UNIVERSIDAD MAYOR DE SAN SIMÓN

PROYECTO DE GRADO

Detección de anomalías de conducción

Autor:
Evelyn CUSI LÓPEZ

Tutor:
Dr. Eduardo DI SANTI

Un proyecto de grado presentado en cumplimiento de los requisitos para el titulo de Licenciatura en Ingeniería de Sistemas

en la

Carrera de Ingeniería de Sistemas
Departamento Informática-Sistemas

19 de octubre de 2019

Agradecimientos

Le agradezco a Dios por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizajes, experiencias y sobre todo felicidad.

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

Índice de figuras

Índice de cuadros

UNIVERSIDAD MAYOR DE SAN SIMÓN

Resumen

Facultad de Ciencias y Tecnología
Departamento Informática-Sistemas

Licenciatura en Ingeniería de Sistemas

Detección de anomalías de conducción

por Evelyn CUSI LÓPEZ

El presente trabajo describe el desarrollo de un mecanismo de detección de anomalías de manejo, el cual es implementado usando un dispositivo móvil y técnicas de Inteligencia Artificial (IA).

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

El trabajo cuenta con dos partes principales: un modelo ajustado al comportamiento normal de manejo de un agente, y un método de detección de anomalías.....

La precisión de detección del método proupuesto en este proyecto es de .... que fue evaluado con ....

Capítulo 1

Introducción

El presente documento describe el desarrollo de un método para la detección de anomalías en la conducción. Se propone el uso de técnicas de Inteligencia Artificial 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 manejo.

La idea principal, es generar un modelo que aprenda el comportamiento normal de conducción de un usuario 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 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

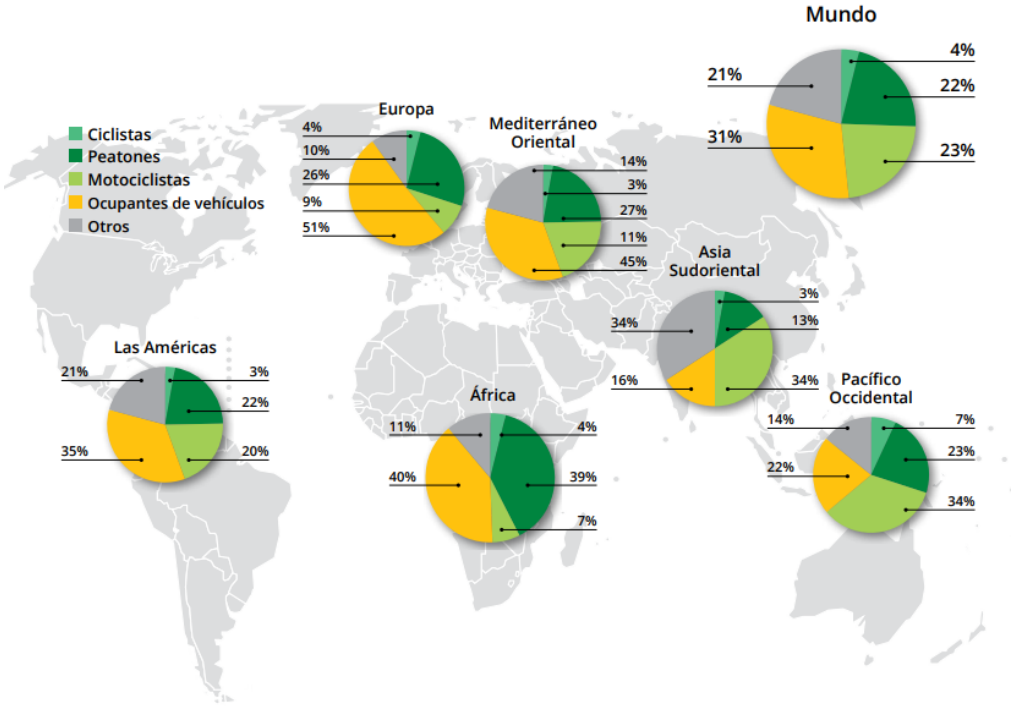


FIGURA 1.1: Muertes por accidentes de tránsito por región en función del tipo de usuario (2013), OMS

alertas al encontrar un comportamiento anómalo en el manejo, de manera que se pueda evitar o en todo caso minimizar los efectos del mismo.

