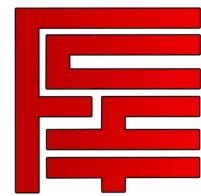




Universidad Mayor de San Simón
Facultad de Ciencias y Tecnología
Carrera de Ingeniería de Sistemas



Detección de Anomalías de Conducción

Proyecto de grado presentado en cumplimiento de los requisitos
para optar por el título de Licenciatura en Ingeniería de Sistemas

Presentado por:

Evelyn CUSI LÓPEZ

Tutor:

Ph.D. Eduardo DI SANTI

COCHABAMBA - BOLIVIA

Marzo, 2020

Agradecimientos

A mi familia, por su comprensión y estímulo constante, además de su apoyo incondicional a lo largo de mis estudios.

A mi asesor y mentor: Ph.D. Eduardo Di Santi, quien me brindó su valiosa y desinteresada orientación y guía en la elaboración del presente trabajo, por su paciencia, apoyo e ideas que motivaron esta investigación; a quien le debo gran parte de mi aprendizaje y mi gusto por el área de la Inteligencia Artificial.

A mis amigos, que de una y otra forma me apoyaron en la realización de este trabajo.

A la Facultad de Ciencias y Tecnología y la Universidad Mayor de San Simón, por abrirme las puertas y ser la casa de mi formación durante toda mi carrera universitaria.

*A mi madre, por estar conmigo, por enseñarme a crecer y a levantarme,
por apoyarme y guiarme, por ser la base que me ayudó a llegar hasta
aquí.*

Índice general

| | |
|---|------------|
| Agradecimientos | III |
| Resumen | xv |
| 1. INTRODUCCIÓN | 1 |
| 1.1. Planteamiento del problema | 1 |
| 1.2. Objetivo general | 2 |
| 1.3. Objetivos específicos | 3 |
| 1.4. Justificación | 3 |
| 1.4.1. Justificación práctica | 4 |
| 1.4.2. Justificación metodológica | 4 |
| 1.5. Límites y alcances | 4 |
| 1.6. Método de investigación | 4 |
| 2. FUNDAMENTOS DE LA DETECCIÓN DE ANOMALÍAS | 5 |
| 2.1. Detección de anomalías | 5 |
| 2.2. Desafíos en la detección de anomalías | 7 |
| 2.2.1. Enfoques de detección de anomalías | 8 |
| Detección de anomalías supervisada | 8 |
| Detección de anomalías semi-supervisada | 8 |
| Detección de anomalías no supervisada | 8 |
| 2.3. Trabajo relacionado | 8 |
| 2.4. Enfoque sobre el problema | 10 |
| 3. APRENDIZAJE AUTOMÁTICO PARA LA DETECCIÓN DE ANOMALÍAS | 13 |
| 3.1. Aprendizaje Supervisado, Aprendizaje no Supervisado y Aprendizaje Semi supervisado | 13 |
| 3.1.1. Aprendizaje Supervisado | 14 |
| 3.1.2. Aprendizaje no Supervisado | 14 |
| 3.1.3. Aprendizaje Semi-Supervisado | 14 |
| 3.2. Modelos Generativos y Discriminativos | 15 |
| 3.3. Redes Neuronales Artificiales | 16 |
| 3.3.1. Neuronas o nodos | 16 |
| 3.3.2. Tipos de funciones de activación | 18 |
| Función de activación sigmoide (función logística) | 18 |
| Función de tangente hiperbólica - tanh | 20 |
| Función de unidad lineal rectificada (Rectified Linear Unit - ReLU) | 20 |

| | |
|---|-----------|
| Leaky ReLU (LReLU) | 21 |
| Función de Unidad Lineal Exponencial (ELU) | 22 |
| 3.3.3. Arquitectura de las Redes Neuronales | 22 |
| 3.3.4. Proceso de aprendizaje de las Redes Neuronales | 23 |
| Backpropagation | 23 |
| 3.4. Tipos de Redes Neuronales | 24 |
| 3.4.1. Autoencoders | 24 |
| 3.4.2. Redes neuronales convolucionales | 25 |
| 3.4.3. Redes neuronales recurrentes | 26 |
| 3.4.4. LSTM | 27 |
| 3.4.5. GRU | 27 |
| 3.5. Técnicas de detección de anomalías | 29 |
| 3.5.1. One-Class SVM | 29 |
| 3.5.2. Isolation Forest | 30 |
| 3.5.3. Autoencoders | 31 |
| 3.6. Métricas de evaluación | 32 |
| 3.6.1. Precisión de clasificación (accuracy) | 33 |
| 3.6.2. Pérdida logarítmica (Logarithmic Loss) | 33 |
| 3.6.3. Matriz de confusión | 34 |
| 3.6.4. Área bajo la curva (AUC) | 35 |
| Tasa de Verdaderos Positivos (TPR) | 35 |
| Tasa de Verdaderos Negativos (TNR) | 35 |
| ROC (Receiver Operating Characteristics) | 35 |
| 3.6.5. F1 Score | 36 |
| Precisión (Precision) | 36 |
| Recuperación (Recall) | 37 |
| 4. CAPTURA Y PREPARACIÓN DE DATOS | 39 |
| 4.1. Captura de datos | 39 |
| 4.2. Preparación de datos | 41 |
| 4.2.1. Selección de datos | 41 |
| 4.3. Pre-procesamiento de datos | 42 |
| 4.3.1. Limpieza de datos | 43 |
| Técnicas de compensación de registros incompletos | 43 |
| Eliminar ruido (suavizado) de los datos | 43 |
| Fusión o integración de datos | 46 |
| Transformación de datos | 47 |
| Reducción de datos | 48 |
| 5. GENERACIÓN DEL MECANISMO DE DETECCIÓN DE ANOMALÍAS | 57 |
| 5.1. Entorno de desarrollo | 57 |
| 5.2. Conjunto de datos normales y anómalos | 58 |
| 5.2.1. Generación de series temporales | 58 |
| 5.3. Modelo de detección de anomalías | 59 |
| 5.3.1. Modelo del comportamiento normal | 60 |

| | |
|--|-----------|
| Arquitectura del modelo | 60 |
| 5.3.2. Método de detección de anomalías | 63 |
| Umbralización | 63 |
| Isolation Forest | 67 |
| One-Class SVM | 70 |
| Evaluación del método de detección de anomalías | 71 |
| 6. RESULTADOS Y EVALUACIÓN | 75 |
| 6.1. Evaluación de desempeño | 75 |
| 6.1.1. Evaluación en términos de rendimiento de detección | 75 |
| 6.2. Resultados | 76 |
| 6.2.1. Detección de anomalías del tipo zig zag | 77 |
| 6.2.2. Detección de anomalías del tipo giros a alta velocidad | 78 |
| 6.2.3. Detección de anomalías del tipo frenos en seco | 79 |
| 6.2.4. Detección de falsos positivos | 80 |
| 7. CONCLUSIONES Y TRABAJOS FUTUROS | 83 |
| 7.1. Conclusiones | 83 |
| 7.2. Trabajos futuros | 84 |
| BIBLIOGRAFÍA | 87 |
| Referencias | 87 |
| A. Experimentos de diferentes arquitecturas para los autoencoders | 93 |
| A.1. Redes densas | 94 |
| A.1.1. Redes densas para 3 componentes | 94 |
| A.1.2. Evaluación redes densas | 95 |
| A.2. Redes convolucionales | 95 |
| A.2.1. Redes convolucionales para 3 componentes | 95 |
| A.2.2. Evaluación redes convolucionales | 96 |
| A.3. Redes recurrentes | 96 |
| A.3.1. Redes recurrentes para 3 componentes | 96 |
| A.3.2. Evaluación redes recurrentes | 97 |
| B. Arquitectura del Sistema de Demostración | 99 |
| B.1. Arquitectura Física | 99 |
| B.2. Arquitectura Lógica | 100 |

Índice de figuras

| | |
|--|----|
| 1.1. Muertes por accidentes de tránsito por región en función del tipo de usuario (de Salud (OMS), s.f.). | 2 |
| 1.2. Árbol de problemas (Elaboración propia). | 3 |
| 2.1. Ejemplo de anomalías de punto en un conjunto de datos de 2 dimensiones (Varun y Arindam, 2009). | 6 |
| 2.2. Anomalía contextual t_2 en una serie temporal de temperatura (Varun y Arindam, 2009). | 6 |
| 2.3. Anomalía colectiva correspondiente a una contracción prematura auricular en un electrocardiograma humano (Varun y Arindam, 2009). | 7 |
| 2.4. Método de detección de anomalías propuesto (Elaboración propia). | 11 |
| 3.1. Gráfico de una neurona biológica. Reproducido desde (Wikipedia, s.f.). | 17 |
| 3.2. Gráfico de una neurona artificial. Reproducido desde (Jayesh, s.f.). | 18 |
| 3.3. Funciones de activación (Jing y Guanci, 2018). | 19 |
| a. Función Sigmoid | 19 |
| b. Función Tanh | 19 |
| c. Función ReLU | 19 |
| d. Función Leaky ReLu | 19 |
| e. Función ELU | 19 |
| 3.4. Arquitectura de una neurona artificial. Reproducido desde (Michael, 2015). | 22 |
| 3.5. Gráfico de un Autoencoder (Elaboración propia). | 24 |
| 3.6. Arquitectura de una Red Neuronal Convolucional (CNN) (Muhammad, s.f.). | 25 |
| 3.7. Procesamiento secuencial en una red neuronal recurrente (RNN) (Olah, s.f.). | 26 |
| 3.8. Estructura de LSTM. Reproducido desde Yan (Yan, s.f.). | 28 |
| 3.9. Estructura de GRU. Reproducido desde (Zhang, Lipton, Li, y Smola, 2019). | 29 |
| 3.10. One-Class SVM. Reproducido desde (Alashwal, Bin D., y Othman, 2006) | 30 |
| 3.11. Comparacion del rendimiento entre los algoritmos One-Clas SVM e Isolation Forest. Reproducido desde (0.22, s.f.). | 31 |
| 3.12. Detección de anomalías con autoencoder (Elaboración propia). | 32 |
| 3.13. Ejemplo de un curva AUC-ROC (Özler, s.f.). | 36 |
| 4.1. Recolección de datos, con intervalo de un segundo (Elaboración propia). | 40 |
| 4.2. Soporte para celular de parabrisas, posición horizontal (Elaboración propia). | 40 |
| 4.3. Fragmento del conjunto de datos obtenido (Elaboración propia). | 41 |
| 4.4. Gráfica de los sensores capturados en diferentes posiciones (Elaboración propia). | 42 |
| 4.5. Histograma de frecuencias del conjunto de datos (Elaboración propia). | 44 |

| | |
|--|-----|
| 4.6. Tabla de resultados estadísticos del conjunto de datos (Elaboración propia) | 45 |
| 4.7. Regla 68-95-99.7 (Galarnyk, s.f.) | 46 |
| 4.8. Regla 68-95-99.7 aplicada a los valores de los sensores del acelerómetro en Z (Elaboración propia) | 46 |
| 4.9. División del conjunto de entrenamiento (Elaboración propia) | 50 |
| 4.10. Visualización de los parámetros de conducción capturados (Elaboración propia) | 51 |
| 4.11. Gráfica resultante de aplicar diferentes tipos de normalizaciones a un conjunto de datos (Elaboración propia) | 52 |
| a. Min max Scaler | 52 |
| b. Standard Scaler | 52 |
| c. Max Scaler | 52 |
| d. Robust Scaler | 52 |
| 4.12. Gráfico de la varianza vs. el número de componentes (Elaboración propia) | 56 |
| 5.1. Gráfica resultante de diferentes tamaños de series de tiempo (Elaboración propia) | 59 |
| a. Serie de tiempo de 2 pasos. | 59 |
| b. Serie de tiempo de 3 pasos. | 59 |
| c. Serie de tiempo de 4 pasos. | 59 |
| d. Serie de tiempo de 5 pasos. | 59 |
| 5.2. Resultados (Elaboración propia) | 64 |
| a. Resultados de la red NN_33 | 64 |
| b. Resultados de la red CNN_33 | 64 |
| c. Resultados de la red RNN_33 | 64 |
| 5.3. Curva de los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia) | 65 |
| 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) | 66 |
| 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.) | 68 |
| 5.6. Representación gráfica del modelo de comportamiento normal o autoencoder (Elaboración propia) | 68 |
| 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) | 69 |
| 5.8. Mecanismo de detección de anomalías (Elaboración propia) | 72 |
| 6.1. Resultados de la detección de anomalías del tipo zig zag (Elaboración propia) | 77 |
| 6.2. Resultados de la detección de anomalías del tipo giros a alta velocidad (Elaboración propia) | 79 |
| 6.3. Resultados de la detección de anomalías del tipo frenos en seco (Elaboración propia) | 80 |
| 6.4. Resultados de la detección de falsos positivos (Elaboración propia) | 81 |
| B.1. Arquitectura física del Sistema de Demostración (Elaboración propia) | 99 |
| B.2. Arquitectura Lógica del Sistema de Demostración (Elaboración propia) | 100 |

Índice de cuadros

| | |
|--|----|
| 3.1. Matriz de confusión, para una clasificación binaria (Elaboración propia) | 34 |
| 4.1. Tabla de división del conjunto de datos (Elaboración propia). | 50 |
| 4.2. Tabla con estadísticas descriptivas de los datos escalados con diferentes técnicas (Elaboración propia). | 54 |
| 5.1. Tabla del conjunto de anomalías (Elaboración propia). | 58 |
| 5.2. Tabla de los métodos comparados (Elaboración propia). | 60 |
| 5.3. Arquitectura densa para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia). | 61 |
| 5.4. Arquitectura convolucional para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia). | 61 |
| 5.5. Arquitectura recurrente para una secuencia de 3 pasos y 3 componentes principales (Elaboración propia). | 62 |
| 5.6. Evaluación de las redes NN_33, CNN_33 y RNN_33 (Elaboración propia). | 63 |
| 5.7. Evaluación de la detección de anomalías para cada codo obtenido con los diferentes valores de sensibilidad (Elaboración propia). | 67 |
| 5.8. Evaluación de la detección de anomalías usando Isolation forest para valores compresos (Elaboración propia). | 69 |
| 5.9. Evaluación de la detección de anomalías usando Isolation forest para errores de reconstrucción (Elaboración propia). | 70 |
| 5.10. Evaluación de la detección de anomalías usando One-Class SVM para valores compresos (Elaboración propia). | 70 |
| 5.11. Evaluación de la detección de anomalías usando One-Class SVM para el error de reconstrucción del autoencoder (Elaboración propia). | 70 |
| 5.12. Comparación de los mejores métodos de detección de anomalías (Elaboración propia). | 71 |
| 6.1. Matriz de confusión, para el mecanismo de detección de anomalías (Elaboración propia). | 76 |
| 6.2. Resultados (Elaboración propia). | 78 |
| 6.3. Resultados (Elaboración propia). | 79 |
| 6.4. Resultados (Elaboración propia). | 80 |
| A.1. Arquitectura densa para 3 componentes principales (Elaboración propia). | 94 |
| A.2. Tabla de evaluación de redes densas (Elaboración propia). | 95 |
| A.3. Arquitectura convolucional para 3 componentes principales (Elaboración propia). | 95 |

| | |
|---|----|
| A.4. Tabla de evaluación de redes convolucionales (Elaboración propia). | 96 |
| A.5. Arquitectura recurrente para 3 componentes principales (Elaboración propia). . . . | 96 |
| A.6. Tabla de evaluación de redes recurrentes (Elaboración propia). | 97 |

Resumen

El presente trabajo describe el desarrollo de un mecanismo de detección de anomalías de conducción, el cual es implementado usando un dispositivo móvil y técnicas de Aprendizaje Automático.

El objetivo es crear una herramienta capaz de encontrar comportamientos anómalos en la conducción de un agente humano o autónomo, teniendo un previo conocimiento de las conductas normales de conducción del mismo. Asimismo se presenta antecedentes de trabajos e investigaciones de la detección de anomalías de conducción de todo el mundo, se analiza los parámetros de conducción obtenidos por el dispositivo móvil, y se presenta la propuesta para identificar anomalías mediante el uso de Redes Neuronales y Bosques de Aislamiento, un método de Aprendizaje Automático que es comúnmente utilizado para la detección de anomalías.

El trabajo cuenta con dos partes principales: un modelo ajustado al comportamiento normal de conducción de un agente, y un método de detección de anomalías, los cuales fueron entrenados iterativamente con 30000 muestras, las cuales corresponden sólo al comportamiento normal de conducción.

La precisión de detección del mecanismo completo propuesto en este documento, es de 67.68 % que fue evaluado con 44040 datos, de los cuales 164 corresponden a muestras anómalas, siendo así una de las contribuciones más sobresalientes para la detección de anomalías de conducción semi-supervisada.

Capítulo 1

INTRODUCCIÓN

El presente documento describe el desarrollo de un método para la detección de anomalías en la conducción de automóviles. Se propone el uso de técnicas de Aprendizaje Automático para generar un mecanismo que identifique anomalías de manejo, de tal modo que éstas puedan usarse para alertar oportunamente a los agentes y así logren correjir sus conductas de conducción.

La idea principal, es generar un modelo que aprenda el comportamiento normal de conducción de un agente concreto, para posteriormente detectar de forma autónoma aquellos comportamientos inesperados e informarlos como anomalías, de manera que se pueda evitar un accidente de tránsito o reducir los efectos del mismo.

1.1. Planteamiento del problema

Debido a las graves secuelas que causan sobre las personas y los altos costos económicos asociados a ellos, los accidentes de tránsito se catalogan como un problema social y de salud pública mundial.

Según la Organización Mundial de Salud (OMS) cada año existen aproximadamente 1,25 millones de muertes a causa de accidentes de tránsito, agregando que la mitad de todas estas victimas son peatones, ciclistas y motociclistas (Véase la figura 1.1 pag. 2). Asimismo se puede decir que son una de las causas de muerte más importantes en el mundo, y la principal causa de muerte entre personas de edades comprendidas entre los 15 y los 29 años.

Por otro lado según la Unidad Operativa de Tránsito de Cochabamba los accidentes registrados en 2017 provocaron la muerte de 200 personas y dejaron aproximadamente 2200 heridos.

En la figura 1.2 se muestra las causas por las cuales se ocasiona un accidente de tránsito, se puede observar que gran parte de éstas se deben al factor humano sin embargo hay otras

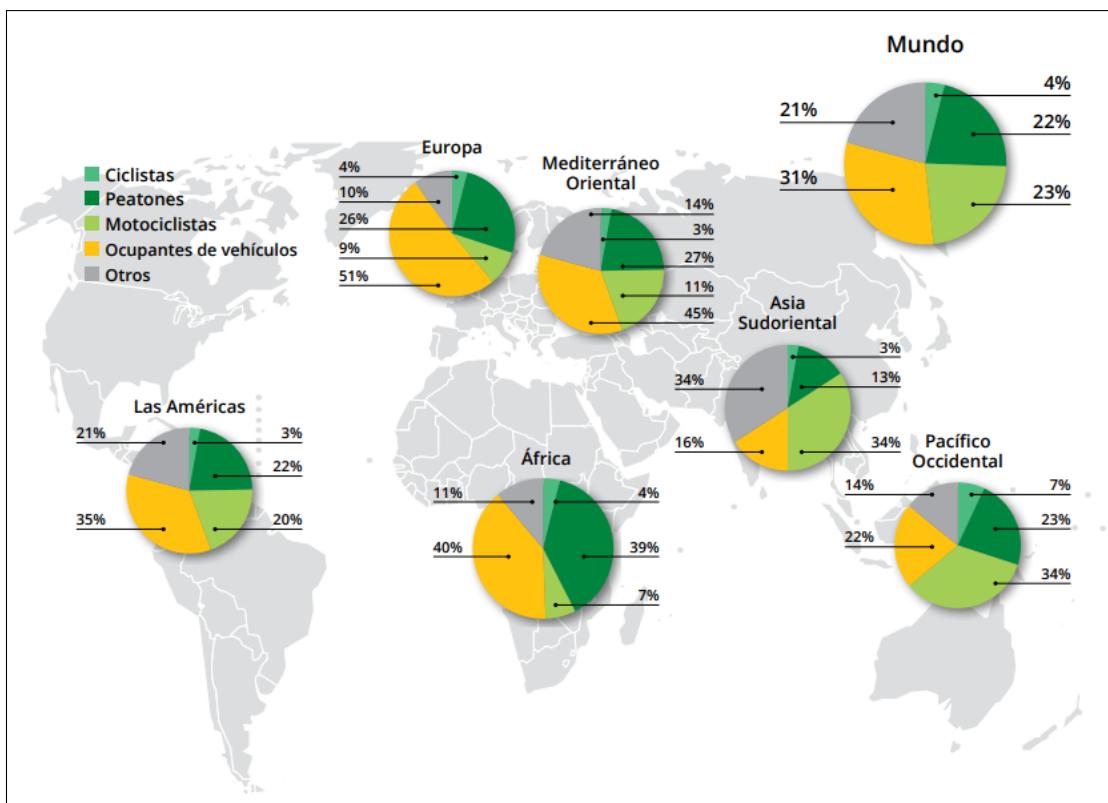


FIGURA 1.1: Muertes por accidentes de tránsito por región en función del tipo de usuario (de Salud (OMS), s.f.).

que conllevan factores medio-ambientales y mecánicos, por lo que se hace imposible evitar completamente los mismos.

