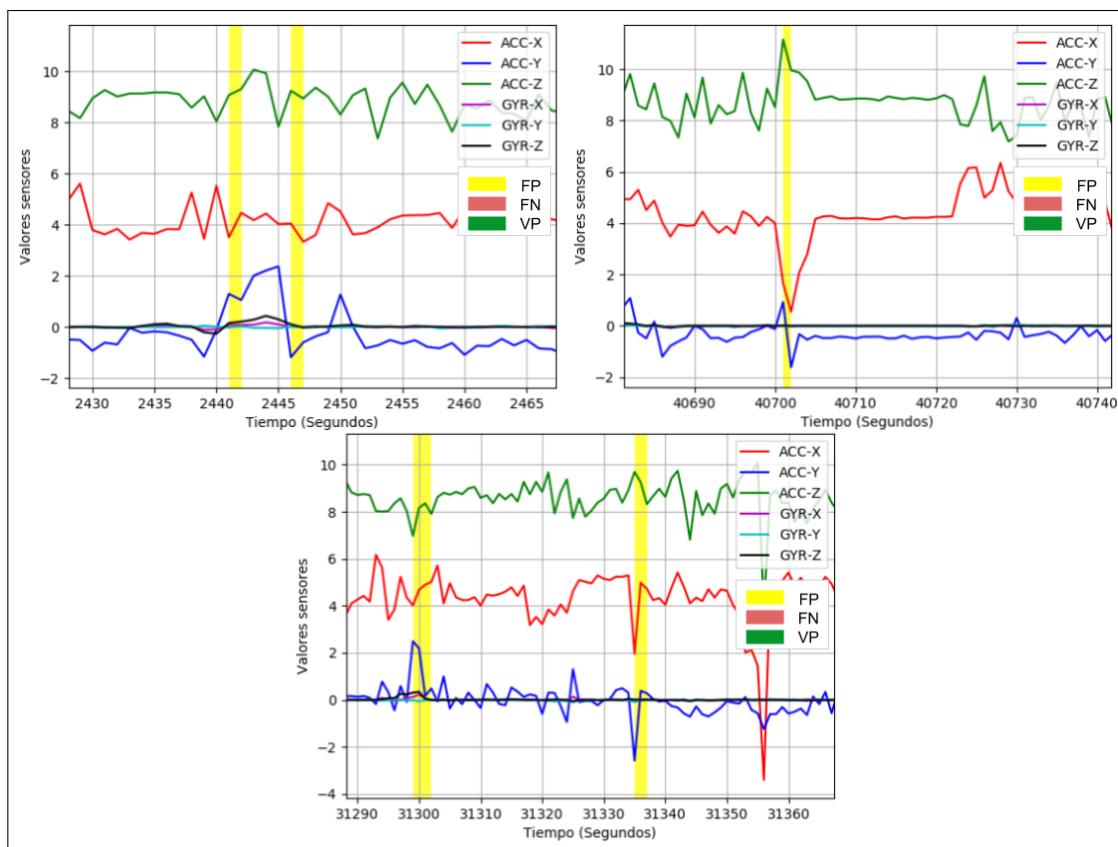


FIGURA 6.4: Resultados de la detección de falsos positivos (Elaboración propia).



## Capítulo 7

# CONCLUSIONES Y TRABAJOS FUTUROS

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 iniciales 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  $t - 1$  ( $diff_t = dato_t - dato_{t-1}$ ), 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

## Referencias

- 0.22, S. (s.f.). *Novelty and outlier detection*. [https://scikit-learn.org/stable/modules/outlier\\_detection.html](https://scikit-learn.org/stable/modules/outlier_detection.html). (Último acceso en 19 de diciembre de 2019)
- Alashwal, H., Bin D., S., y 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., y Araujo, R. (2012, 6). Driving coach: A smartphone application to evaluate driving efficient patterns. *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV); Alcalá de Henares, España*(3–7), 1005–1010. Descargado de [https://www.researchgate.net/publication/261309792\\_Driving\\_coach\\_A\\_smartphone\\_application\\_to\\_evaluate\\_driving\\_efficient\\_patterns](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., y Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw*, 5, 157–166. Descargado de <https://ieeexplore.ieee.org/document/279181/authors#authors>
- Bhoyar, V., Lata, P., Katkar, J., Patil, A., y Javale, D. (2013, 3–4). Symbian based rash driving detection system. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 2, 124–126. Descargado de <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., y 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. Descargado de <https://ieeexplore.ieee.org/abstract/document/5137037>
- Chen, Z., Yu, J., Zhu, Y., Chen, Y., 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. Descargado de <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., y Bengio, Y. (2014a). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv*, 1406.1078. Descargado de <https://www.aclweb.org/anthology/>