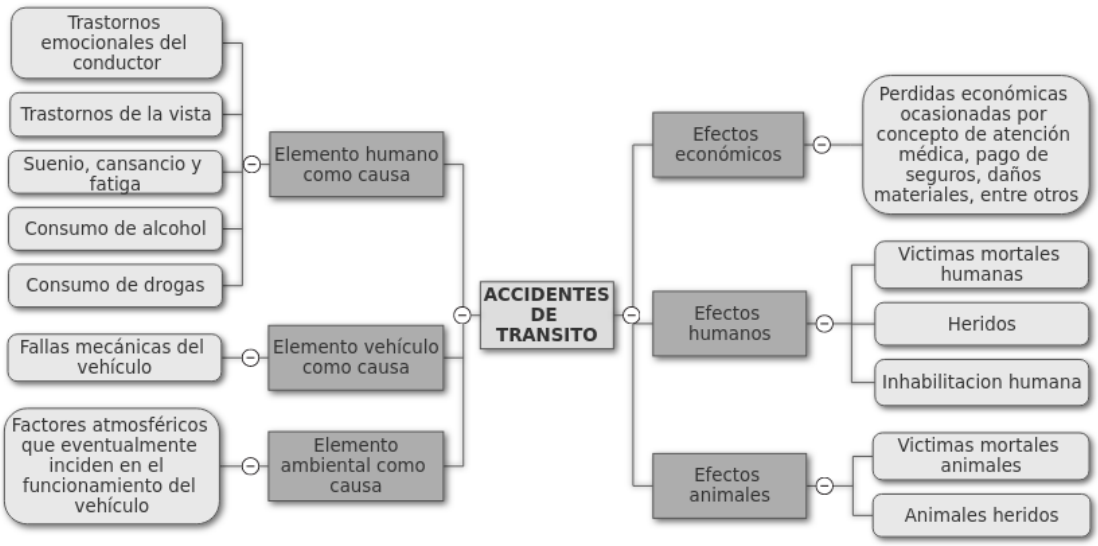


FIGURA 1.2: Árbol de problemas

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.

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 manejen 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 manejo 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 manejo de forma rápida. De esta manera se podrá evitar accidentes de tránsito o minimizar los efectos del mismo, 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 grado 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 U a alta velocidad.

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

1.6. Método de investigación

El presente trabajo de grado 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 Inteligencia Artificial?

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 expondrá los diferentes proyectos e investigaciones enfocados en la detección de anomalías de conducción.

2.1. Detección de nomalías

Para comprender lo que implica la detección de anomalías, es necesario asimilar lo que és una anomalía y de que maneras estas 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.

   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.

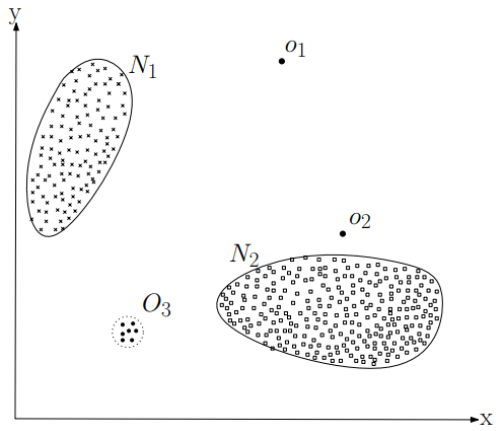


FIGURA 2.1: Ejemplo de anomalías de punto en un conjunto de datos de 2 dimensiones.

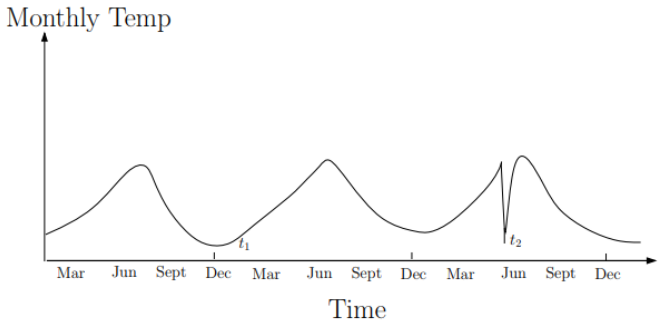


FIGURA 2.2: Anomalía contextual t_2 en una serie temporal de temperatura.

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.