Es por ello que se hace necesario el contar con mecanismos para prevenir y/o actuar de forma oportuna ante posibles accidentes de tránsito, motivo por el cual el presente trabajo se centra en estudiar los comportamientos de conducción, para así generar alertas al encontrar un comportamiento anómalo en el manejo, de manera que se pueda evitar o en todo caso minimizar los efectos del mismo.

1.2. Objetivo general

El objetivo general del presente trabajo es desarrollar un mecanismo de detección de anomalías de conducción, mediante el uso de un dispositivo móvil y algoritmos de Aprendizaje Automático, con el fin de alertar de forma oportuna el hallazgo de un patrón anómalo en el manejo, tal como cansancio, ebriedad, o problemas de salud, ej, epilepsia.

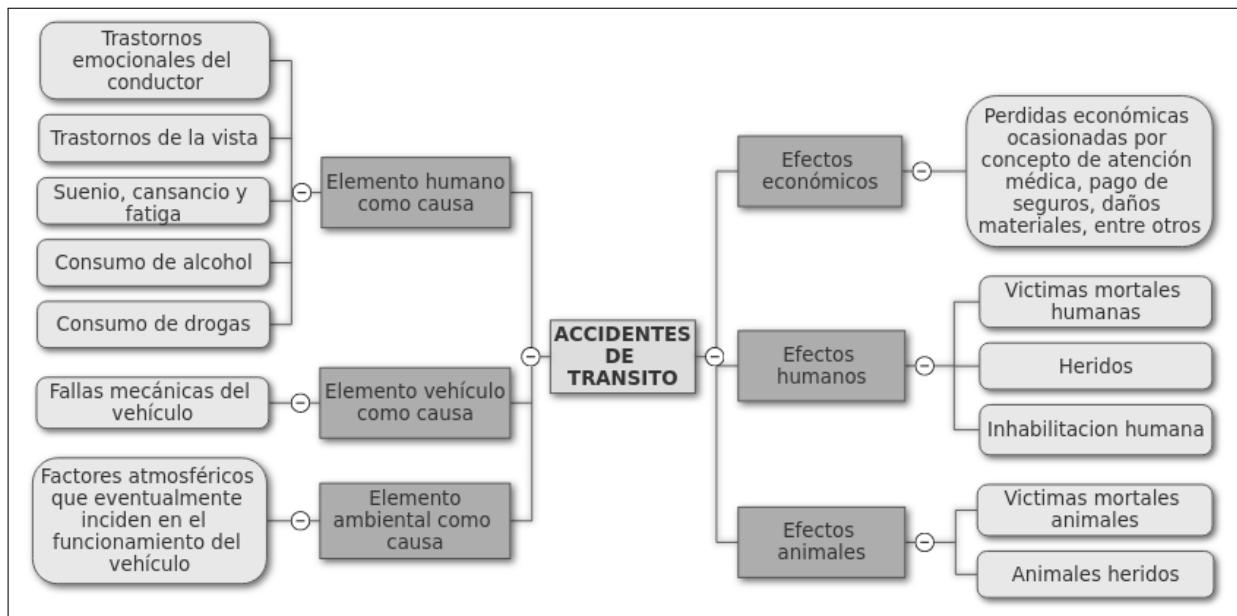


FIGURA 1.2: Árbol de problemas (Elaboración propia).

1.3. Objetivos específicos

- Capturar los parámetros de manejo de un conductor mediante el uso de sensores de un dispositivo móvil.
- Escalar los parámetros de manejo mediante técnicas de pre-procesamiento de datos.
- Generar un modelo de aprendizaje automático que se ajuste a un comportamiento normal de manejo.
- Definir un método de detección de anomalías para generar una alerta de conducción anormal.
- Evaluar el método de detección de anomalías con nuevas muestras.

1.4. Justificación

Los accidentes de tránsito cobran un número inaceptable de víctimas cada año, especialmente en las regiones más pobres del mundo. Esto se debe a diversos aspectos, pero el principal recae en el bajo nivel de conciencia ciudadana que existe, lo que conlleva a que muchas personas conduzcan bajo los efectos del alcohol, con exceso de velocidad, manipulando sus dispositivos móviles, entre otros. Por ello este trabajo busca establecer patrones de comportamientos de conducción mediante el uso de un dispositivo móvil y técnicas de Aprendizaje Automático, de manera que se logre realizar una detección de anomalías de conducción oportuna.

1.4.1. Justificación práctica

Detectar anomalías de conducción permite generar una alerta oportuna a las autoridades o a los agentes para que logren corregir sus conductas de conducción de forma rápida. De esta manera se podrá evitar accidentes de tránsito o minimizar sus efectos, permitiendo así reducir la cantidad de daños, tanto materiales como personales.

1.4.2. Justificación metodológica

El estudio realizado en el desarrollo del presente trabajo de investigación permite resaltar la eficiencia de las técnicas de Inteligencia Artificial en la detección de anomalías.

1.5. Límites y alcances

Debido a que la realización de pruebas de campo para ésta investigación es bastante peligrosa, se limitó los ejemplos de conducción anómala a:

- Frenos en seco.
- Giros hacia la derecha e izquierda a alta velocidad.
- Giros en zig zag bruscos.

Siendo así, los experimentos y pruebas se realizaron sólo sobre un pequeño conjunto de ejemplos anómalos, por lo tanto no se espera que el modelo de detección propuesto funcione de manera correcta sobre aquellos ejemplos que no fueron considerados.

1.6. Método de investigación

El presente estudio se realizó con un enfoque experimental, teniendo como hipótesis la siguiente:

¿Es posible detectar anomalías de conducción mediante el uso de un dispositivo móvil y algoritmos de Aprendizaje Automático?

Capítulo 2

FUNDAMENTOS DE LA DETECCIÓN DE ANOMALÍAS

En este capítulo se aborda los conceptos necesarios que se necesita para comprender la detección de anomalías, así también se expone los diferentes proyectos e investigaciones realizados en el campo de la detección de anomalías de conducción hasta la fecha.

2.1. Detección de anomalías

Para comprender lo que implica la detección de anomalías, es necesario asimilar lo que es una anomalía, y de qué maneras éstas pueden presentarse. Por lo tanto, se puede decir que las **anomalías**, o valores atípicos, son patrones en los datos que no se ajustan a una noción bien definida de un comportamiento normal.

Las anomalías pueden ser clasificadas dentro de una de las tres siguientes categorías:

1. **Anomalías de punto:** Las anomalías de punto son simplemente instancias únicas y anómalas dentro de un conjunto de datos más grande, es decir, éstas se encuentran separadas del resto de los datos. Por ejemplo, en la Figura 2.1, los puntos o_1 , o_2 y la región O_3 se encuentran fuera de los límites de las regiones normales (N_1 y N_2), y por lo tanto son anomalías puntuales debido a que son diferentes al conjunto de datos normales.

Este tipo de anomalía se considera la más simple y es el foco de la mayoría de las investigaciones enfocadas en la detección de valores atípicos.

2. **Anomalías contextuales (o condicionales):** Estos son puntos que sólo se consideran anómalos en un contexto específico. La noción de este contexto es inducida por la estructura en el conjunto de datos y debe especificarse como parte de la formulación del problema.

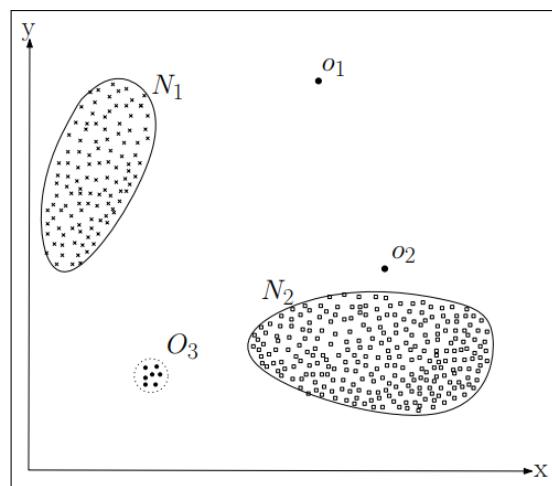


FIGURA 2.1: Ejemplo de anomalías de punto en un conjunto de datos de 2 dimensiones (Varun y Arindam, 2009).

Este tipo de anomalías se han explorado con mayor frecuencia en los datos de series de tiempo y datos espaciales. La figura 2.2 muestra un ejemplo de serie temporal de la temperatura mensual de un área en los últimos 5 años, se debe tener en cuenta que la temperatura en el tiempo t_1 es la misma que en el tiempo t_2 , pero se produce en un contexto diferente, por lo tanto t_2 es considerada una anomalía.

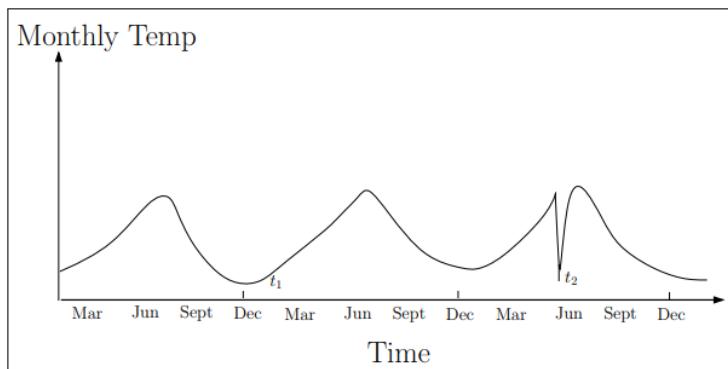


FIGURA 2.2: Anomalía contextual t_2 en una serie temporal de temperatura (Varun y Arindam, 2009).

3. **Anomalías colectivas:** Si una colección de instancias de datos relacionadas es anómala con respecto a todo el conjunto de datos, se denomina anomalía colectiva. Las instancias de datos individuales en una anomalía colectiva pueden no ser anomalías por sí mismas, pero su aparición conjunta como una colección es anómala.

La figura 2.3 ilustra un ejemplo que muestra una salida de electrocardiograma humano, se puede notar que la región resaltada en rojo denota una anomalía porque existe el mismo

valor bajo durante un tiempo anormalmente prolongado (que corresponde a una Contracción prematura auricular). Se debe tener en cuenta que ese valor bajo por sí mismo no es considerada una anomalía.

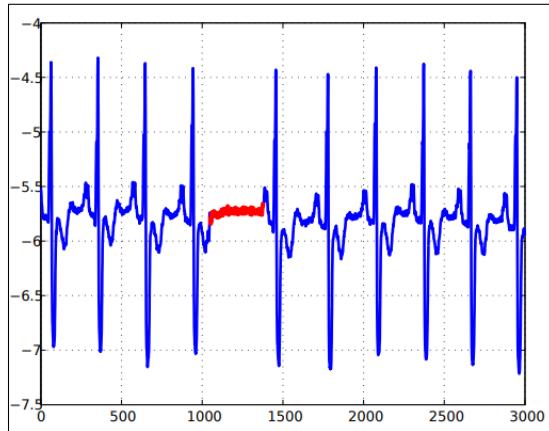


FIGURA 2.3: Anomalía colectiva correspondiente a una contracción prematura auricular en un electrocardiograma humano (Varun y Arindam, 2009).

En relación a lo expuesto previamente se puede definir como **detección de anomalías**, ó **valores atípicos**, a la identificación de puntos de datos, elementos, observaciones o eventos que no se ajustan al patrón esperado de un grupo determinado.

La detección de anomalías, se usa en distintos dominios de aplicaciones, por ejemplo: procesamiento de imágenes, detección de fraudes de tarjeta, sistemas de detección de intrusión de red, etcetera.

2.2. Desafíos en la detección de anomalías

En un nivel abstracto, la detección de anomalías puede parecer una tarea simple. Sin embargo puede llegar a ser una tarea muy desafiante. A continuación se presenta algunos de estos desafíos.

- La definición de regiones normales es bastante difícil. En muchos casos, los límites entre las anomalías y los datos normales no son precisos. Por lo tanto, las observaciones normales podrían considerarse anomalías y viceversa.
- Lo que se considera normal hoy en día, puede no ser normal en el futuro.
- La mayor parte de las veces los enfoques para la detección de anomalías en un campo específico no se pueden utilizar en otro campo.
- La poca disponibilidad de ejemplos positivos (anomalías) para el entrenamiento y validación del modelo de detección de anomalías.

2.2.1. Enfoques de detección de anomalías

Los enfoques que se pueden usar para este propósito se clasifican en las siguientes categorías:

Detección de anomalías supervisada

El uso de técnicas de aprendizaje supervisado requiere la disponibilidad de un conjunto de datos de entrenamiento etiquetados, tanto para clases normales como anómalas. El enfoque principal es construir un modelo predictivo para clases normales vs. anomalías, posteriormente tomar cualquier instancia de datos no visto, comparar con el modelo y determinar a qué clase pertenece.

Existen dos principales inconvenientes que surgen con el uso de ésta técnica.

- La cantidad de instancias anómalas es muy inferior a la de las instancias normales, lo que genera un desequilibrio de distribución de clases durante el entrenamiento.
- La obtención de etiquetas precisas y representativas, en particular para la clase de anomalía, es un desafío.

Detección de anomalías semi-supervisada

Estas técnicas requieren un conjunto de entrenamiento con instancias etiquetadas, pero solo para la clase normal, esto hace que su uso sea más aplicable que las técnicas supervisadas, ya que no se requiere etiquetas para la clase anomalía.

El enfoque típico usado en éstas técnicas es construir un modelo para la clase correspondiente al comportamiento normal, y usar el modelo para identificar anomalías en los datos de prueba.

Detección de anomalías no supervisada

Las técnicas que operan de manera no supervisada no requieren datos de entrenamiento, razón por la cual son las más ampliamente utilizadas. Estas técnicas suponen que las instancias normales son mucho más frecuentes que las anomalías en los datos de prueba, en caso de que esta suposición no sea cierta, tales técnicas sufren de una alta tasa de falsas alarmas.

2.3. Trabajo relacionado

La identificación de comportamientos de conducción anormal es una parte indispensable para mejorar la seguridad de conducción, sin embargo, como se describió previamente, ésta no

es una tarea sencilla. En los últimos años, se han propuesto varias técnicas para detectar conductas de conducción. Esta sección esta dedicada al repaso de las mismas.

Dang-Nhac y cols. (2018), proponen un sistema combinado que se compone de dos módulos: uno para detectar el tipo de vehículo de los usuarios y el otro para detectar los eventos de conducción instantánea, independientemente de la orientación y la posición de los teléfonos inteligentes, éste sistema logra una precisión promedio del 98.33 % en la detección del tipo del vehículo (automóvil, motocicleta, bicicleta, entre otros) y una precisión promedio de 98.95 % en el reconocimiento de los eventos de conducción de los motociclistas al usar Random Forest como clasificador.

Por otra parte Ferreira y cols. (2017) presentan una evaluación cuantitativa de 4 algoritmos de Aprendizaje Automático (Bayesian Network BN, Artificial Neural Network ANN, Random Forest RF y Support Vector Machine SVM) con diferentes configuraciones, aplicadas en la detección de 7 tipos de eventos de conducción, entre eventos normales y agresivos, utilizando datos recopilados de 4 sensores de teléfonos inteligentes Android (acelerómetro, aceleración lineal, magnetómetro y giroscopio); dando como resultado que el giroscopio y el acelerómetro son los mejores sensores para detectar eventos de conducción y que Random Forest (RF) es por lejos el Algoritmo de Aprendizaje Automático de mejor rendimiento, seguido de la forma más simple de ANN el Multi Layer Perceptron (MLP).

Johnson y Trivedi (2011) proponen el sistema MIROAD el cual muestra que el Dynamic Time Warping (DTW) es un algoritmo válido para detectar maniobras de conducción potencialmente agresivas, donde casi todos los eventos agresivos (97 %) se identificaron correctamente, utilizando el conjunto de sensores T (acelerómetro, giroscopio y el tono de voz). Así también en el trabajo de Kridalukmana, Yan-Lu, y Naderpour (2017) se propone un sistema enfocado a desarrollar la conciencia del conductor mediante notificaciones en situaciones críticas que pueden desencadenar maniobras de conducción inseguras, mediante un modelo para detectar situaciones peligrosas basadas en Object-Oriented Bayesian Network (OOBN).

Así como los trabajos presentados previamente existe gran cantidad de trabajos (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) que usan los sensores de los teléfonos inteligentes (acelerómetro y giroscopio) para la detección de conducción agresiva, esto debido a que se tiene la ventaja de no comprar ni instalar ningún dispositivo y además de ser altamente portátil, sin embargo se depende bastante del rendimiento del receptor GPS y no es aplicable en áreas no disponibles para GPS.

Existen además otros enfoques para la detección de conducción agresiva de un conductor, por ejemplo en el trabajo de Who-Lee, Sik-Yoon, Min-Song, y Ryoung-Park (2018) se propone un

método basado en Convolutional Neural Network (CNN) para detectar la emoción de conducción agresiva, mediante la utilización de imágenes faciales de un conductor obtenidas con una cámara de luz NIR y una cámara térmica.

2.4. Enfoque sobre el problema

Es evidente que éste tema fue ampliamente investigado y que tiene una gran variedad de propuestas de solución, sin embargo la mayoría de estas se basan en la detección mediante técnicas de aprendizaje supervisado, lo cual presenta la gran desventaja de requerir datos etiquetados para generar el modelo de detección; además gran parte de los trabajos relacionados proponen modelos generalizados para la detección y no así modelos específicos por cada agente, lo cual es crucial debido a que cada agente presenta conductas individuales de conducción y conducen en condiciones distintas, es decir, la conducción de un agente que circula por avenidas pavimentadas será distinta a la conducción de un agente que circula por calles empedradas o la conducción de un agente que circula por avenidas o calles concurridas será distinta a de los agentes que circulen por calles relativamente descongestionadas.

Con la propuesta que se hace en este trabajo se pretende brindar un prototipo de una herramienta que ayude a analizar las secuencias de los sensores de movimiento de un dispositivo móvil y permita detectar anomalías a partir de la información obtenida de este análisis.

Para lograr este objetivo, se captura el conjunto de datos de los sensores de movimiento de un teléfono inteligente, mediante una aplicación móvil, posteriormente se divide y prepara los datos con técnicas de pre-procesamiento de datos. Una vez terminadas estas fases se entrena un modelo con el conjunto de entrenamiento y se valida con el conjunto de desarrollo. Finalmente se elige un modelo óptimo y una técnica para clasificar los valores atípicos, con el objetivo de cubrir un rango completo de los comportamientos normales y excluir con la mayor precisión posible las anomalías. Este método se puede apreciar mejor en la Figura 2.4.