- D14-1179.pdf
- Dang-Nhac, L., Duc-Nhan, N., Thi-Hau, N., y Ha-Nam, N. (2018, 4). Vehicle mode and driving activity detection based on analyzing sensor data of smartphones. *Sensors*, 18(4), 1036. Descargado de <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5948751/>
- Dauphin, Y. N., Fan, A., Auli, M., y Grangiera, D. (2017, 9). Language modeling with gated convolutional networks. *arXiv*, 1612.08083v3(8). Descargado de <https://arxiv.org/pdf/1612.08083.pdf>
- de Salud (OMS), O. M. (s.f.). *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](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. Descargado de <https://crl.ucsd.edu/~elman/Papers/fsit.pdf>
- Eren, H., Makinist, S., Akin, E., y 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., y Pessin, G. (2017). Driver behavior profiling: An investigation with different smartphone sensors and machine learning. *PLoS One*, 12(4), e0174959. Descargado de <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5386255/>
- Galarnyk, M. (s.f.). *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., y Li, P. (2018). Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. *Proceedings of Machine Learning Research*, 95, 97–112. Descargado de <http://proceedings.mlr.press/v95/guo18a/guo18a.pdf>
- Hawkins, S., He, H., Williams, G., y Baxter, R. (2002, 9). Outlier detection using replicator neural networks. *International Conference on Data Warehousing and Knowledge Discovery*, 170–180. Descargado de <https://togaware.com/papers/dawak02.pdf>
- Hochreiter, S., y Schmidhuber, J. (1997, 11). Long short-term memory. *Neural Comput*, 9(8)(15), 1735—1780. Descargado de <https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735>
- Hsu, A., y Griffiths, T. (2010). Effects of generative and discriminative learning on use of category variability. Descargado de <https://cocosci.princeton.edu/tom/papers/discgencat.pdf>
- Jayesh, B. (s.f.). *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., y Guanci, Y. (2018, 03). Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. *Algorithms*, 11, 28.
- Johnson, D., y 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. Descargado de [http://cvrr.ucsd.edu/publications/2011/Johnson\\_ITSC2011.pdf](http://cvrr.ucsd.edu/publications/2011/Johnson_ITSC2011.pdf)
- Klos, M., y Waszczyszyn, Z. (2011). Modal analysis and modified cascade neural networks in

- identification of geometrical parameters of circular arches. *Computers & Structures*, 89, 581–589. Descargado de <http://cames.ippt.gov.pl/index.php/cames/article/view/110>
- Koh, D. W., y 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. Descargado de <https://ieeexplore.ieee.org/abstract/document/7225655>
- Kridalukmana, R., Yan-Lu, H., y Naderpour, M. (2017). An object oriented bayesian network approach for unsafe driving maneuvers prevention system. *12th International IEEE Conference*. Descargado de <https://opus.lib.uts.edu.au/bitstream/10453/122196/4/633634.pdf>
- Lecun, Y., Bengio, Y., y Hinton, G. (2015, 5). Deep learning. *Nature*, 521(7553)(27), 436—444. Descargado de <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. Descargado de <http://yann.lecun.com/exdb/publis/pdf/lecun-89c.pdf>
- Mass, A., Hannun, A., y Ng, A. (2013). Rectifier nonlinearities improve neural network acoustic models. *International Conference on Machine Learning (icml)*. Descargado de [https://ai.stanford.edu/~amaas/papers/relu\\_hybrid\\_icml2013\\_final.pdf](https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf)
- Michael, A. (2015). *Neural networks and deep learning*. Determination Press. Descargado de <http://neuralnetworksanddeeplearning.com/index.html>
- Moindrot, O., y Genthal, 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. (s.f.). *Convolutional neural network. in a nut shell*. <https://engmrk.com/convolutional-neural-network-3/>. (Último acceso en 19 de diciembre de 2019)
- Olah, C. (s.f.). *Understanding lstm networks*. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. (Último acceso en 11 de octubre de 2019)
- Özler, H. (s.f.). *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., y 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. Descargado de <http://proceedings.mlr.press/v28/pascanu13.pdf>
- Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann Publishers, Inc. Descargado de <https://pdfs.semanticscholar.org/470a/828d5e3962f2917a0092cc6ba46ccfe41a2a.pdf>
- Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. Descargado de <https://psycnet.apa.org/record/1987-33645-001>
- Russakovsky, O. e. a. (2014). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*. Descargado de <http://link.springer.com/article/10.1007/s11263-015-0816-y#>
- Schölkopf, B., y Smola, A. J. (2002). *Support vector machines, regularization, optimization, and*