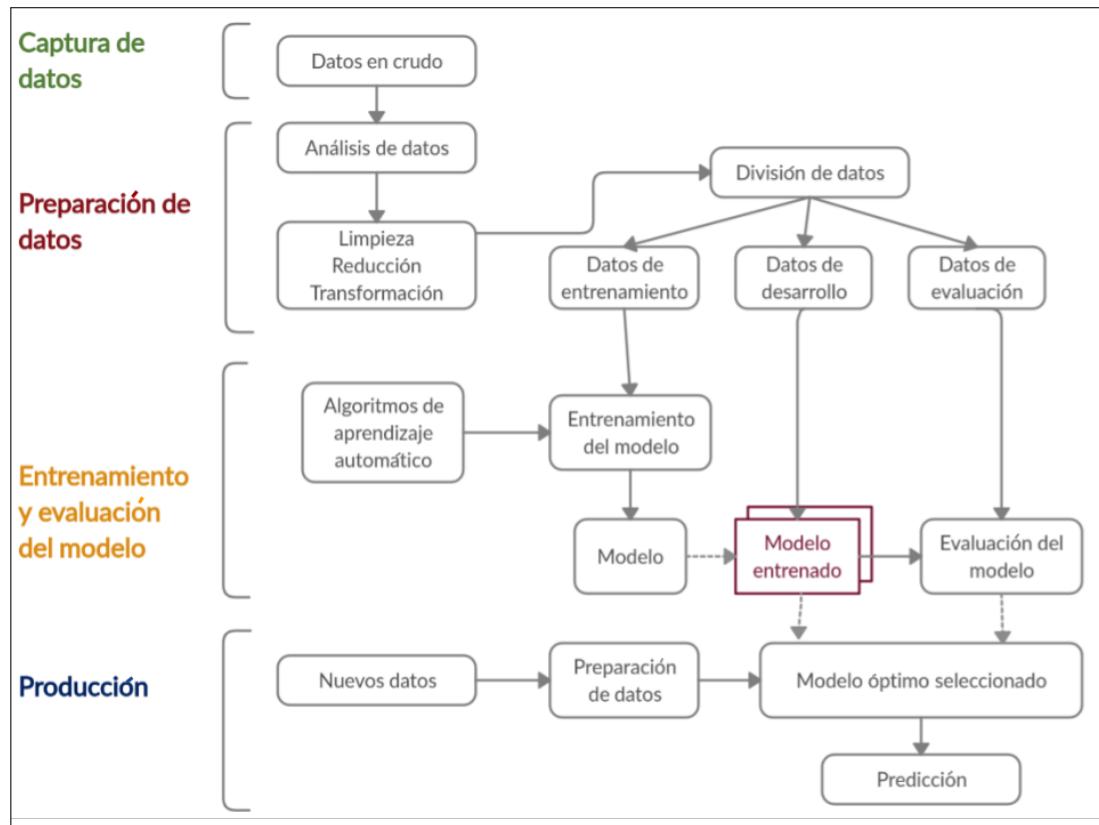


FIGURA 2.4: Método de detección de anomalías propuesto (Elaboración propia).

Capítulo 3

APRENDIZAJE AUTOMÁTICO PARA LA DETECCIÓN DE ANOMALÍAS

El término Aprendizaje Automático se refiere a la detección automática de patrones significativos dentro de un conjunto de datos (Shai y Shai, 2014). En las últimas décadas se ha convertido en una herramienta común en casi cualquier tarea que requiera la extracción de información de gran cantidad de datos, por lo cual se ha convertido en una de las áreas de más rápido crecimiento de la informática.

Si bien el Aprendizaje Automático puede resolver algunos problemas que son resueltos con algoritmos tradicionales, ha superado a éstos en problemas tales como el reconocimiento de imágenes, voz, lenguaje, escritura, juegos, robótica, análisis de datos, análisis de series de tiempo, etc. Desde esta perspectiva, se espera que mediante la aplicación del Aprendizaje Automático se pueda generar un modelo que se ajuste a un comportamiento normal esperado para el agente.

Por lo tanto en este capítulo se detalla las bases teóricas necesarias para abordar el desarrollo del método de detección de anomalías de conducción. En primer lugar, se describe los diferentes paradigmas de aprendizaje que existen y los diferentes enfoques de modelos, luego, se realiza una descripción del funcionamiento de las redes neuronales y se muestra los diferentes tipos de redes, así como también se presenta las diferentes técnicas de detección de anomalías que hay y por último se muestra las diferentes métricas de evaluación que presentan los modelos de aprendizaje automático.

3.1. Aprendizaje Supervisado, Aprendizaje no Supervisado y Aprendizaje Semi supervisado

Existen diversas formas de clasificar los paradigmas de aprendizaje que existen, sin embargo en el presente trabajo sólo se tratarán el supervisado, el no supervisado y el semi-supervisado.

3.1.1. Aprendizaje Supervisado

El **Aprendizaje Supervisado** es aquel que cuenta con variables de entrada (X) y una variable de salida (Y), este tipo de aprendizaje utiliza un algoritmo para aprender la función de mapeo desde la entrada hasta la salida.

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

El objetivo de este tipo de aprendizaje es aproximar la función de mapeo de tal forma que cuando tenga datos de entrada nuevos (X) pueda predecir las variables de salida (Y) para esos datos.

Este tipo de aprendizaje aborda dos tipos de problemas: clasificación y regresión. Los problemas de **Clasificación** son aquellos donde la variable de salida es una categoría, como por ejemplo: "Rojo", "Azul", o "Sano", "Enfermo", por otra parte en los problemas de **Regresión** la variable de salida es un valor real, tal como: "precio" o "altura". Algunos de los tipos de problemas más comunes construidos sobre la clasificación y la regresión incluyen la recomendación y la predicción de series temporales.

3.1.2. Aprendizaje no Supervisado

Por otro lado el **Aprendizaje no Supervisado** es aquel donde sólo se cuenta con datos de entrada (X) y no hay variables de salida correspondientes, su objetivo principal consiste en modelar la estructura o distribución subyacente en los datos para aprender más acerca de los mismos.

En cuanto a los problemas del Aprendizaje sin supervisión, pueden ser agrupados en dos: agrupamiento y asociación. El **Agrupamiento** es aquel donde se desea descubrir las agrupaciones inherentes en el conjunto de datos, como por ejemplo agrupar clientes por comportamiento de compra. Por otra parte la **Asociación** es aquella que desea descubrir reglas que describen grandes porciones de sus datos, por ejemplo las personas que compran X también tienden a comprar Y. Algunos de los algoritmos de aprendizaje sin supervisión más populares son: K-means (para problemas de agrupamiento) y algoritmo Apriori (para problemas de aprendizaje de reglas de asociación).

3.1.3. Aprendizaje Semi-Supervisado

Por último se encuentra el **Aprendizaje Semi-supervisado**, el cual abarca aquellos problemas donde se tiene gran cantidad de datos de entrada (X) y sólo algunos de los datos están etiquetados (Y). Este tipo de problemas se encuentran entre el aprendizaje supervisado y el no supervisado, además es importante señalar que muchos de los problemas de Aprendizaje

Automático en el mundo real se encuentran en esta área, esto debido a que resulta costoso o puede requerir mucho tiempo etiquetar el conjunto de datos, mientras que los datos no etiquetados son baratos, además de ser fáciles de recolectar y almacenar. Este tipo de problemas pueden usar una combinación de técnicas supervisadas y no supervisadas para ser resueltos.

Dado que los métodos de Aprendizaje Supervisado requieren una gran cantidad de datos de entrenamiento etiquetados, es importante aclarar que la recolección de muestras negativas (conducción anómala) es difícil y riesgosa para este estudio en particular; además el enfoque supervisado presenta una limitación potencial, la cual es: la detección de nuevos patrones atípicos, esto debido a que el modelo resultante sólo está entrenado para reconocer un conjunto limitado de patrones anómalos, por lo cual al momento en que se presente un nuevo patrón este modelo será incapaz de reconocerlo.

Por otra parte el enfoque sin supervisión tiene la ventaja de no requerir información etiquetada, sin embargo a menudo sufre altas tasas de falsas alarmas y bajas tasas de detección (Xue, Shang, y Feng, 2010).

En muchas aplicaciones, incluyendo la del presente estudio, los ejemplos normales son fáciles de conseguir, mientras que los anómalos son bastante difíciles de obtener, en consecuencia, para la realización de este estudio, se ha optado por la aplicación del enfoque Semi-supervisado. De esta manera, como se mencionó en el Capítulo 2, el enfoque de **detección de anomalías Semi-supervisado** sólo dispone de muestras normales en el conjunto de entrenamiento; es decir, no se puede obtener información sobre anomalías, por lo tanto las muestras desconocidas se clasifican como valores atípicos, siempre y cuando su comportamiento sea muy diferente al de las muestras normales ya conocidas.

Como se mencionó en esta sección todos estos enfoques de aprendizaje se basan en generar un **Modelo** capaz de ayudar ya sea en tareas de clasificación, agrupación, etc; sin embargo existe más de un tipo de modelos. En la siguiente sección se detallará en profundidad los diferentes tipos de modelos que existen.

3.2. Modelos Generativos y Discriminativos

Cuando se utiliza Aprendizaje Automático existen dos principales enfoques para entender (modelar) el mundo real y tomar decisiones. Estos dos enfoques son los modelos discriminativos y generativos. Más formalmente los modelos generativos y discriminativos representan dos distintas estrategias para estimar la probabilidad que un objeto en particular pertenece a una categoría (Hsu y Griffiths, 2010).

Los **modelos discriminativos** se basan en la probabilidad condicionada $P(Y|X)$, es decir, aprenden un mapa directo de un conjunto de características X a etiquetas de clases Y . Este tipo de modelos intentan modelar simplemente dependiendo de los datos observados (conjunto de datos), además hacen menos suposiciones sobre las distribuciones; sin embargo dependen en gran medida de la calidad de los datos. Algunos ejemplos de modelos discriminativos son: Regresión Logística, SVM (Support Vector Machine - Máquina de vectores de soporte), Redes Neuronales, Random Forest, entre otros.

Por otra parte los **modelos generativos** apuntan a una descripción probabilística completa del conjunto de datos, su objetivo es desarrollar la distribución de probabilidad conjunta $P(X,Y)$, ya sea directamente o calculando $P(Y|X)$ y $P(X)$, para luego inferir las probabilidades condicionadas requeridas para clasificar nuevos datos. Estos modelos ayudan a especificar la incertidumbre de un modelo, algunos ejemplos de modelos generativos son: Gaussian Mixture Model, Hidden Markov Model, Restricted Boltzmann Machine, Generative Adversarial Networks (GAN), entre otros.

Los modelos discriminativos han estado a la vanguardia del éxito del Aprendizaje Automático en los últimos años, ya que estos modelos hacen predicciones que dependen de una entrada dada, aunque no puedan generar nuevas muestras o datos, por lo que en el presente estudio se dará preferencia al uso de modelos discriminativos.

A continuación se realizará un repaso de las bases teóricas fundamentales de algunas técnicas del Aprendizaje Semi-Supervisado con un enfoque discriminativo, para posteriormente detallar que tipo de algoritmos se aplicará en el método propuesto en este trabajo de investigación.

3.3. Redes Neuronales Artificiales

La Red Neuronal Artificial o ANN¹ es un paradigma de procesamiento de información inspirado en la manera en la que el sistema nervioso biológico procesa la información. Se compone de una gran cantidad de elementos de procesamiento (neuronas) altamente interconectados que trabajan al unísono para resolver un problema específico.

3.3.1. Neuronas o nodos

Las **neuronas biológicas** (células nerviosas) son las unidades fundamentales del cerebro y del sistema nervioso. Las neuronas son las células responsables de recibir información sensorial del mundo externo a través de las dendritas, procesarla y dar una salida a través del axón (Ver Figura 3.1).

¹ ANN, Artificial Neural Network (Red Neuronal Artificial)

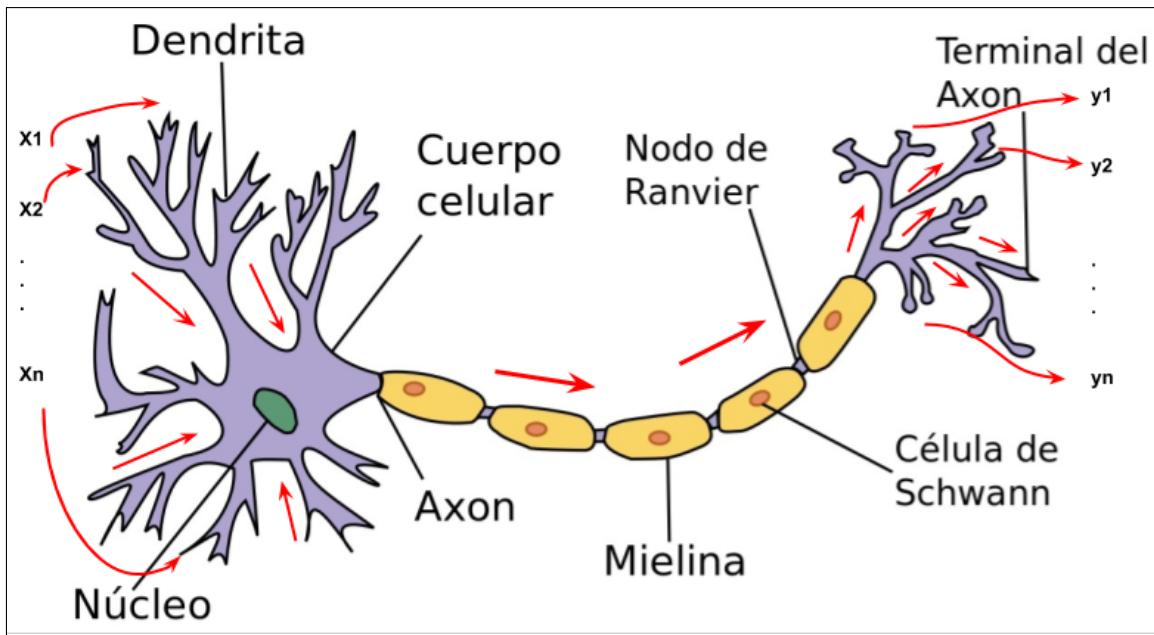


FIGURA 3.1: Gráfico de una neurona biológica. Reproducido desde (Wikipedia, s.f.).

Una neurona cerebral puede recibir unas 10000 entradas y enviar a su vez su salida a cientos de neuronas.

La conexión entre neuronas se llama **sinapsis**, esta no es una conexión física, sino que existe 2 mm. de separación entre neuronas. Estas conexiones son unidireccionales, donde la transmisión de la información se hace de forma eléctrica en el interior de la neurona y de forma química entre neuronas, gracias a los neurotransmisores.

Una **neurona artificial** es un procesador elemental, debido a que procesa un vector $x(x_1, x_2, \dots, x_n)$ de entradas y produce una respuesta o salida única. Los elementos principales de una neurona artificial son los siguientes:

- **Las entradas** que reciben los datos de otras neuronas, estas entradas serían las dendritas de una neurona biológica.
- **Los pesos sinápticos w_{ij}** . En una neurona artificial a aquellas entradas que vienen de otras neuronas se les asigna un peso (factor de importancia). Este peso es un valor numérico que se modifica durante el proceso de entrenamiento de una red neuronal, y por lo tanto es aquí donde se almacena la información que hace que la red sirva para un propósito u otro.
- **Regla de propagación**. Con las entradas y los pesos sinápticos, se suele hacer algún tipo de operación para obtener el valor potencial postsináptico; una de las operaciones

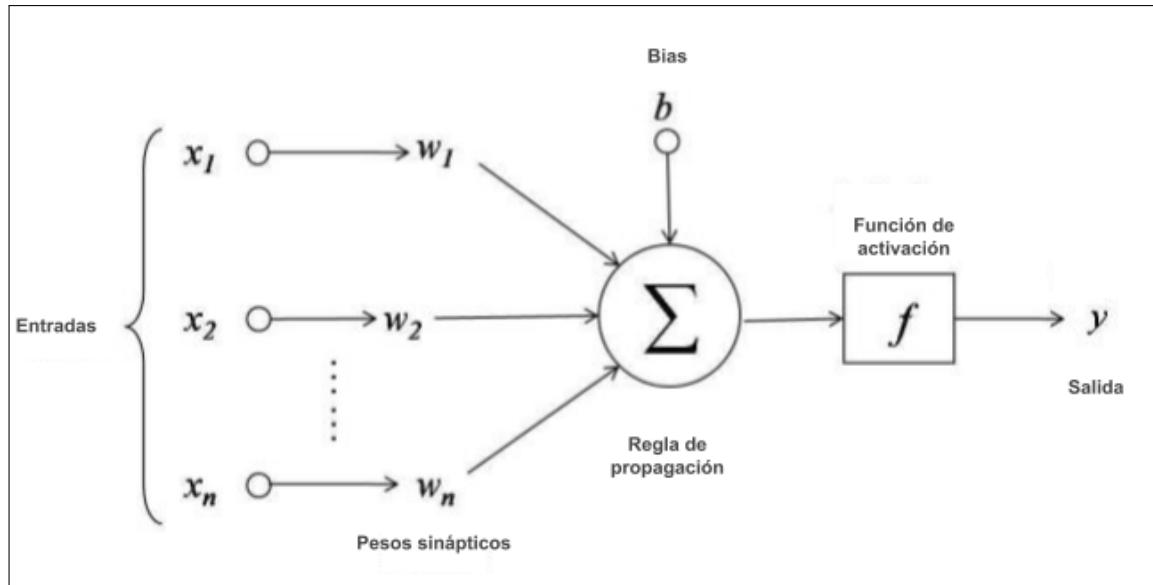


FIGURA 3.2: Gráfico de una neurona artificial. Reproducido desde (Jayesh, s.f.).

más comunes es sumar las entradas, pero teniendo en cuenta la importancia (peso sináptico) de cada una; esta operación se llama *suma ponderada*. Sin embargo otras operaciones también son posibles. Otra regla de propagación que es habitual es la distancia euclídea.

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

- **Función de activación.** El valor obtenido con la regla de propagación, se filtra a través de una función conocida como *función de activación* y es la que da la salida de la neurona. La función de activación es importante debido a que es la que decide si una neurona debe activarse o no, además si esta función no se aplica la señal de salida de la neurona sería simplemente una función lineal.

3.3.2. Tipos de funciones de activación

Existen diferentes funciones de activación, a continuación solo se presentará las más usadas en el ámbito de las redes neuronales.

Función de activación sigmoide (función logística)

Una función sigmoide es una función matemática que tiene una curva característica en forma de "S" o una curva sigmoidea que oscila entre 0 y 1 (Ver Figura 3.3a), por lo que esta función suele ser utilizada en modelos donde se necesita predecir una probabilidad como una salida. Esta función viene definida por siguiente fórmula:

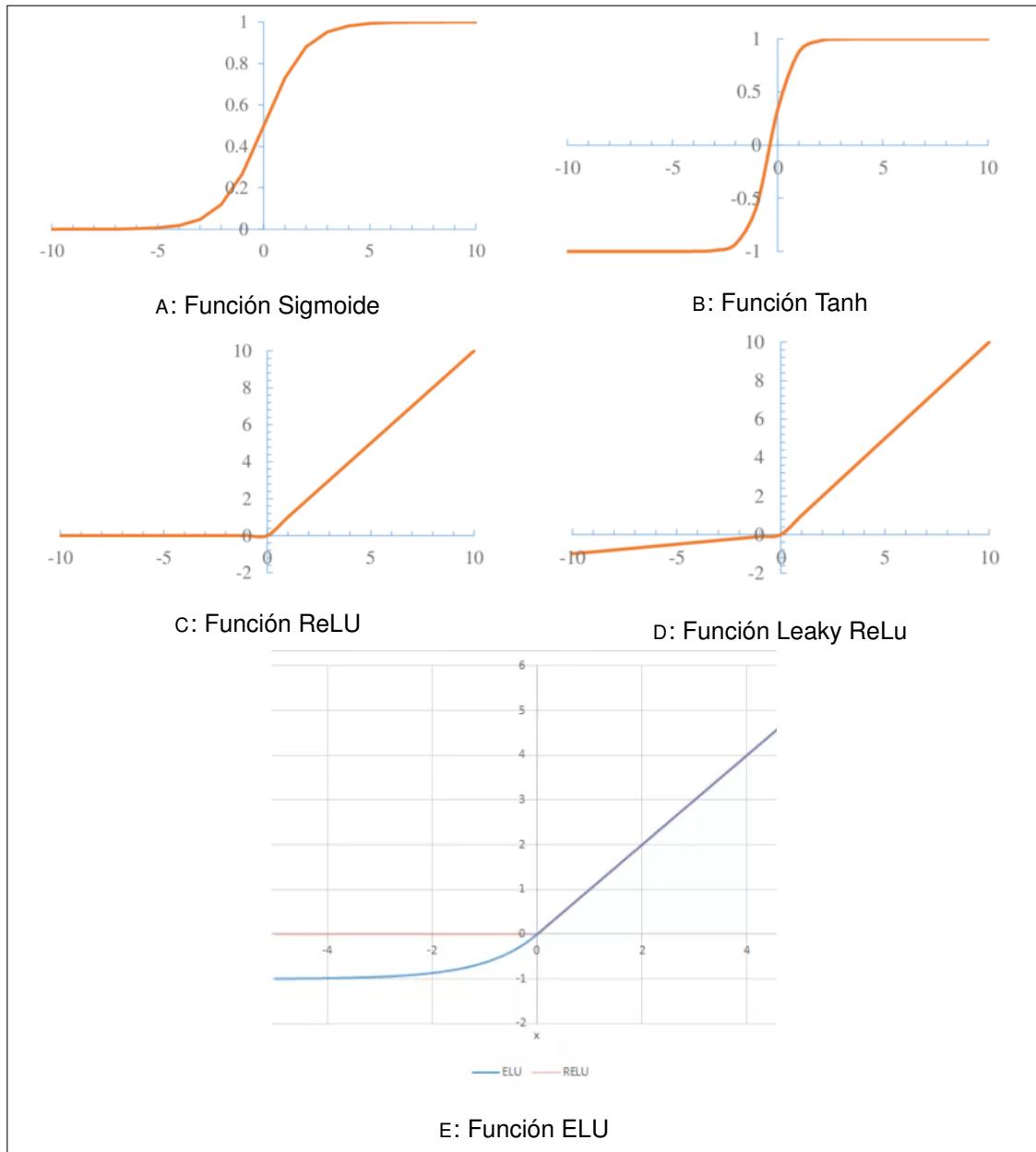


FIGURA 3.3: Funciones de activación (Jing y Guanci, 2018).

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

La función sigmoide se aplicó con éxito en problemas de clasificación binaria, modelado de tareas de regresión logística, así como otros dominios de red neuronal, sin embargo, sufre inconvenientes importantes que incluyen gradientes húmedos agudos durante la propagación hacia atrás desde capas ocultas más profundas a las capas de entrada, saturación de gradiente, convergencia lenta y salida no centrada en cero, lo que hace que las actualizaciones de gradiente se propaguen en diferentes direcciones.

Función de tangente hiperbólica - tanh

Es bastante similar a Sigmoid pero tiene un rendimiento mucho mejor respecto al entrenamiento de redes neuronales multicapa, su naturaleza es no lineal. Esta función está centrada en 0 y su rango se encuentra entre -1 y 1 (Ver Figura 3.3b), por lo tanto, su salida esta definida por:

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

Aunque esta función tenga un mejor rendimiento que la sigmoide, no pudo resolver el problema de gradiente de fuga que tienen las funciones sigmoideas. Una de las principales ventajas de la función tangencial es que produce una salida centrada a cero, lo cual ayuda al proceso de propagación hacia atrás.

Las funciones de tangente se han utilizado principalmente en redes neuronales recurrentes para el procesamiento del lenguaje natural (Dauphin, Fan, Auli, y Grangiera, 2017) y tareas de reconocimiento del habla (Mass, Hannun, y Ng, 2013).

Función de unidad lineal rectificada (Rectified Linear Unit - ReLU)

La función ReLU fue propuesta por Nair y Hinton en 2010, y desde entonces ha sido la función de activación más ampliamente utilizada para aplicaciones de aprendizaje automático con redes neuronales. ReLU es una función de activación de aprendizaje más rápido (Lecun, Bengio, y Hinton, 2015), por lo que demostró ser la función más exitosa y más usada. Esta función ofrece un mejor rendimiento y generalización que las funciones sigmoide y tangente en el aprendizaje con redes neuronales.

ReLU representa una función casi lineal y, por lo tanto, conserva las propiedades de los modelos lineales que lo hace fácil de optimizar, con métodos de descenso de gradiente.

La función de activación de ReLU realiza una operación de umbral para cada elemento de entrada donde los valores inferiores a cero se establecen en cero (Ver Figura 3.3c), por lo que ReLU esta definida por:

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

Esta función rectifica los valores de las entradas inferiores a cero, obligándolos a convertirse en cero, con lo cual elimina el problema de gradiente de fuga observado en los tipos anteriores de función de activación. La función ReLU se ha usado dentro de las unidades ocultas de las redes neuronales.

La principal ventaja de utilizar ReLU es que garantiza un cálculo más rápido, ya que no calcula exponenciales y divisiones, con una velocidad general de cálculo mejorada (Zeiler y cols., 2013). Otra propiedad de ReLU es que introduce la escasez en las unidades ocultas, ya que reduce los valores entre cero y máximo. Sin embargo, ReLU tiene la limitación de que se sobreajusta fácilmente en comparación con la función sigmoidea, aunque se ha adoptado la técnica de abandono para reducir el efecto de sobreajuste de ReLU y las redes rectificadas mejoraron el rendimiento de las redes neuronales.

ReLU tiene una limitación significativa de que a veces es frágil durante el entrenamiento, causando la muerte de algunos de los gradientes. Esto hace que algunas neuronas también estén muertas, para resolver los problemas de neuronas muertas, se propuso la función de activación Leaky ReLU.

Leaky ReLU (LReLU)

Fue propuesta el año 2013 como una función de activación, esta función introduce una pequeña pendiente negativa a ReLU para mantener y mantener vivas las actualizaciones de peso durante el proceso de propagación (Mass y cols., 2013). El parámetro α fue introducido como una solución a los problemas de neuronas muertas de ReLU. Esta función calcula el gradiente con un valor constante muy pequeño para el gradiente negativo α en el rango de 0.01, por lo que LReLU (Ver Figura 3.3d) se calcula como:

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

Función de Unidad Lineal Exponencial (ELU)

La función ELU² tiende a converger el costo a cero más rápido y produce resultados más precisos. A diferencia de otras funciones de activación ELU tiene una constante alfa adicional que debería ser un número positivo.

Es muy similar a ReLU ya que ambas tienen una función identidad para las entradas positivas, sin embargo en las entradas negativas ELU se suaviza lentamente hasta que su salida es igual $-\alpha$ mientras que en ReLU se suaviza bruscamente. La función ELU se calcula según la ecuación 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. Arquitectura de las Redes Neuronales

Una red neuronal regular consiste de una cadena de capas interconectadas de neuronas, estas capas son: una capa de entrada, una o varias capas ocultas y una capa de salida.

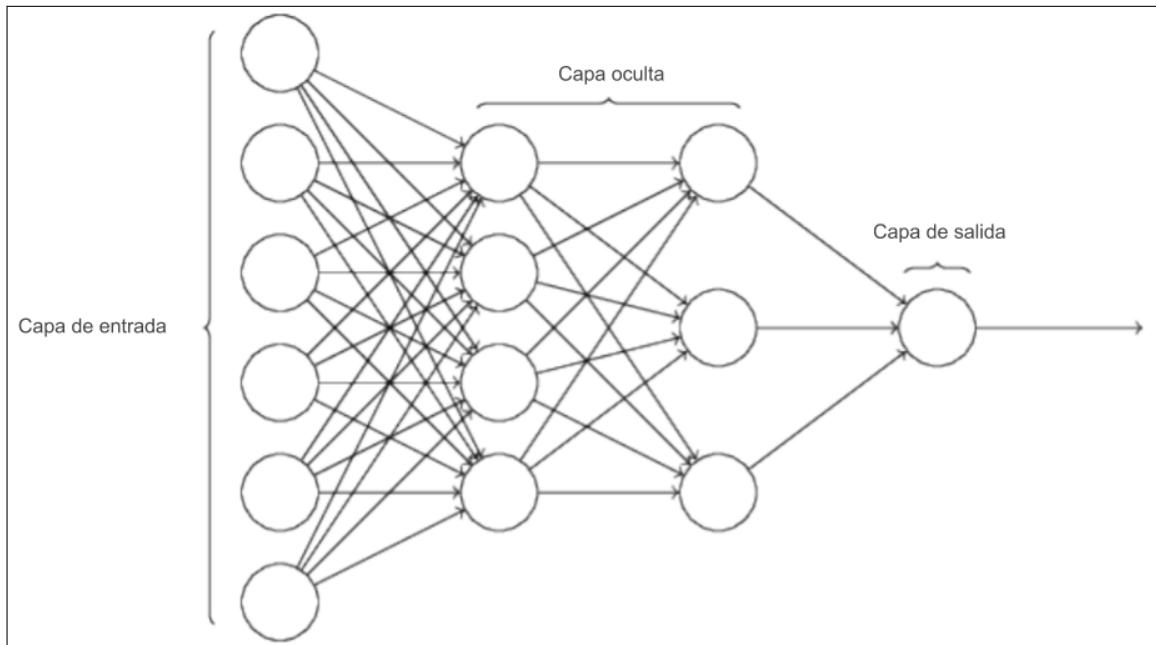


FIGURA 3.4: Arquitectura de una neurona artificial. Reproducido desde (Michael, 2015).

La figura 3.4 es un ejemplo de una red neuronal muy simple; en esta figura la capa más a la izquierda se llama **capa de entrada**, y las neuronas dentro de esta capa se llaman neuronas de

²ELU, Exponential Lineal Unit

entrada. La capa más a la derecha o de **salida** contiene las neuronas de salida. Y por último las dos capas intermedias son las **capas ocultas** de la red neuronal, estas se llaman así debido a que las neuronas de esta capa no son de entrada o de salida; una red neuronal puede tener una o más capas ocultas.

A diferencia del cerebro humano, una Red Neuronal Artificial tiene una estructura predefinida bastante estricta, las conexiones entre neuronas son siempre hacia adelante (feedforward): las conexiones van desde las neuronas de una determinada capa hacia las neuronas de la siguiente capa, es decir, no existen conexiones laterales ni conexiones hacia atrás. Esto significa que una neurona que fue activada en la capa 3 no puede activar a una neurona de la capa 2 o anterior.

3.3.4. Proceso de aprendizaje de las Redes Neuronales

Una característica clave de las redes neuronales es su proceso de aprendizaje iterativo, es decir, cada ejemplo del conjunto de entrenamiento se presenta a la red, uno a la vez, con lo que los pesos asociados con los valores de entrada se ajustan cada vez. Durante esta fase de aprendizaje, la red se entrena ajustando los pesos para predecir una salida correcta para las muestras de entrada.

Las redes neuronales tienen la ventaja de tener una alta tolerancia a los datos ruidosos, como también una alta capacidad para clasificar patrones con los que no han sido entrenados. La técnica de entrenamiento de redes neuronales más popular es el **algoritmo de retropropagación** (Backpropagation).

Una vez que se define la estructura de una red para una aplicación en particular, está lista para ser capacitada. Para comenzar este proceso, los pesos iniciales se eligen al azar, para luego proceder con el entrenamiento (aprendizaje).

Backpropagation

Una red neuronal propaga la señal de los datos de entrada hacia adelante a través de sus parámetros en el momento de la decisión; para luego propagar hacia atrás la información sobre el error, para que se pueda alterar los parámetros. Esto sucede siguiendo los siguientes pasos:

- La red adivina los datos de salida, usando sus parámetros.
- La red mide su precisión con una función de pérdida.
- El error es propagado hacia atrás para ajustar los parámetros equivocados.

Por lo tanto se puede decir que el algoritmo de Backpropagation toma el error asociado con una suposición errónea por parte de la red neuronal, y usa ese error para ajustar los parámetros de la red neuronal en la dirección que genere un menor error.

3.4. Tipos de Redes Neuronales

3.4.1. Autoencoders

Un Autoencoder es una Red Neuronal Artificial usada para aprendizaje automático no supervisado, esta entrenada para reconstruir sus propias entradas, es decir, predecir el valor de la salida \hat{x} dada una entrada x vía una capa oculta h , ver Figura 3.5. Los autoencoders suelen ser usados para reducción de dimensionalidad y aprendizaje de características.

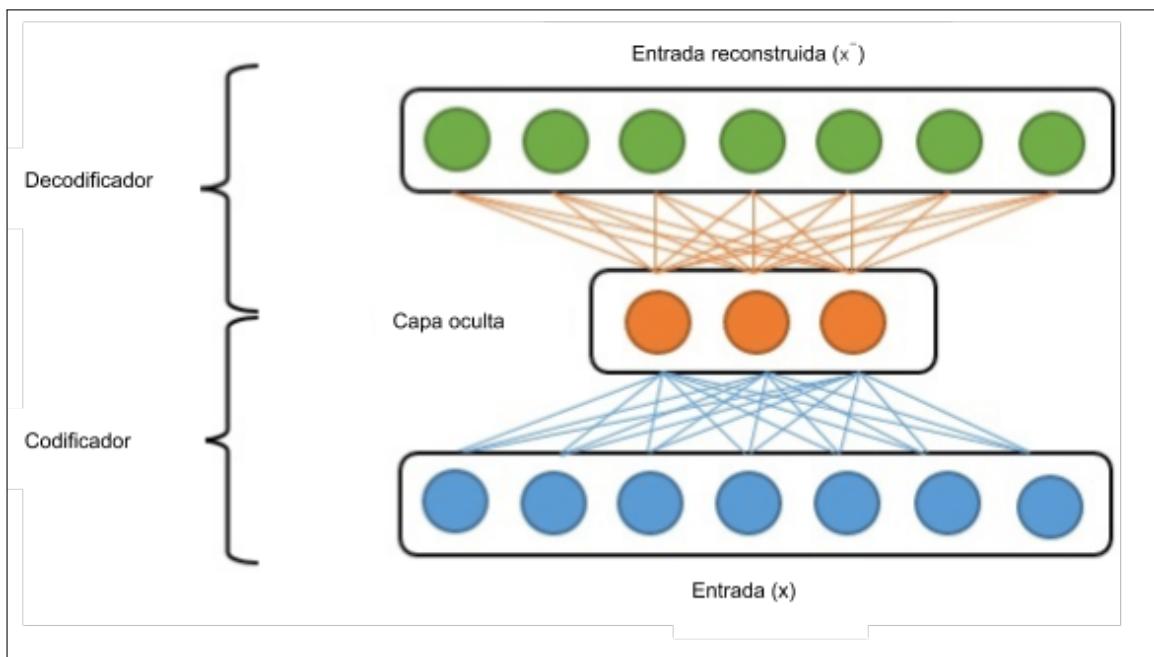


FIGURA 3.5: Gráfico de un Autoencoder (Elaboración propia).

Los autoencoders están compuestos de dos partes: el codificador y decodificador. El codificador aprende una representación compresa de los datos de entrada, este puede ser definido con la función de codificación $h = \text{encoder}(x)$, el cual es definido por una función lineal o no lineal. Si la función del codificador es no lineal el autoencoder será capaz de aprender más características que un PCA lineal. El propósito del decodificador es reconstruir su propia entrada vía la función de decodificación, $\hat{x} = \text{decoder}(h)$.

La diferencia entre la entrada y la entrada reconstruida es el **error de reconstrucción**. Durante el entrenamiento, el autoencoder minimiza el error de reconstrucción como una función

objetivo. Los autoencoders se usan a menudo para la generación de datos como modelos generativos. El decodificador de un autoencoder puede generar una salida dada una representación comprimida asignada artificialmente.

3.4.2. Redes neuronales convolucionales

Para algunos tipos de datos, específicamente para imágenes, las redes neuronales convolucionales no están bien adaptadas; lo cual conlleva que en el estudio Lecun y cols. (1998) propongan las redes neuronales convolucionales (CNN³) para solucionar ese problema. Las CNN han revolucionado el procesamiento de imágenes y han eliminado la extracción manual de características. Una CNN actúa directamente en matrices, o incluso en tensores para imágenes con tres canales de color RGB; por lo que actualmente, las CNN se usan ampliamente para la clasificación de imágenes, reconocimiento de objetos, reconocimiento de rostros, entre otros.

Las CNN no solo brindan un mejor rendimiento en comparación con otros algoritmos de detección; sino que incluso en algunos casos superan a los humanos, como por ejemplo en la clasificación de objetos en categorías específicas como la raza particular de un perro o una especie de ave (Russakovsky, 2014).

Al apilar múltiples y diferentes capas en una CNN, se crean arquitecturas complejas para los problemas de clasificación. Los cuatro tipos de capas que son más comunes son: capa de convolución, capa de agrupación/submuestreo, capa no lineal y capa completamente conectada. En la Figura 3.6 se puede ver un ejemplo de una CNN, donde la primera y tercera capa son capas convolucionales, la segunda y la cuarta son capas de submuestreo y por último la quinta y la sexta capa con capas completamente conectadas.

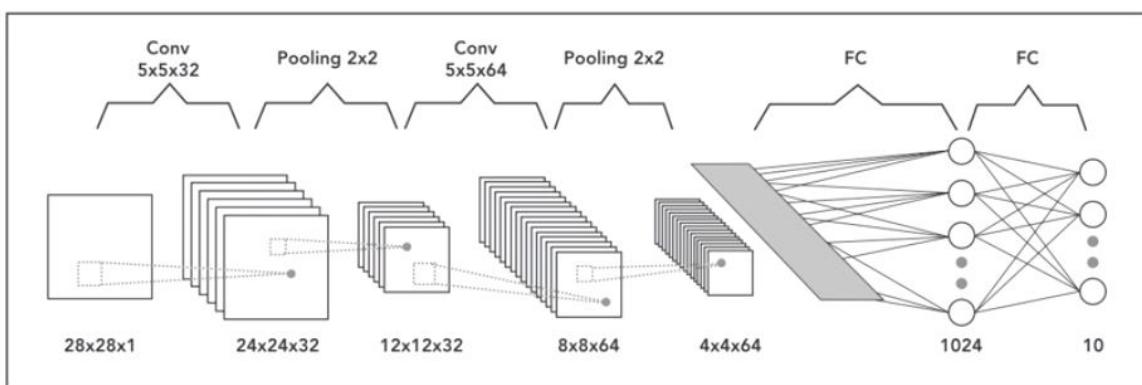


FIGURA 3.6: Arquitectura de una Red Neuronal Convolucional (CNN) (Muhammad, s.f.).

³CNN, Convolutional Neural Network

3.4.3. Redes neuronales recurrentes

Para entender la importancia de las series de tiempo se puede tomar la siguiente analogía, los seres humanos no comienzan a pensar desde cero cada segundo, por lo que al leer un documento se comprende cada palabra basándose en la comprensión de las palabras anteriores, es decir, no se elimina todo y se empieza a pensar de cero cada vez, dada ésta afirmación se puede decir que los pensamientos de los seres humanos tienen persistencia.

Las redes neuronales tradicionales no tienen persistencia de los datos, lo que para algunos problemas en concreto, incluyendo el que se aborda en este trabajo, es una gran deficiencia. Con el fin de resolver este tipo de problemas aparecen las Redes Neuronales Recurrentes (RNN⁴), las cuales son un tipo de red neuronal artificial propuesta en los años 80 (Rumelhart, Hinton, y Williams, 1986; Elman, 1990; Werbos, 1988) diseñada para reconocer patrones en secuencias de datos, como texto, genomas, escritura a mano, datos de series de tiempo numéricos que emanan de sensores, entre otros.

Las RNN son una familia particular de redes neuronales donde la red contiene una o más conexiones de retroalimentación, de modo que la activación de un grupo de neuronas puede fluir en un bucle. Esta propiedad hace que el modelo pueda retener información sobre el pasado, lo que le permite descubrir correlaciones temporales entre eventos que están muy lejos unos de otros en los datos.

Las RNN tienen una cierta memoria de lo que sucedió anteriormente en una secuencia de datos, esto ayuda al sistema a ganar contexto de los datos. Teóricamente se dice que las RNN tiene memoria infinita, es decir, este tipo de redes tienen la capacidad de mirar hacia atrás indefinidamente; sin embargo en la práctica sólo se puede mirar atrás unos últimos pasos.

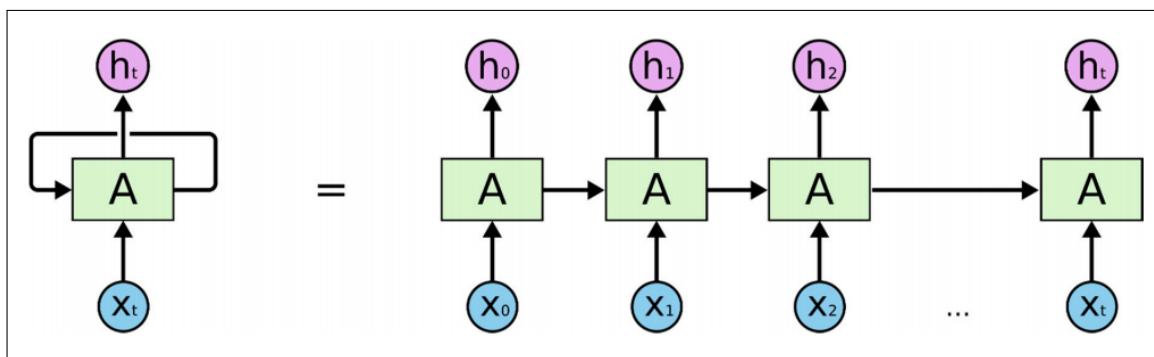


FIGURA 3.7: Procesamiento secuencial en una red neuronal recurrente (RNN)
(Olah, s.f.).