- beyond.* MIT Press.
- Shai, S., y Shai, B. (2014). *Understanding machine learning: From theory to algorithms.* Cambridge University Press. Descargado de <https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf>
- Smits, P., Dellepiane, S., y 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., y Wesam, S. (2017). Review of data preprocessing techniques in data mining. *Review of Scientific Instruments*, 12(16), 4102–4107. Descargado de <http://docsdrive.com/pdfs/medwelljournals/jeasci/2017/4102-4107.pdf>
- Varun, C., y Arindam, K., B.and Vipin. (2009). *Anomaly detection: A survey.* Universidad de Minnesota. Descargado de [https://www.researchgate.net/publication/220565847\\_Anomaly\\_Detection\\_A\\_Survey](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. Descargado de <https://www.sciencedirect.com/science/article/abs/pii/089360808890007X>
- Who-Lee, K., Sik-Yoon, H., Min-Song, J., y 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. Descargado de <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5948584/>
- Wikipedia. (s.f.). *Neuron.* <https://simple.wikipedia.org/wiki/Neuron>. (Último acceso en 19 de diciembre de 2019)
- Williams, G., y Baxter, R. (2002, 12). A comparative study of rnn for outlier detection in data mining. *IEEE International Conference on Data Mining*, 1–16. Descargado de <https://towardsai.net/papers/tr02102.pdf>
- Wolpher, M. (s.f.). *Anomaly detection in unstructured time series data using an lstm auto-encoder.* <http://www.diva-portal.org/smash/get/diva2:1225367/FULLTEXT01.pdf>. (Último acceso en 11 de octubre de 2019)
- Xue, Z., Shang, Y., 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. Descargado de [https://www.researchgate.net/publication/220348246\\_Semi-supervised\\_outlier\\_detection\\_based\\_on\\_fuzzy\\_rough\\_C-means\\_clustering](https://www.researchgate.net/publication/220348246_Semi-supervised_outlier_detection_based_on_fuzzy_rough_C-means_clustering)
- Yan, S. (s.f.). *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., y 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., y Hinton, G. E. (2013). On rectified linear units for speech processing. *International Conference on Acoustics, Speech and Signal Processing. IEEE*, 3517—3521. Descargado de <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. Descargado de <http://cames.ippt.gov.pl/index.php/cames/article/view/110>
- Zhang, A., Lipton, Z., Li, M., y Smola, A. (2019, 9). *Dive into deep learning*. <https://en.d2l.ai/d2l-en.pdf>. (Último acceso en 11 de octubre de 2019)





## Apéndice 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
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

CUADRO 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

CUADRO 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
	CNN_33	Conv1D	(3,2)	elu
	CNN_33	MaxPooling1D	(2,2)	0
	CNN_33	Conv1D	(2,4)	elu
	CNN_33	MaxPooling1D	(1,4)	0
	CNN_33	Conv1D	(1,6)	elu
	CNN_33	UpSampling1D	(3,6)	0
	CNN_33	Conv1D	(3,3)	tanh
Redes Conv - 3 componentes principales	CNN_43	Input	(4,3)	0
	CNN_43	Conv1D	(4,2)	elu
	CNN_43	MaxPooling1D	(2,2)	0
	CNN_43	Conv1D	(2,4)	elu
	CNN_43	MaxPooling1D	(1,4)	0
	CNN_43	Conv1D	(1,6)	elu
	CNN_43	UpSampling1D	(4,6)	0
	CNN_43	Conv1D	(4,3)	tanh
Redes Conv - 3 componentes principales	CNN_53	Input	(5,3)	0
	CNN_53	Conv1D	(5,2)	elu
	CNN_53	MaxPooling1D	(3,2)	0
	CNN_53	Conv1D	(3,4)	elu
	CNN_53	MaxPooling1D	(2,4)	0
	CNN_53	Conv1D	(2,5)	elu
	CNN_53	Reshape	(5,2)	0
	CNN_53	Conv1D	(5,3)	tanh