En la Figura 3.7 se ilustra una simple RNN con una unidad de entrada, una unidad de salida y una unidad oculta recurrente expandida en una red completa, donde x_t es la entrada en el

⁴RNN, Recurrent Neural Network

paso de tiempo t y y_t es la salida en el paso de tiempo t . Por otra parte en el proceso de entrenamiento las RNN usan el algoritmo de Backpropagation a través del tiempo (BPTT⁵). El proceso BPTT utiliza un enfoque de trabajo hacia atrás, capa por capa, de la salida final de la red, ajustando los pesos de cada unidad de acuerdo con la porción calculada de la unidad del error de la salida total. La repetición de los bucles de información da como resultado grandes actualizaciones de los pesos del modelo de red neuronal y conduce a una red inestable debido a la acumulación de gradientes de error durante el proceso de actualización. Por lo tanto, BPTT no es lo suficientemente eficiente como para aprender un patrón de dependencia a largo plazo debido a la desaparición del gradiente y la explosión de los problemas del gradiente (Bengio, Simard, y Frasconi, 1994). Para superar los problemas de gradiente de desaparición y explosión que tienen las RNN estándar, se pueden usar LSTM y GRU (Pascanu, Mikolov, y Bengio, 2013).

3.4.4. LSTM

LSTM⁶ es una evolución de RNN, fue introducida por Hochreiter y Schmidhuber en (1997) para abordar los problemas de las RNN estándar que fueron mencionados antes, agregando interacciones adicionales por módulo (o celda). Los LSTM son un tipo especial de RNN, capaces de aprender dependencias a largo plazo y recordar información por períodos prolongados por defecto.

El modelo LSTM está organizado en forma de estructura de cadena. Sin embargo, el módulo de repetición tiene una estructura diferente. En lugar de una sola red neuronal como un RNN estándar, tiene cuatro capas interactivas con un método único de comunicación. La estructura de la red neuronal LSTM se muestra en la Figura 3.8.

Una típica red LSTM está compuesta por bloques de memoria llamados celdas. Dos estados se transfieren a la siguiente celda, el estado de la celda y el estado oculto. El **estado de la celda** es la cadena principal del flujo de datos, lo que permite que los datos fluyan hacia adelante, esencialmente, sin cambios. Sin embargo, pueden ocurrir algunas transformaciones lineales. Los datos pueden agregar o eliminar el estado de la celda a través de puertas sigmoides.

Una puerta es similar a una capa o una serie de operaciones matriciales, la cual contiene diferentes pesos individuales. Los LSTM están diseñados para evitar el problema de dependencia a largo plazo porque utilizan puertas para controlar el proceso de memorización.

3.4.5. GRU

GRU⁷ fue diseñado por primera vez por Kyunghyun Cho en (2014a). Esta estructura de RNN sólo contiene dos puertas. La puerta de actualización controla la información que fluye hacia

⁵BPTT, Backpropagation Through The Time

⁶LSTM, Long Short-Term Memory

⁷GRU, Gated Recurrent Unit

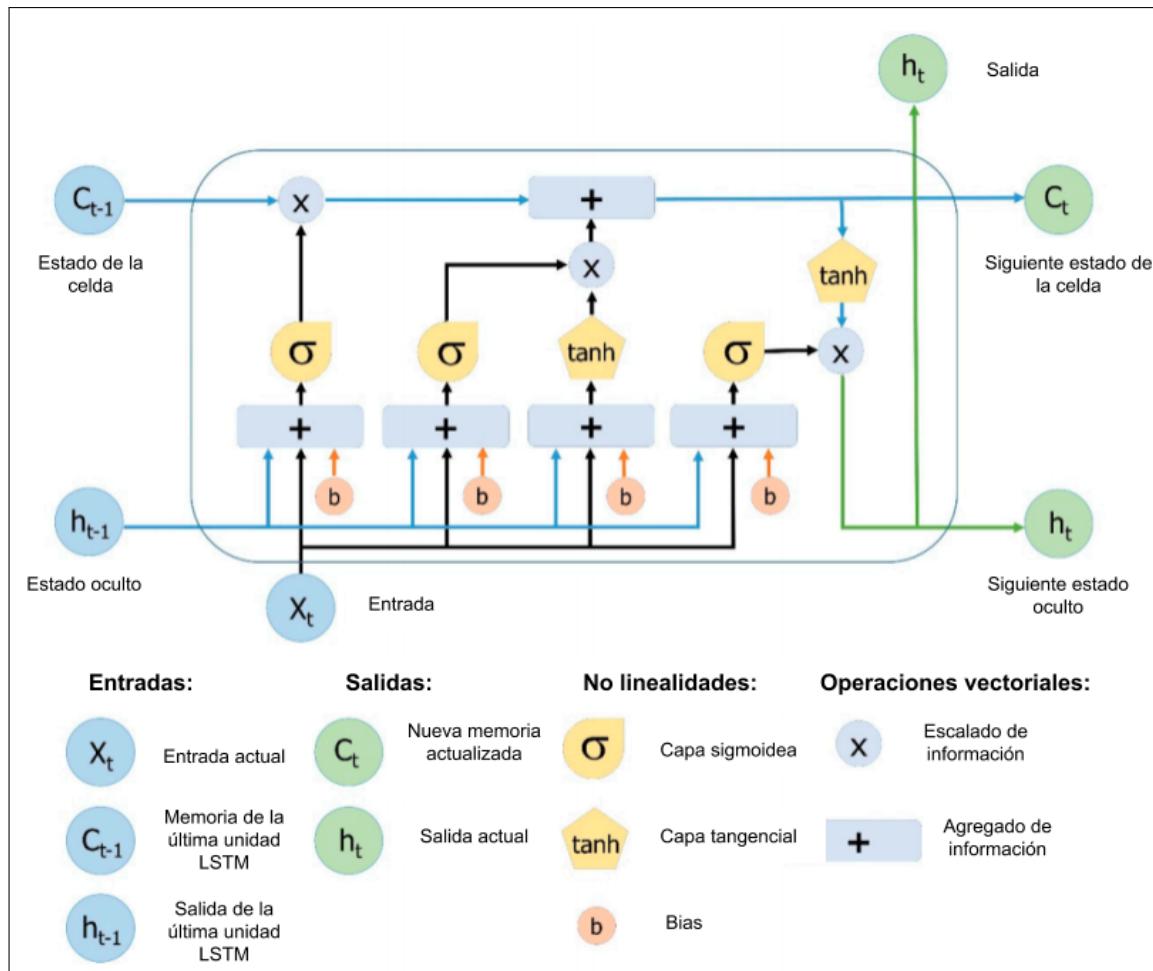


FIGURA 3.8: Estructura de LSTM. Reproducido desde Yan (Yan, s.f.).

la memoria, mientras que la puerta de reinicio controla la información que fluye fuera de la memoria. De manera similar a la LSTM, GRU tiene unidades de compuerta que modulan el flujo de información dentro de la unidad; sin embargo, sin tener una celda de memoria separada. Se podría decir que GRU es una variante de RNN un poco más simplificada que a menudo ofrece un rendimiento comparable a LSTM y es significativamente más rápido de calcular.

En resumen, las GRU tienen las siguientes dos características distintivas:

- **Las puertas de reinicio** ayudan a capturar dependencias a corto plazo en series de tiempo.
- **Las puertas de actualización** ayudan a capturar dependencias a largo plazo en series de tiempo.

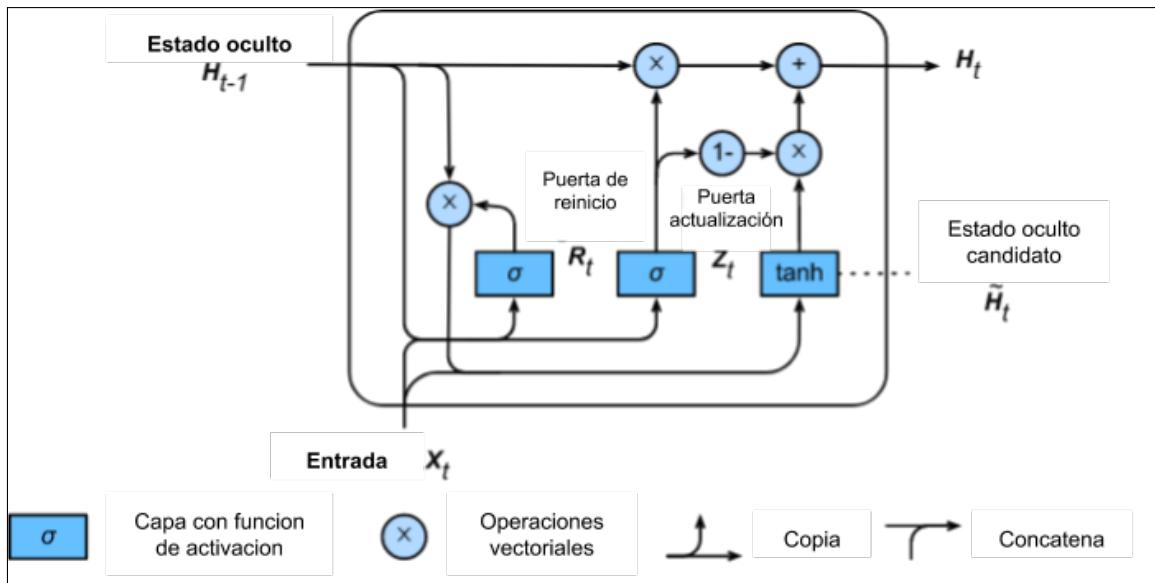


FIGURA 3.9: Estructura de GRU. Reproducido desde (Zhang y cols., 2019).

3.5. Técnicas de detección de anomalías

Existen varios enfoques disponibles para la detección de anomalías, en esta sección se describirá algunos de los algoritmos más usados.

3.5.1. One-Class SVM

One-Class SVM⁸ (OC-SVM) es un enfoque ampliamente utilizado para descubrir anomalías de forma no supervisada (Schölkopf y Smola, 2002). Las OC-SVM son una de las técnicas de aprendizaje semi-supervisado más utilizadas, debido a que da buenos resultados incluso para conjuntos de datos de alta dimensión. Sin embargo, este algoritmo tiene la desventaja de que requiere mucho tiempo y memoria en la práctica y su complejidad crece de forma cuadrática con el número de registros.

Este algoritmo sólo se entrena con ejemplos positivos (clases normales). La idea general de éste algoritmo es transformar el espacio de atributos y dibujar un hiperplano divisorio para que las observaciones se encuentren lo más lejos posible del origen, como se puede observar en la Figura 3.10.

Como resultado, se obtiene un margen, en un lado del cual las observaciones de la muestra de entrenamiento se agrupan lo más densamente posible (observaciones normales $\hat{y}_i = 1$), y en

⁸SVM, Support Vector Machine

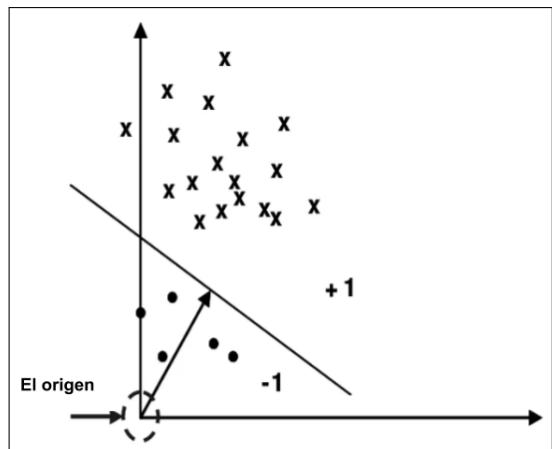


FIGURA 3.10: One-Class SVM. Reproducido desde (Alashwal y cols., 2006)

el otro, se encuentran los valores anormales ($\hat{y}_i = -1$), no similares a lo que el algoritmo vió durante el entrenamiento.

3.5.2. Isolation Forest

Este modelo se usa en un escenario similar al SVM de una clase, específicamente en un entorno no supervisado. El bosque de aislamiento adopta un enfoque diferente del SVM de una clase, ya que en lugar de agrupar datos normales, trata de aislar los datos anómalos.

El componente básico del Isolation Forest (Bosque de aislamiento) es el árbol de aislamiento, que es un árbol binario simple donde en cada nodo T_i tanto la característica como el umbral para la regla de división se seleccionan aleatoriamente. Un nodo existente deja de generar hijos si y solo si solo hay un ejemplo siguiendo la regla de división para esa ruta específica (lo que significa que el ejemplo se ha aislado) o se ha alcanzado una altura máxima. Esto significa que al final del proceso de capacitación tendremos un árbol de clasificación aleatorio completamente sobreajustado, que puede usarse para fines de detección de anomalías. La intuición principal de este algoritmo es que si un ejemplo es anómalo, se aislará después de algunos cortes en el espacio de características, lo que se traduce en tener una baja altura en el árbol de aislamiento.

A diferencia de otros métodos como la agrupación o clasificación, los bosques de aislamiento no aprenden un perfil de lo que es normal, sino que atacan directamente las anomalías. No se utiliza métrica de distancia en este algoritmo lo cual ahorra tiempo en los cálculos; por lo que los bosques de aislamiento tienen una complejidad temporal lineal.

En la Figura 3.11 se presenta una comparación de la capacidad de los algoritmos Isolation Forest y One-Class SVM para hacer frente a diferentes conjuntos de datos de dos dimensiones, con el objetivo de dar cierta intuición acerca del comportamiento de estos algoritmos.

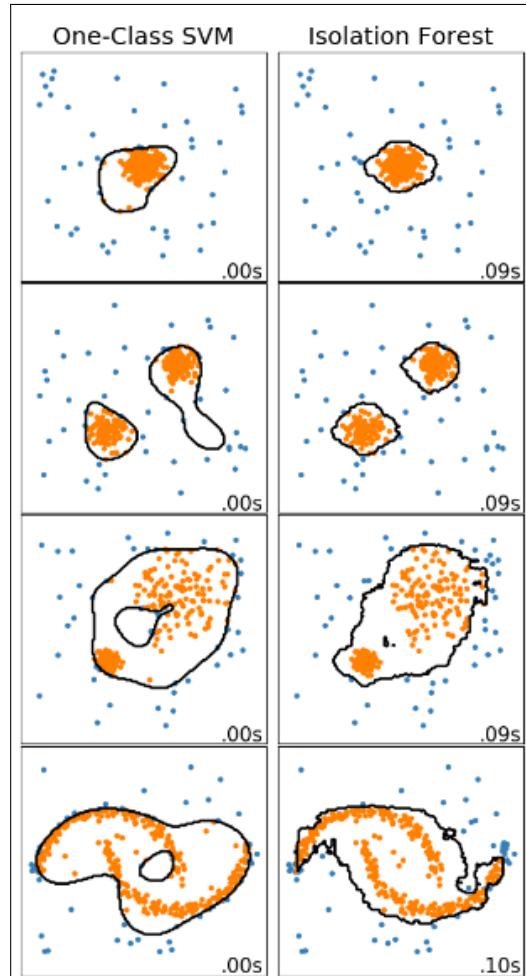


FIGURA 3.11: Comparacion del rendimiento entre los algoritmos One-Clas SVM e Isolation Forest. Reproducido desde (0.22, s.f.).

3.5.3. Autoencoders

Hoy en día, los autoencoders se han utilizado ampliamente en la clasificación de imágenes, traducción automática y procesamiento de voz; esto debido a su capacidad de compresión de datos sin supervisión. Hasta donde se sabe, Hawkins y cols. (2002) y Williams y Baxter (2002) fueron los primeros que propusieron autocodificadores para la detección de anomalías. Desde entonces, la capacidad de los autoencoders para detectar valores atípicos se demostró en diferentes dominios como por ejemplo la detección de anomalías en rayos X.

El método tradicional de detección de anomalías basada en autoencoder se basa principalmente en el error de reconstrucción, considerando como anomalías aquellas muestras que presentan un alto error de reconstrucción. En la fase de entrenamiento, solo se usan datos normales para entrenar el autoencoder, con el objetivo de minimizar el error de reconstrucción, de modo que el autoencoder pueda reconocer las características de los datos normales. En la fase de prueba, el autoencoder entrenado podrá reconstruir datos normales con pequeños errores de reconstrucción, pero fallarán con datos anómalos que el autoencoder no ha encontrado antes y, por lo tanto, tienen errores de reconstrucción relativamente más altos en comparación a los datos normales. Por lo tanto, al comparar si el puntaje de reconstrucción de una anomalía está por encima de un umbral (threshold) predefinido, el autoencoder determinará si los datos presentados para la prueba son anómalos (Guo y cols., 2018) (Ver Figura 3.12). La ecuación 3.8 se observa como esta técnica determina lo que es una anomalía y lo que no lo es; donde S_z representa el error de reconstrucción.

$$C(z) = \begin{cases} \text{Comportamiento normal} & \text{si } S_z \leq \text{Umbral} \\ \text{Anomalía} & \text{si } S_z > \text{Umbral} \end{cases} \quad (3.8)$$

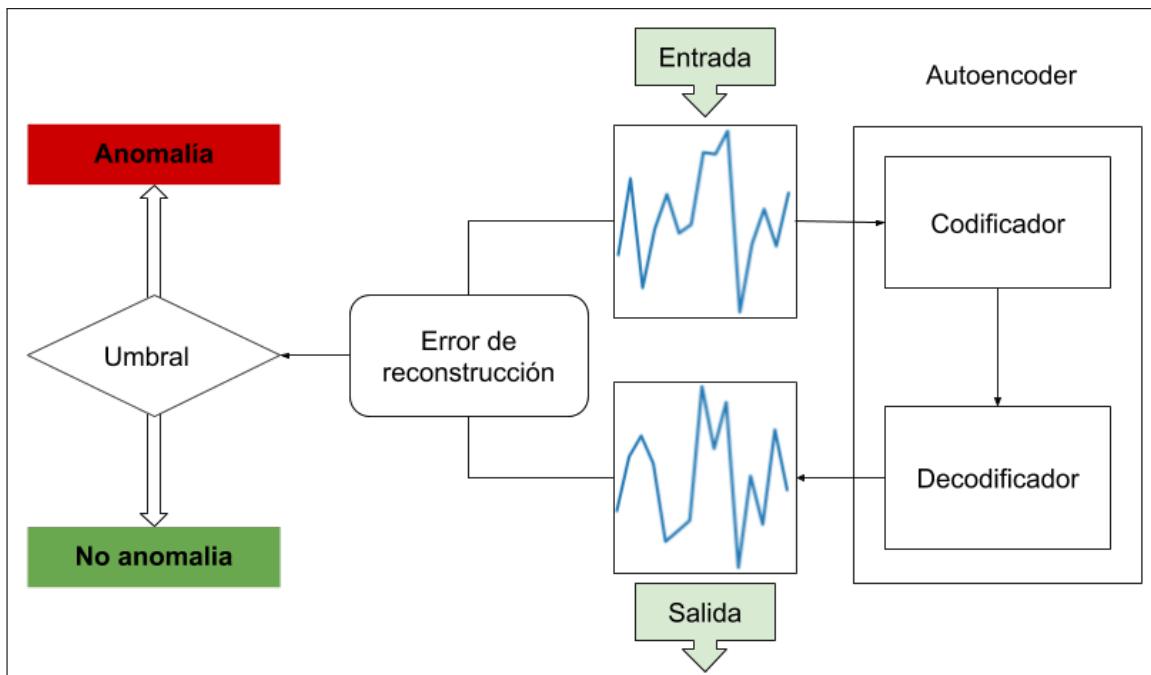


FIGURA 3.12: Detección de anomalías con autoencoder (Elaboración propia).

3.6. Métricas de evaluación

Evaluar los algoritmos de aprendizaje automático es una parte esencial para cualquier proyecto, debido a que un modelo puede brindar resultados satisfactorios cuando es evaluado con una

métrica; pero puede dar un resultado deficiente cuando se utiliza otra métrica. Comúnmente se usa la precisión de clasificación para medir el rendimiento de un modelo, sin embargo, no es suficiente para juzgar un modelo. A continuación se cubrirá diferentes tipos de métricas de evaluación.

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 (AUC⁹) 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 (TPR¹⁰), 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 (TNR¹¹), 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

⁹AUC, Area Under Curve

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

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

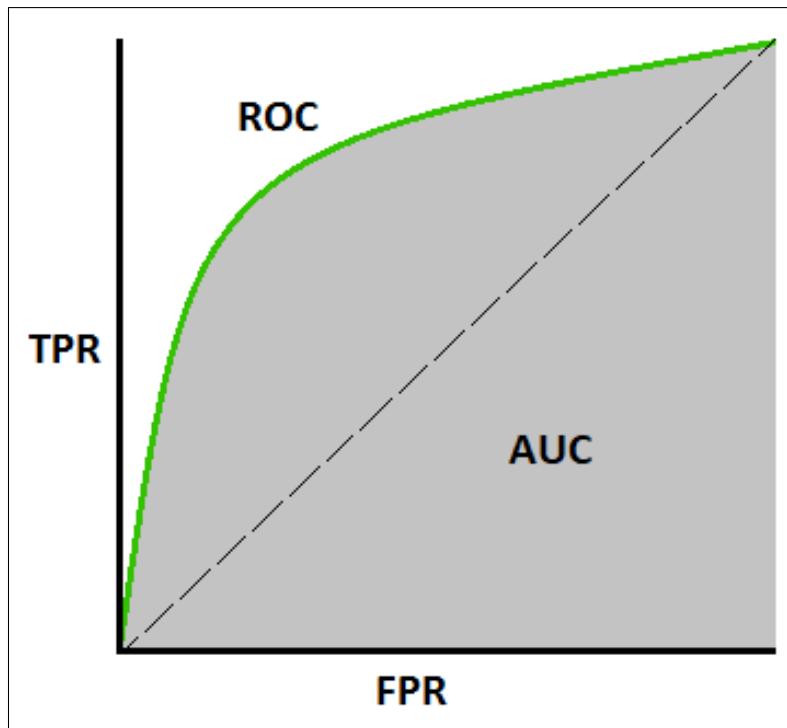


FIGURA 3.13: Ejemplo de un curva AUC-ROC (Özler, s.f.).

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)

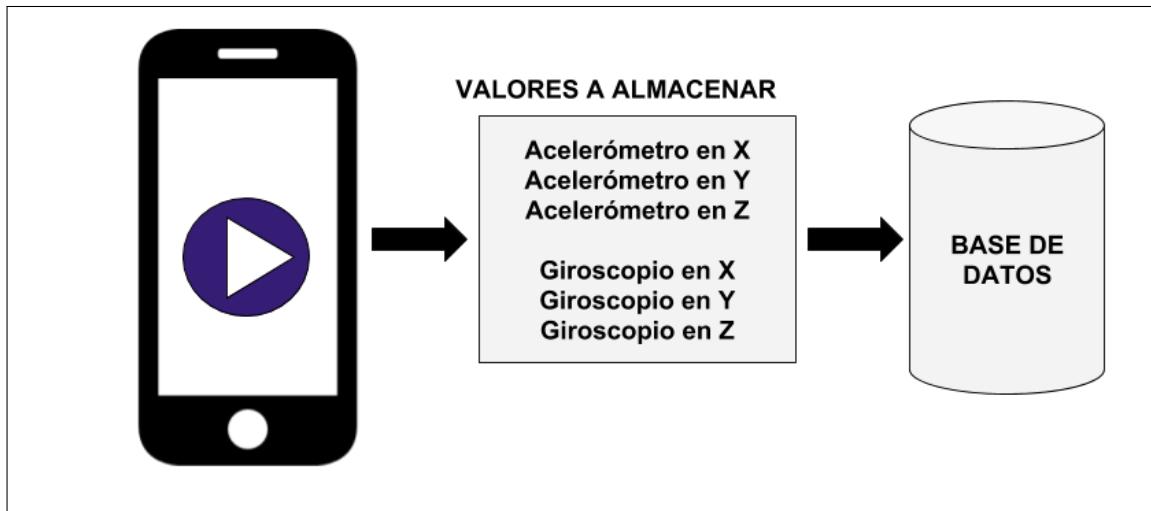


FIGURA 4.1: Recolección de datos, con intervalo de un segundo (Elaboración propia).

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.

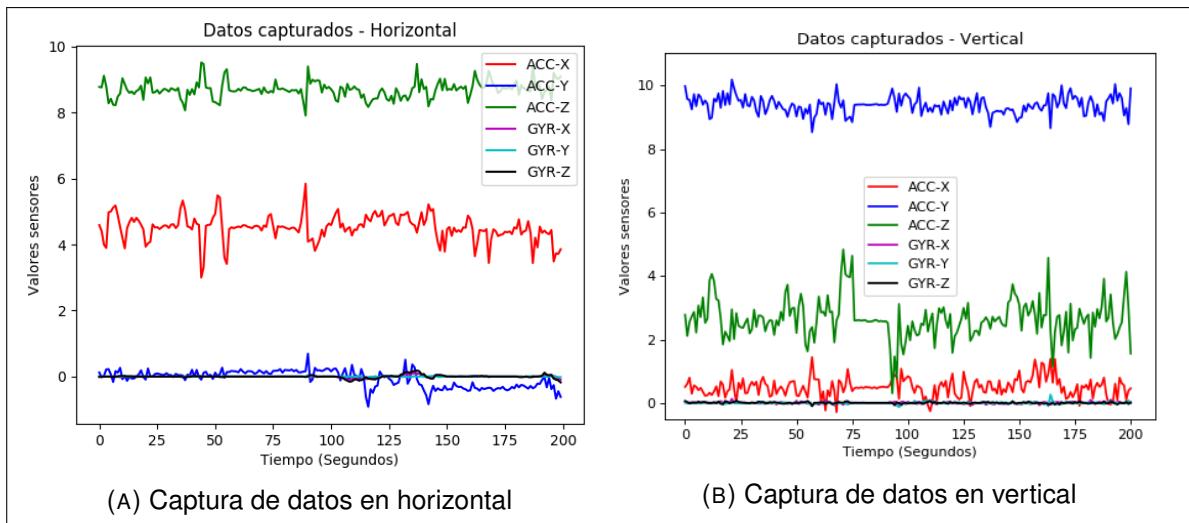


FIGURA 4.4: Gráfica de los sensores capturados en diferentes posiciones (Ela-
boración propia).

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.

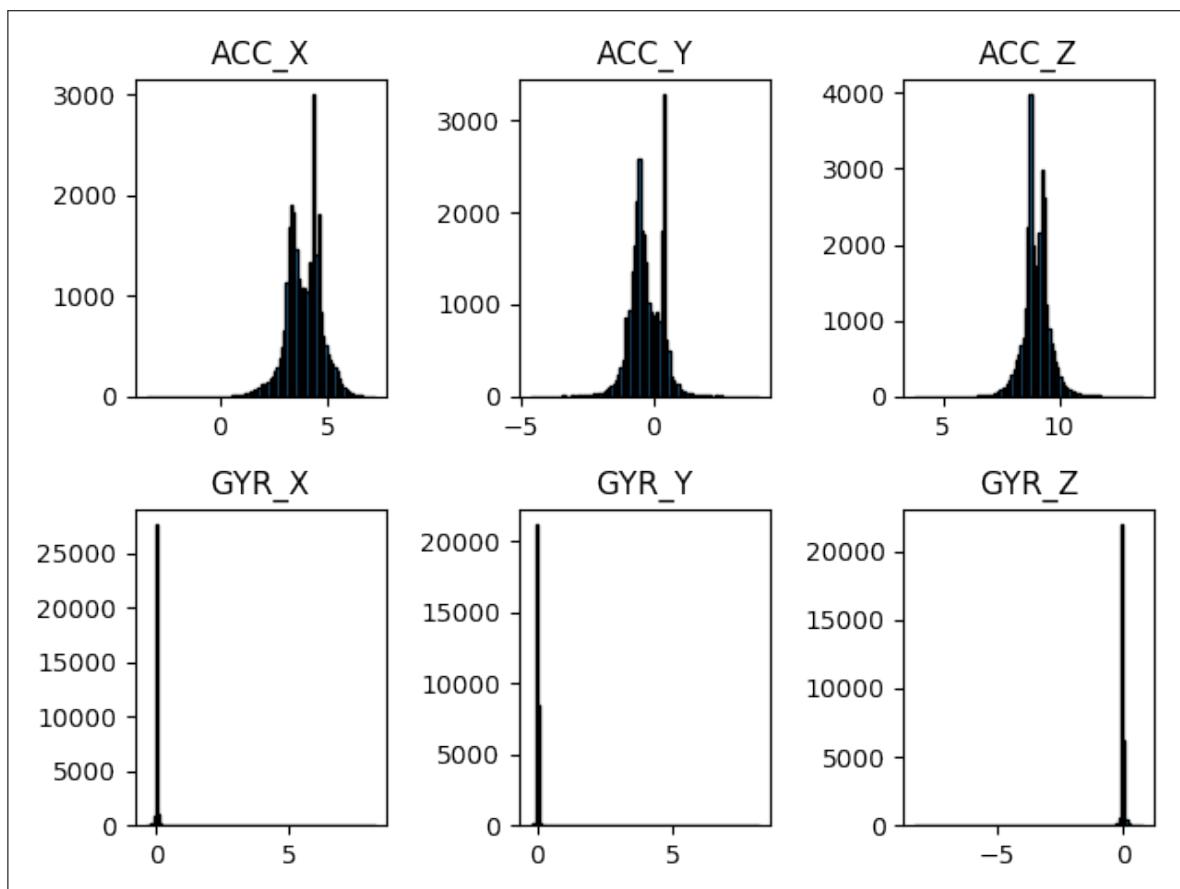


FIGURA 4.5: Histograma de frecuencias del conjunto de datos (Elaboración propia).

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.

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

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.

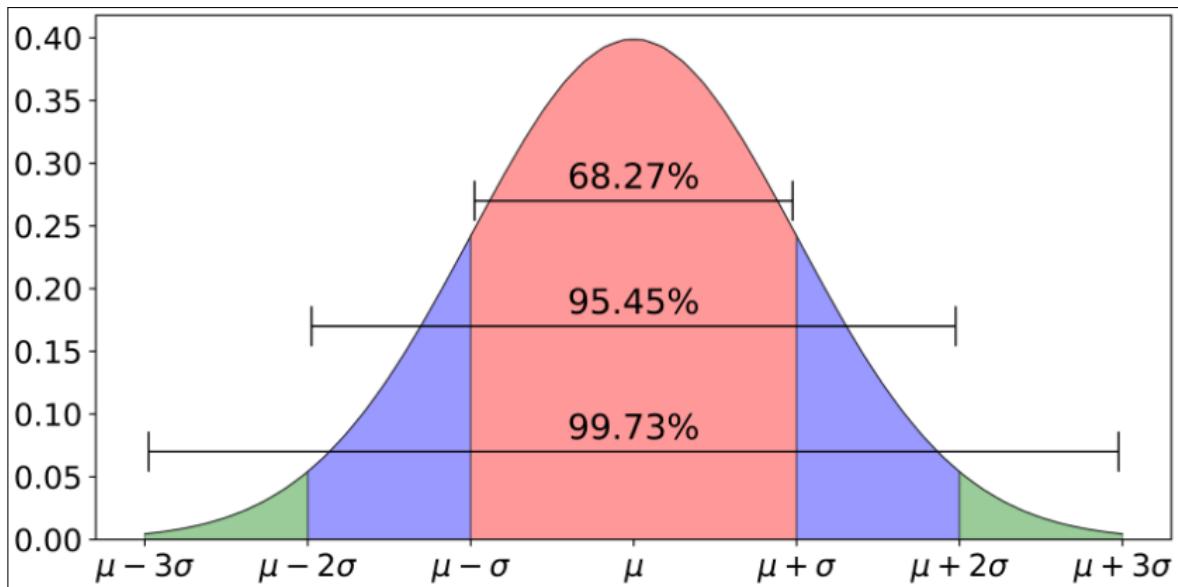


FIGURA 4.7: Regla 68-95-99.7 (Galarnyk, s.f.).

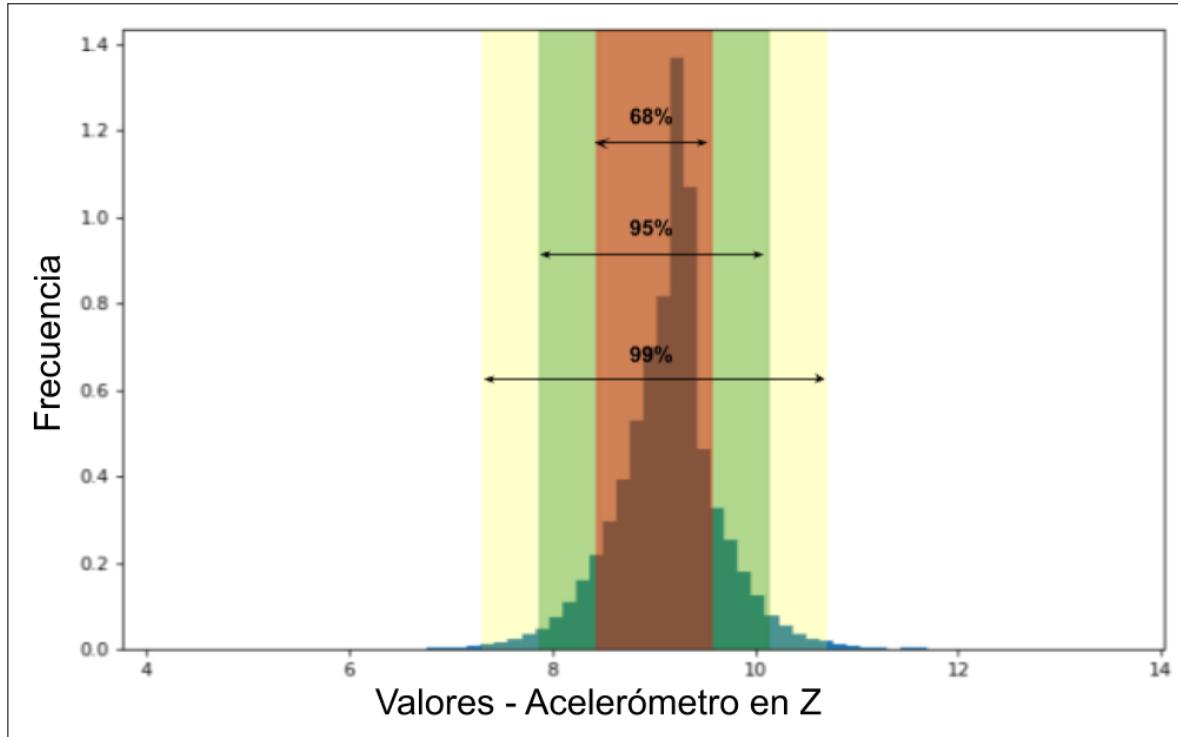


FIGURA 4.8: Regla 68-95-99.7 aplicada a los valores de los sensores del acelerómetro en Z (Elaboración propia).

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, KNN¹, K-Means², 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:

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

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

²K-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 principales³) 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 entrenamiento⁴, desarrollo⁵ y prueba⁶, 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.

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

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

⁵**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.

⁶**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.

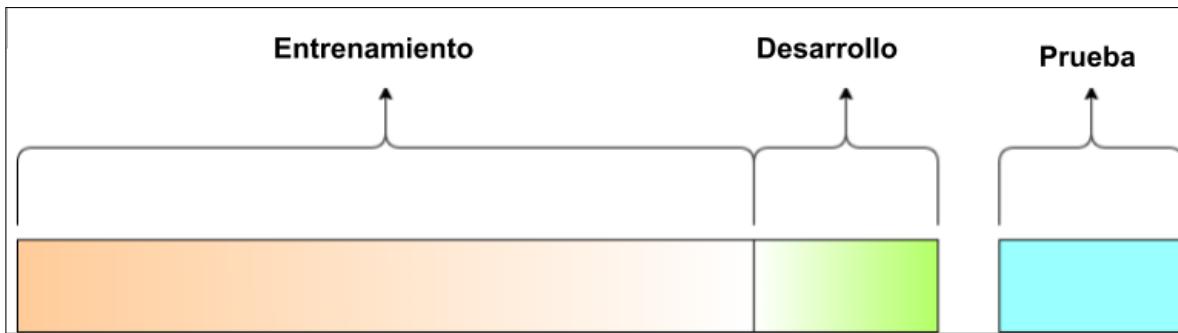


FIGURA 4.9: División del conjunto de entrenamiento (Elaboración propia).

| | | |
|---------------------------|-------|-------|
| 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.

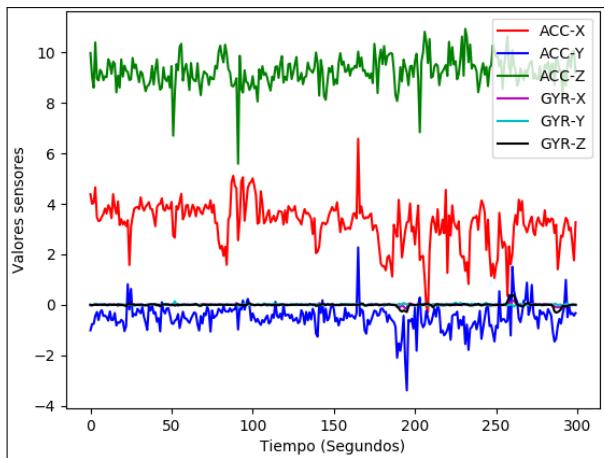


FIGURA 4.10: Visualización de los parámetros de conducción capturados (Elaboración propia).

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.

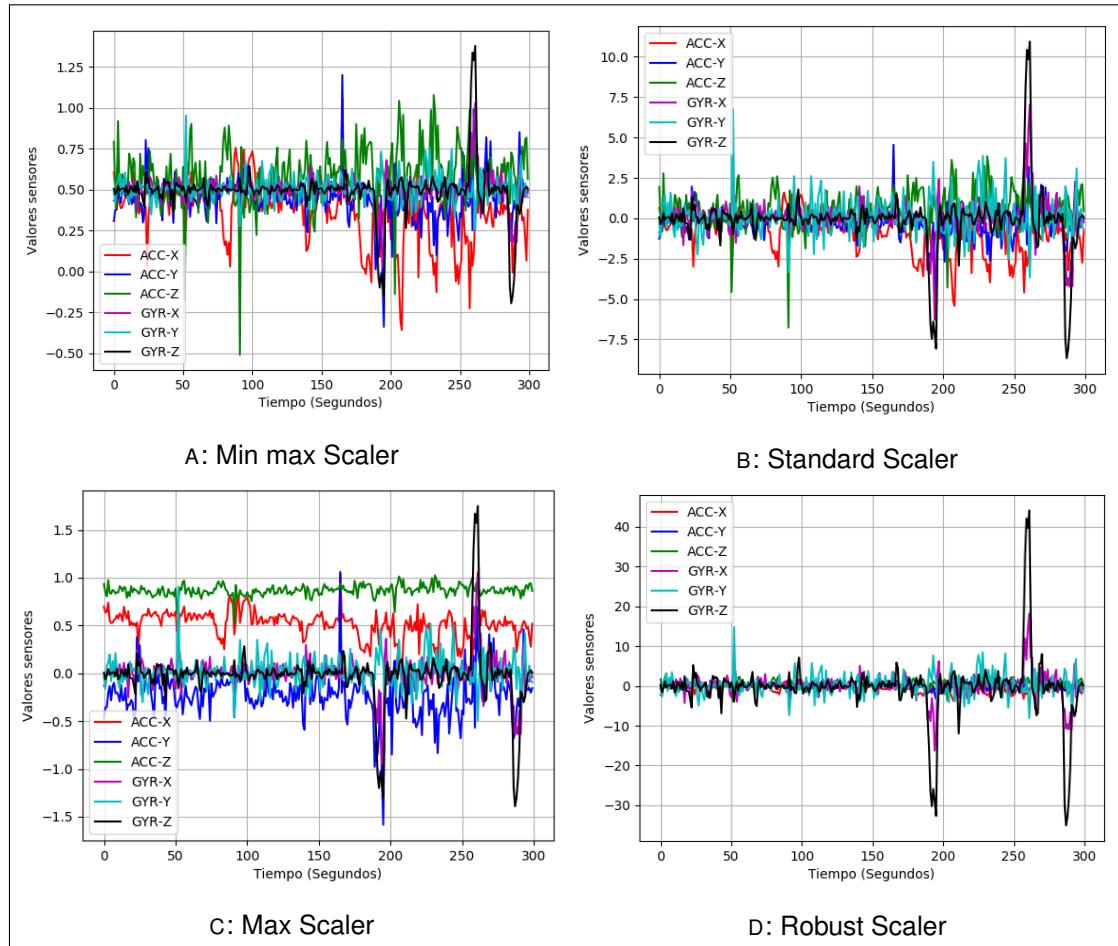


FIGURA 4.11: Gráfica resultante de aplicar diferentes tipos de normalizaciones a un conjunto de datos (Elaboración propia).