CUADRO 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

CUADRO 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	
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	
RNN_53	Reshape	(4,3)		0	
	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	
Redes Rec - 3 componentes principales					
TimeDistributed(Dense(3))					
(3,5)					
Reshape					

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

**A.3.2. Evaluación redes recurrentes**

<b>Red</b>	<b>val_loss</b>	<b>val_acc</b>	<b>loss</b>	<b>acc</b>
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

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



## Apéndice B

# Arquitectura del Sistema de Demostración

Si bien el trabajo presentado no promete un sistema que implemente el mecanismo 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

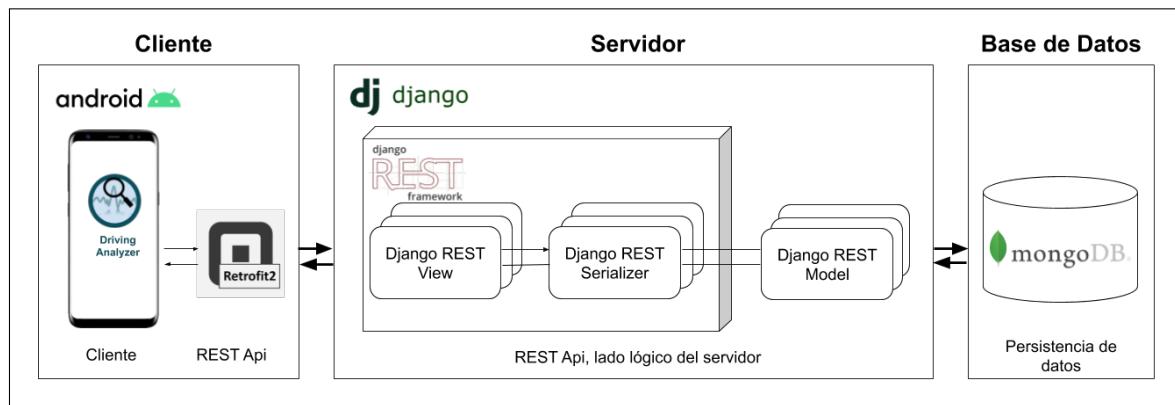


FIGURA B.1: Arquitectura física del Sistema de Demostración (Elaboración propia).

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 iniciales, 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 si 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

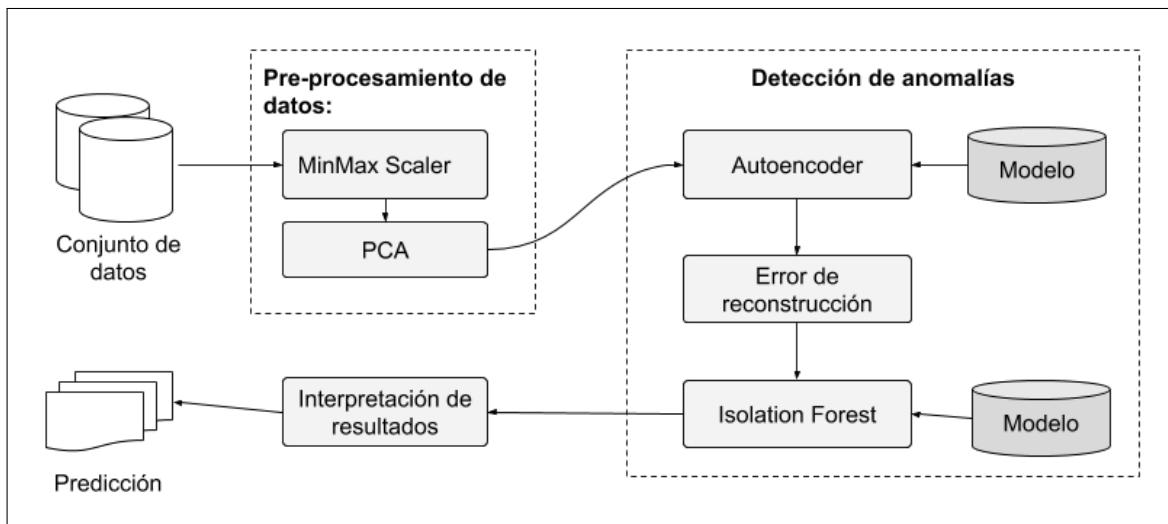


FIGURA B.2: Arquitectura Lógica del Sistema de Demostración (Elaboración propia).

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.