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.

| | | ACC X | ACC Y | ACC Z | GYR X | GYR Y | GYR Z |
|--------------|------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 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.

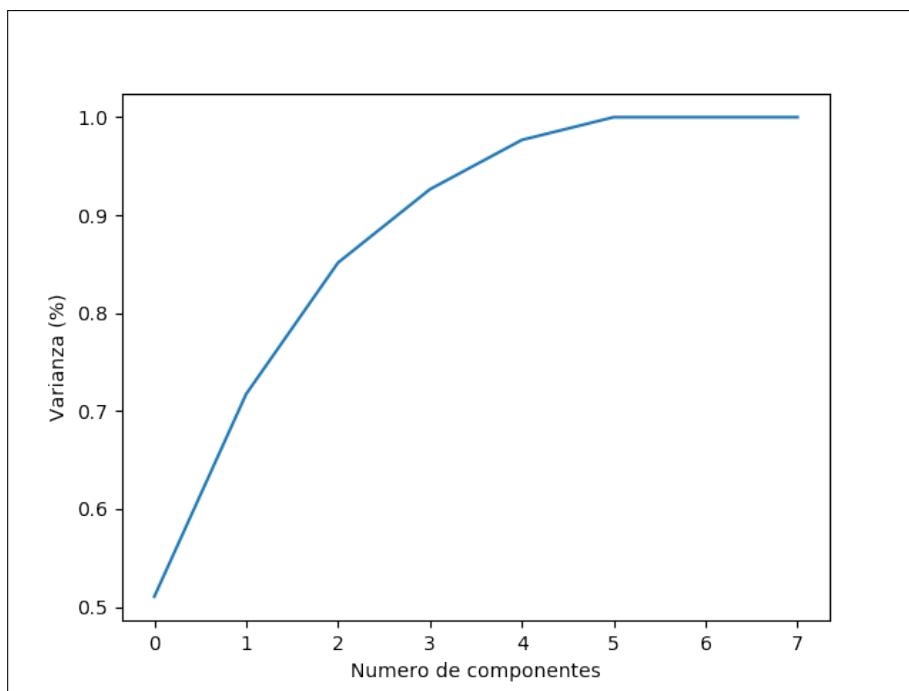


FIGURA 4.12: Gráfico de la varianza vs. el número de componentes (Elaboración propia).

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.

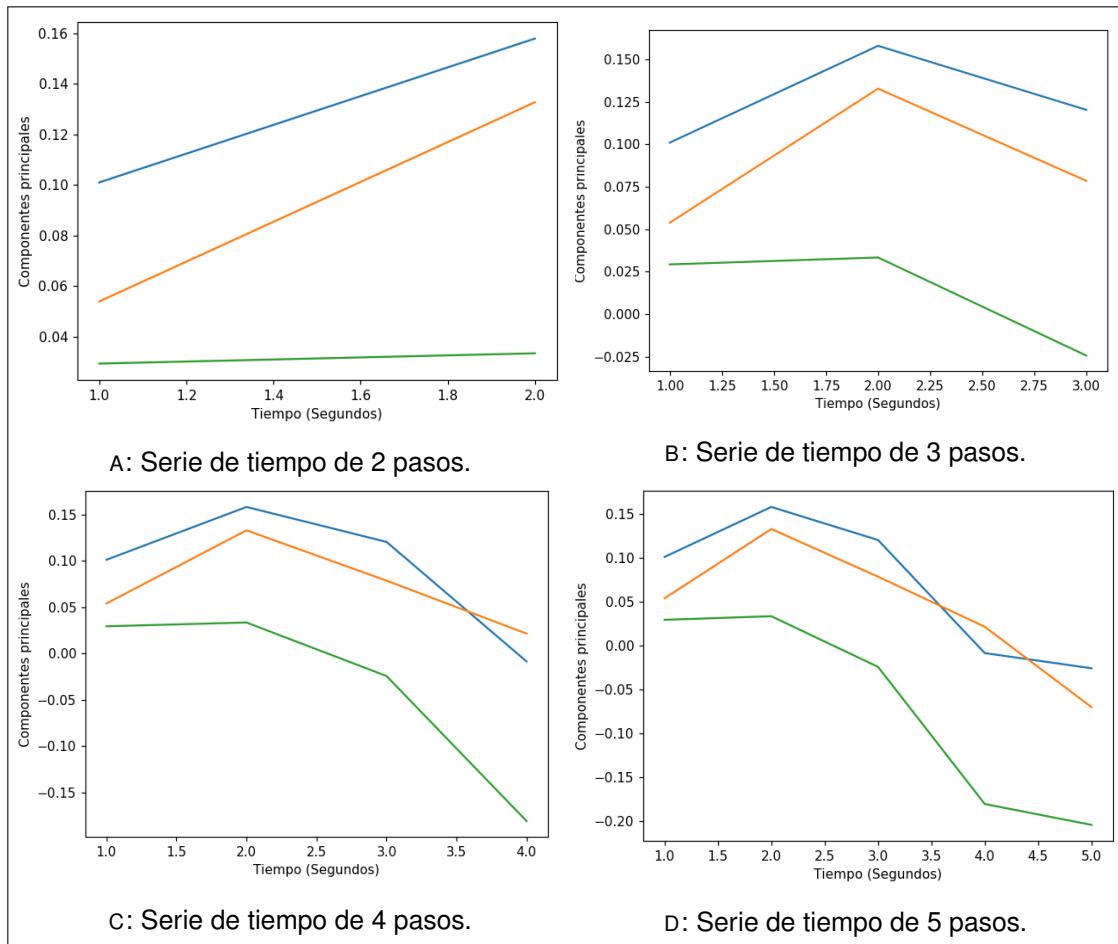


FIGURA 5.1: Gráfica resultante de diferentes tamaños de series de tiempo (Ela-
boración propia).

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**.

| Red | Precisión | Loss | Tiempo ejecución |
|------------|--------------------|----------------------|-------------------------|
| 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

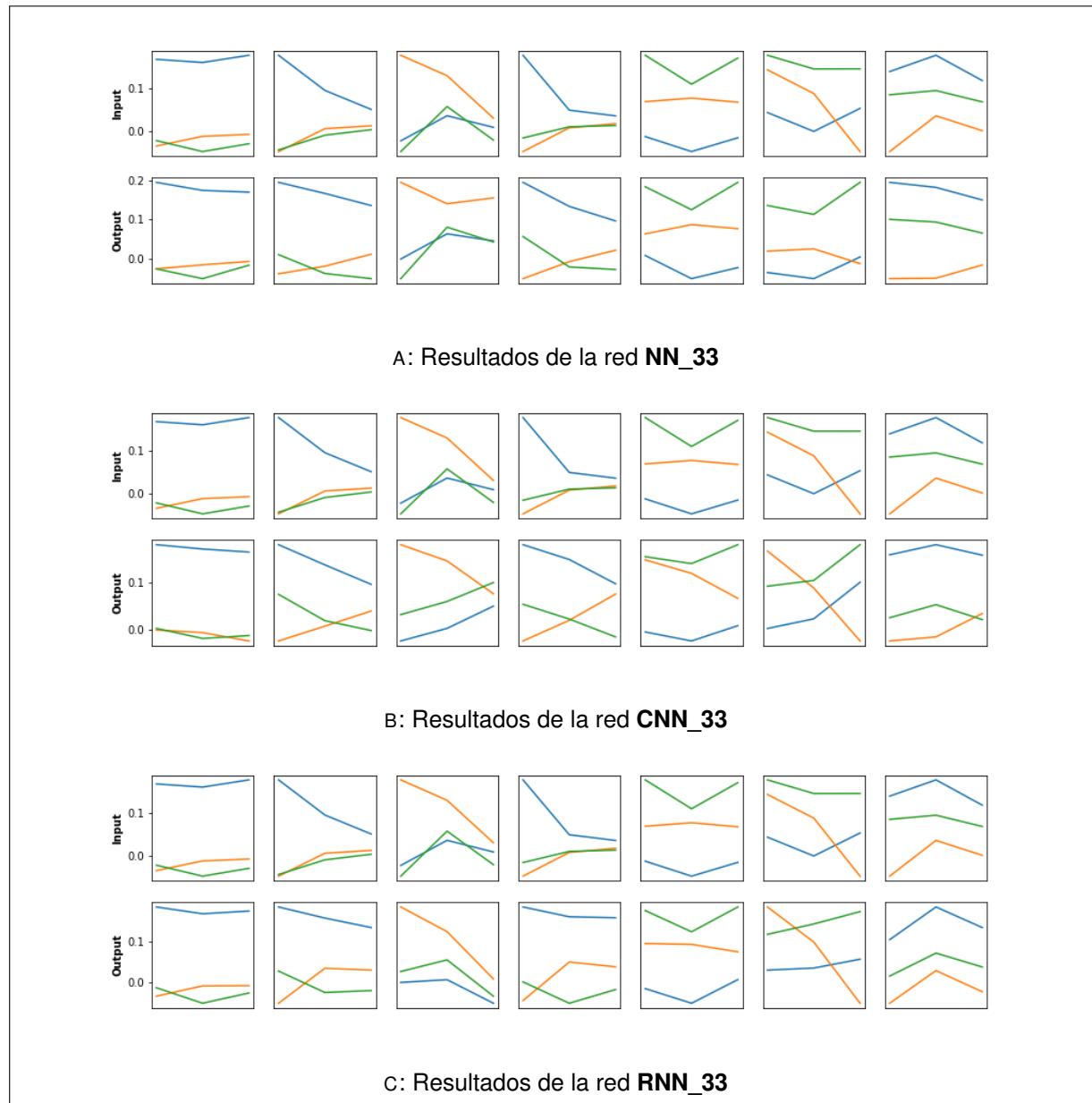


FIGURA 5.2: Resultados (Elaboración propia).

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.

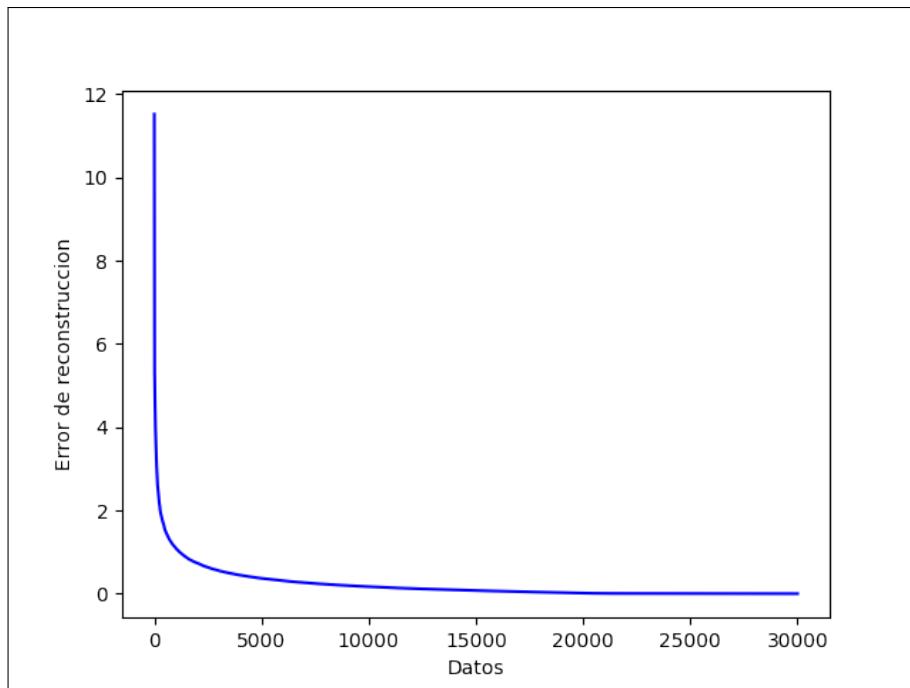


FIGURA 5.3: Curva de los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia).

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).

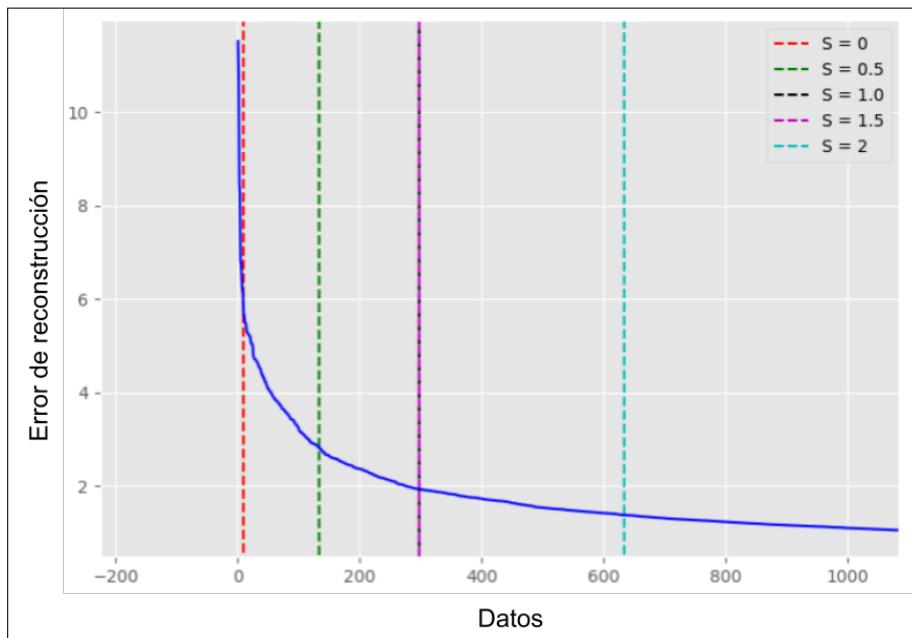


FIGURA 5.4: Resultados de la obtención de codos con diferentes valores de Sensibilidad, para los valores de reconstrucción obtenidos con el modelo del comportamiento normal (Elaboración propia).

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.

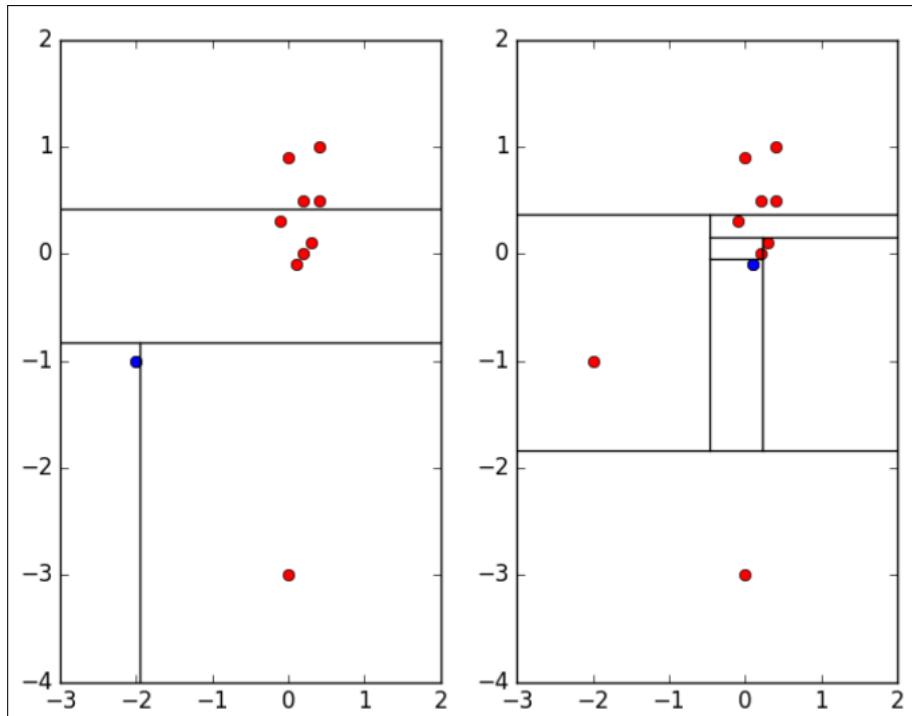


FIGURA 5.5: La figura de la izquierda muestra el aislamiento de una anomalía, que requiere solo tres particiones. A la derecha, el aislamiento de un punto normal requiere seis particiones (Wolpher, s.f.).

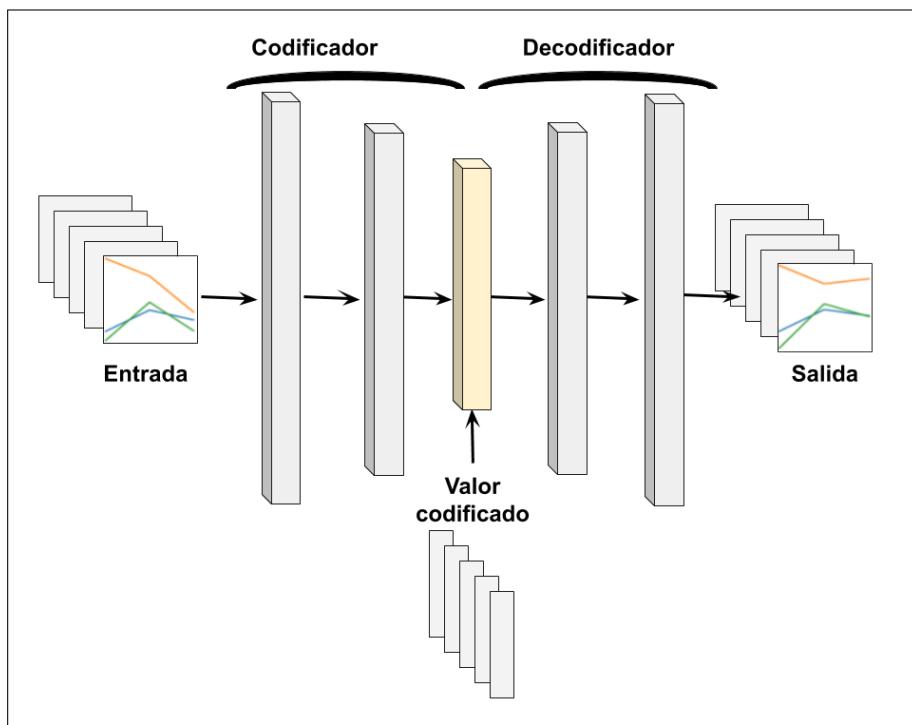


FIGURA 5.6: Representación gráfica del modelo de comportamiento normal o autoencoder (Elaboración propia).

- **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.

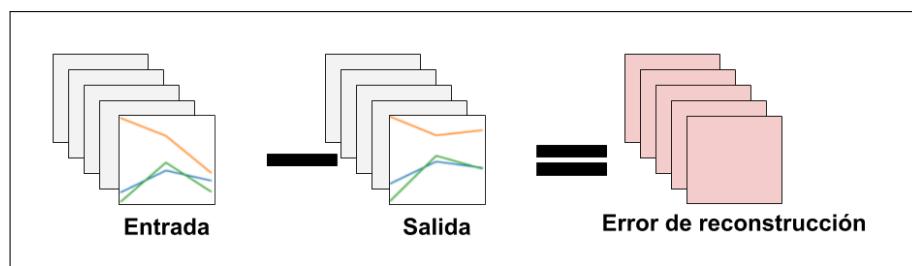


FIGURA 5.7: Representación gráfica del error de reconstrucción usado para el entrenamiento de los bosques de aislamiento y los SVM de una clase (Elaboración propia).

| 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.

| Nombre método | VP | VN | FN | FP | Sensibilidad | Especificidad |
|--|-----|-------|----|------|--------------|---------------|
| 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.

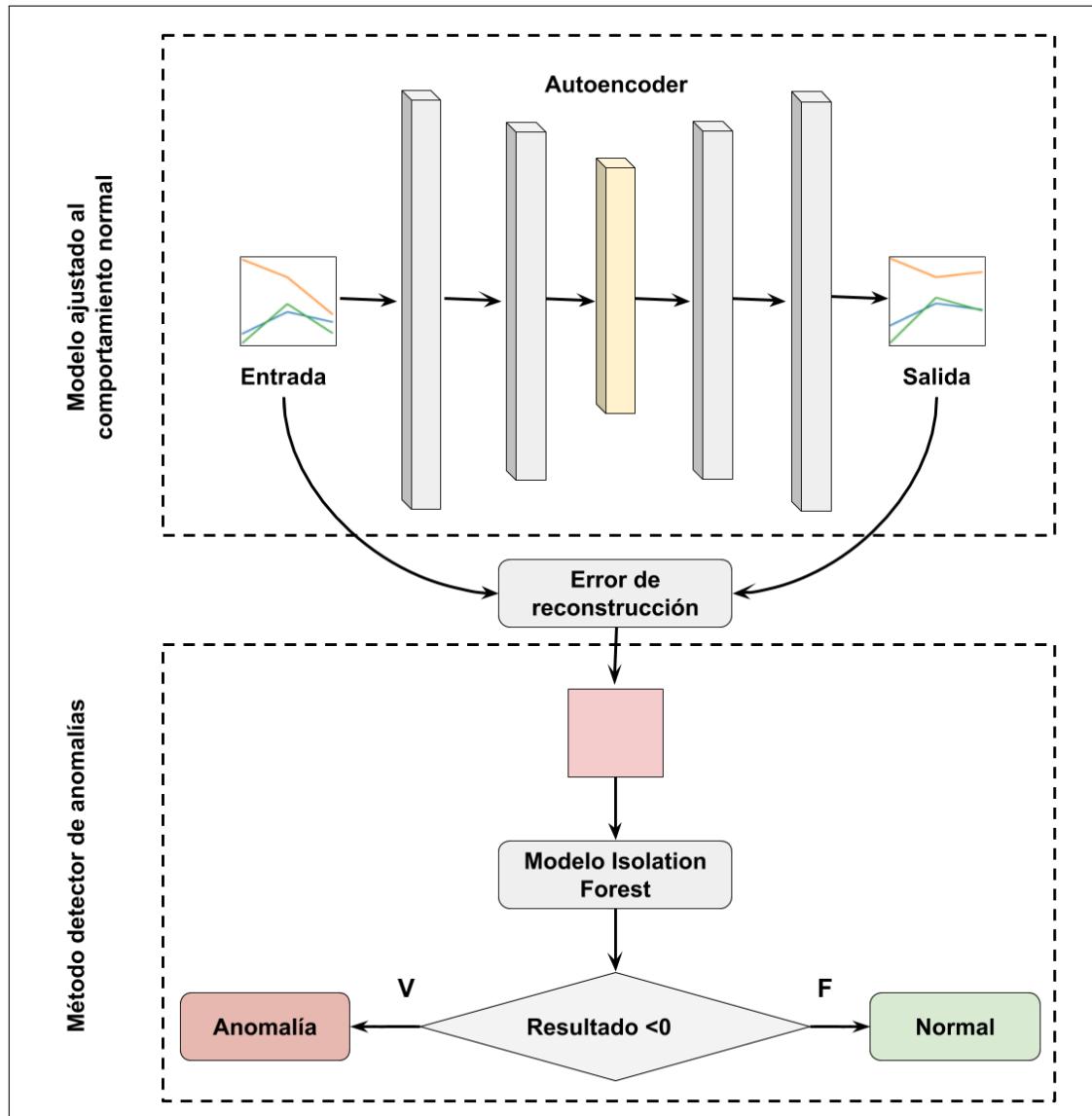


FIGURA 5.8: Mecanismo de detección de anomalías (Elaboración propia).

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.

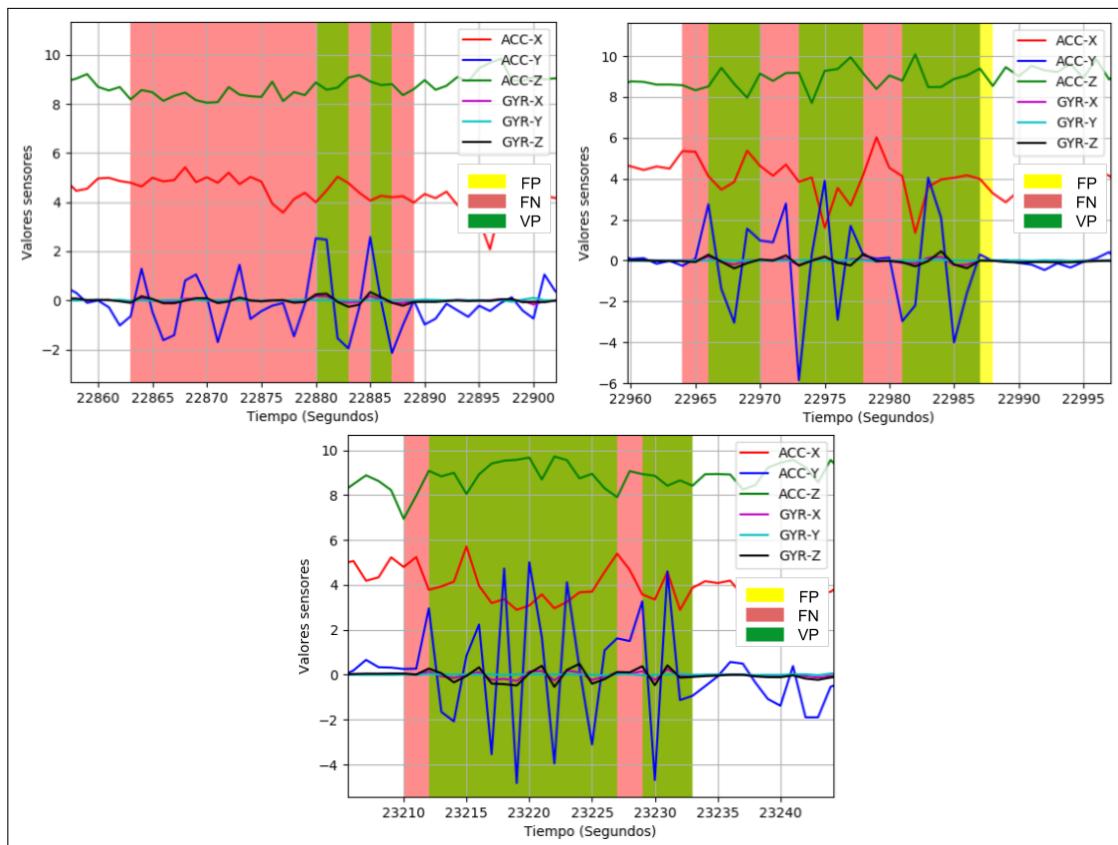


FIGURA 6.1: Resultados de la detección de anomalías del tipo zig zag (Elaboración propia).

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.

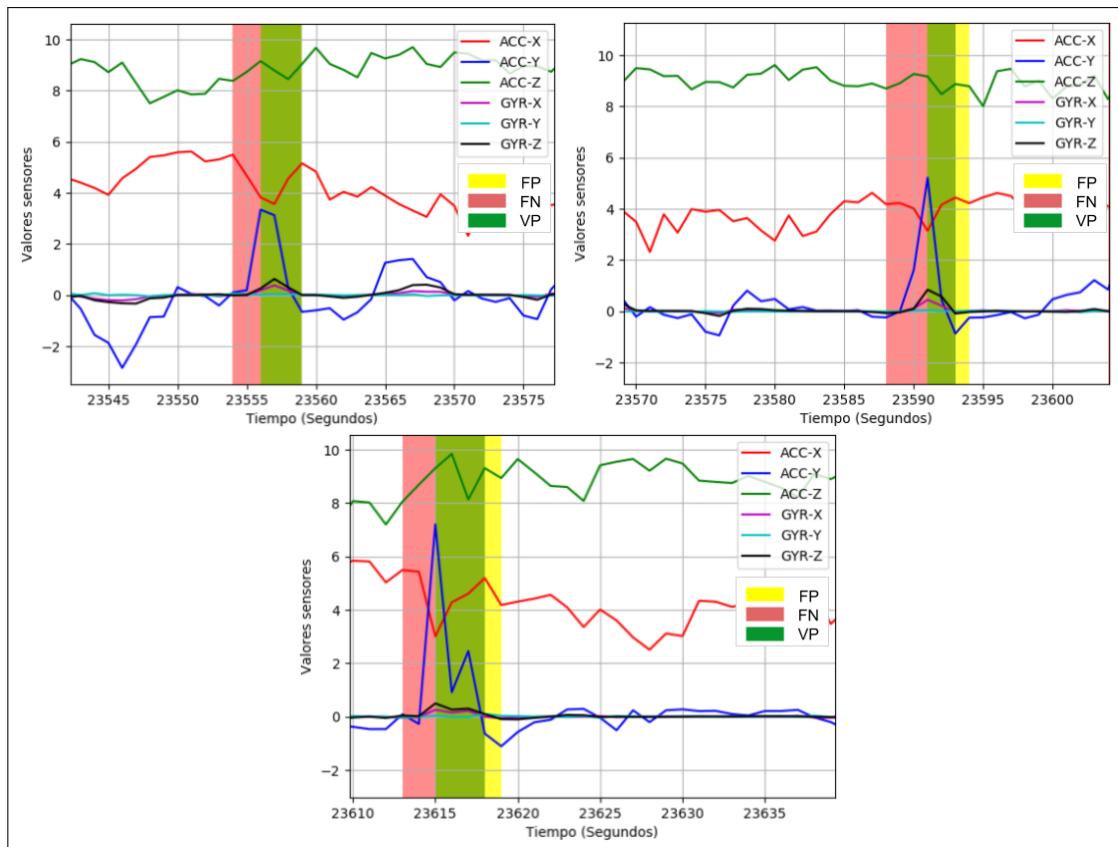


FIGURA 6.2: Resultados de la detección de anomalías del tipo giros a alta velocidad (Elaboración propia).

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.

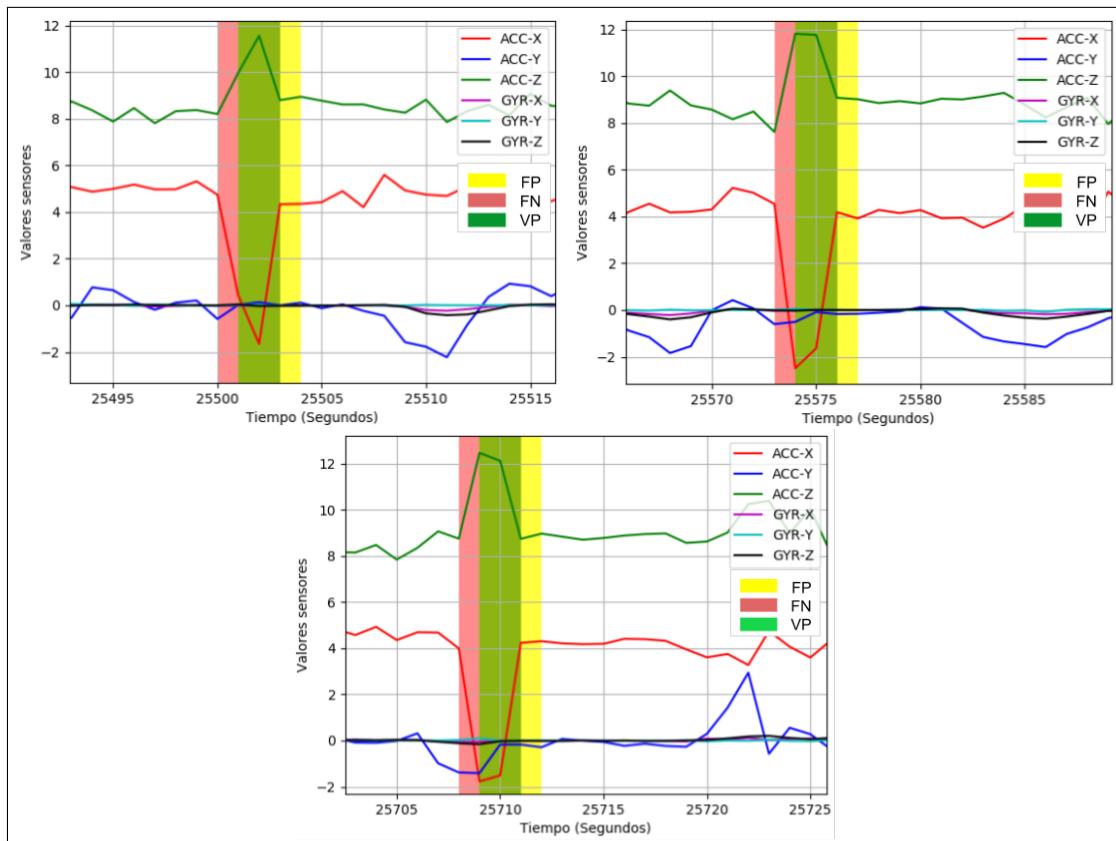


FIGURA 6.3: Resultados de la detección de anomalías del tipo frenos en seco
(Elaboración propia).

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.

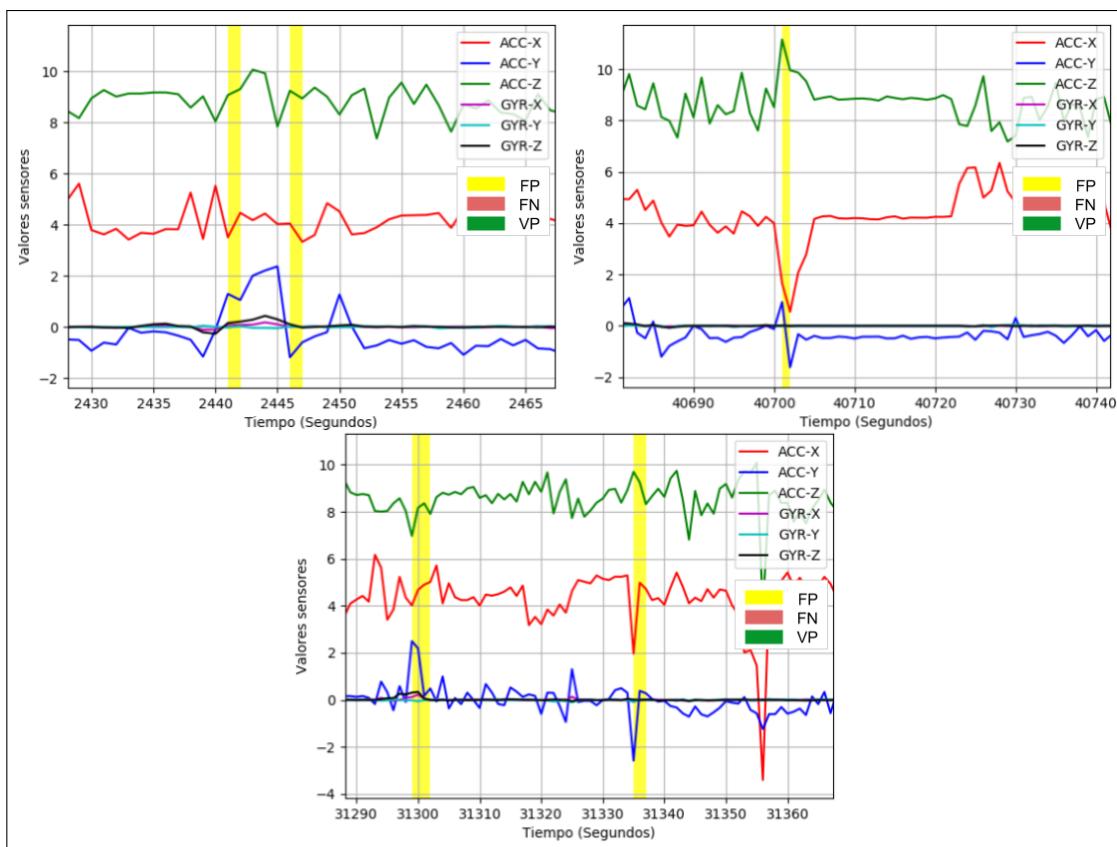


FIGURA 6.4: Resultados de la detección de falsos positivos (Elaboración propia).

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

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

7.2. Trabajos futuros

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

- Agregar la velocidad del vehículo como un nuevo parámetro del conjunto de datos, debido a que esto podría brindar un mejor entendimiento del comportamiento normal de conducción, así como también de las anomalías.
- En lugar de trabajar con los datos en crudo, usar la diferencia entre un dato capturado en el tiempo t y un dato capturado en $t - 1$ ($diff_t = dato_t - dato_{t-1}$), 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.

BIBLIOGRAFÍA

Referencias

- 0.22, S. (s.f.). *Novelty and outlier detection*. https://scikit-learn.org/stable/modules/outlier_detection.html. (Último acceso en 19 de diciembre de 2019)
- Alashwal, H., Bin D., S., 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
- 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. (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
- 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
- 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
- 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
- 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)) | | | | |
| Reshape | | | | |

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

A.3.2. Evaluación redes recurrentes

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

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

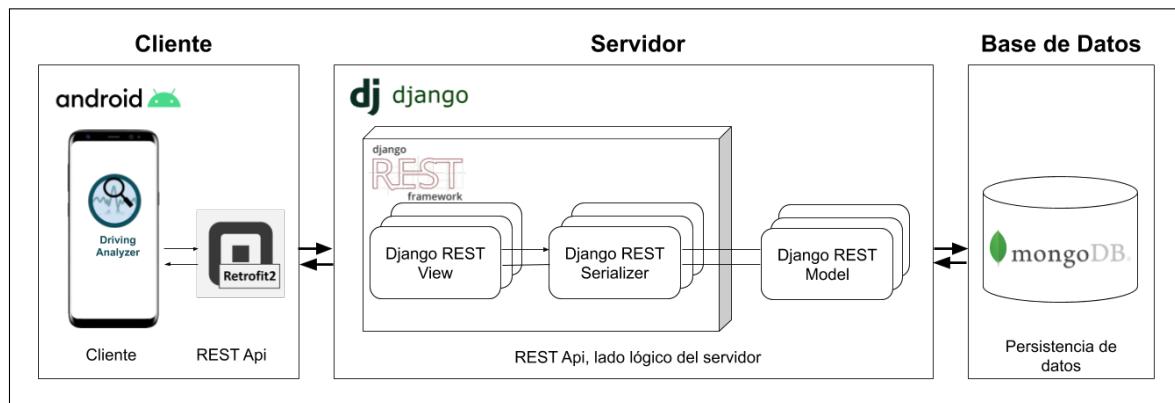


FIGURA B.1: Arquitectura física del Sistema de Demostración (Elaboración propia).

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

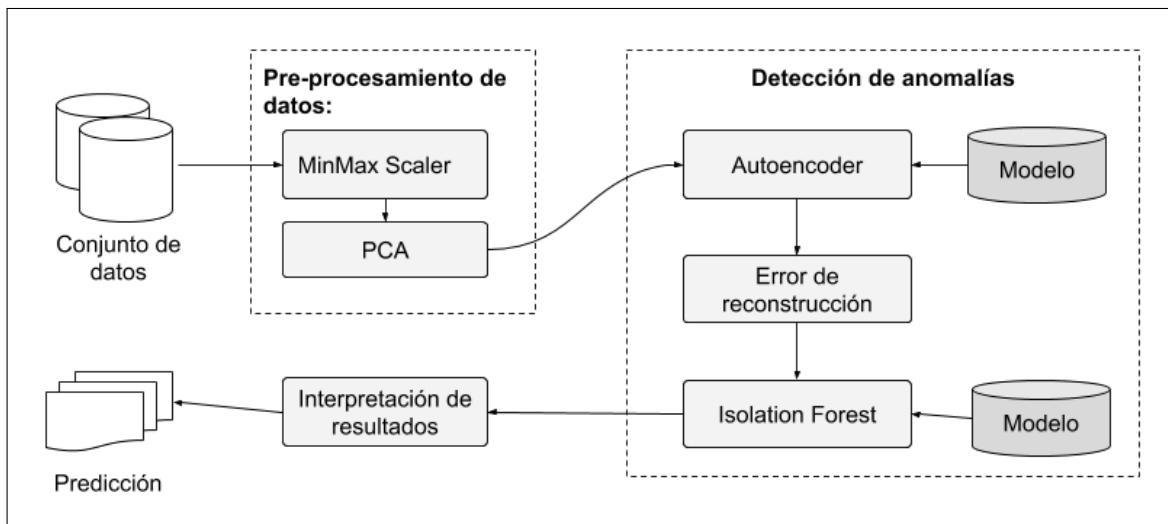


FIGURA B.2: Arquitectura Lógica del Sistema de Demostración (Elaboración propia).

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.