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

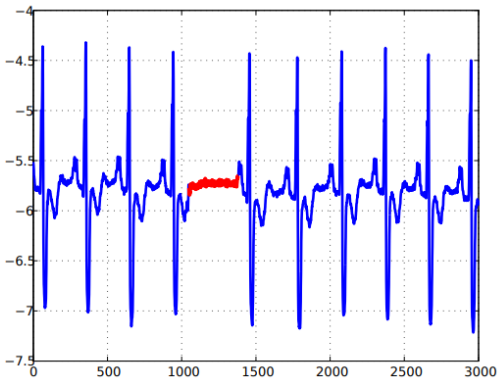


FIGURA 2.3: Anomalía colectiva correspondiente a una contracción prematura auricular en un electrocardiograma humano.

de intrusion de red, étcetera.

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 en 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 que 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 manejo. Esta sección esta dedicada al repaso de las mismas.

Dang-Nhac et al. [**20**], 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 en [**21**] Ferreira et al. 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).

En [**23**] 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 [**24**] se propone un sistema enfocado a desarrollar la conciencia del conductor mediante notificaciones en situaciones críticas que pueden desencadenar maniobras de manejo 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 ([**25**], [**26**], [**27**], [**28**], [**29**]) 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 portatil, 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 [**22**] proponen 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 manejo y manejan 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 de grado 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.

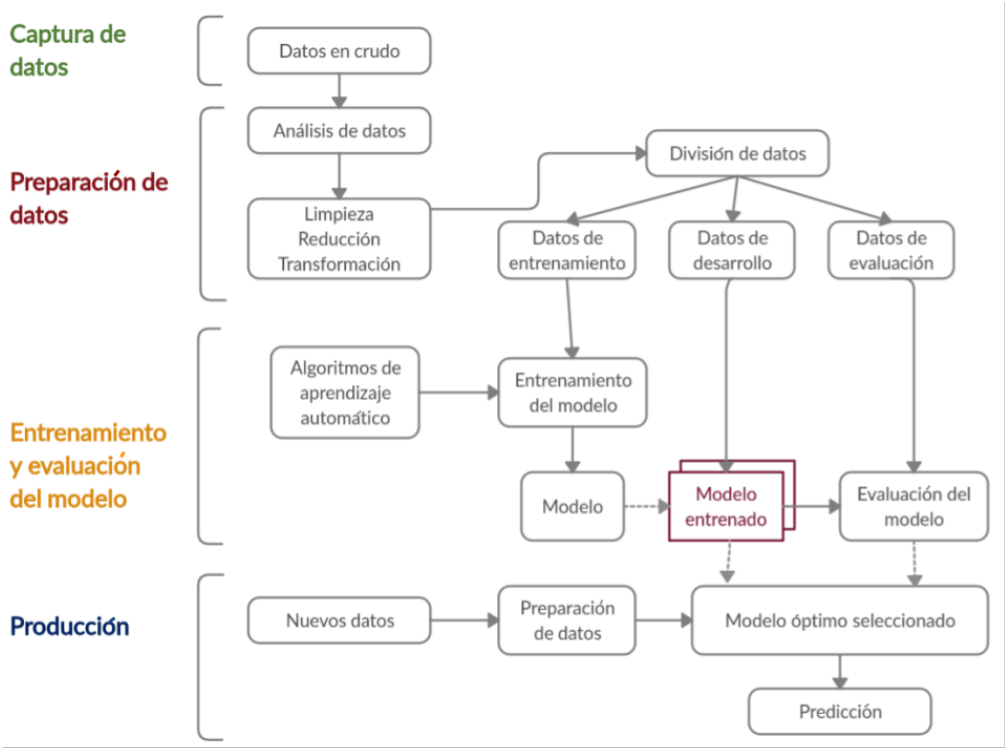


FIGURA 2.4: Método de detección de anomalías propuesto.

2.5. Resumen del capítulo

De acuerdo a los conceptos presentados acerca de la detección de anomalías, los desafíos que presenta la detección de las mismas y los trabajos de investigación realizados en este campo hasta la fecha, se propone un método de detección de anomalías de conducción mediante el uso de técnicas de Aprendizaje Automático y el uso de un dispositivo móvil, los mismos serán tratados en profundidad en los siguientes capítulos.

Capítulo 3

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 realiz´ para la presente investigación y las diferentes técnicas de análisis de datos que se aplicó.

3.1. Captura de datos

Actualmente existen varios enfoques para acceder a la información del coductor (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) ([**30**], [**31**]), 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 inerciales (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 grado 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 3.1)

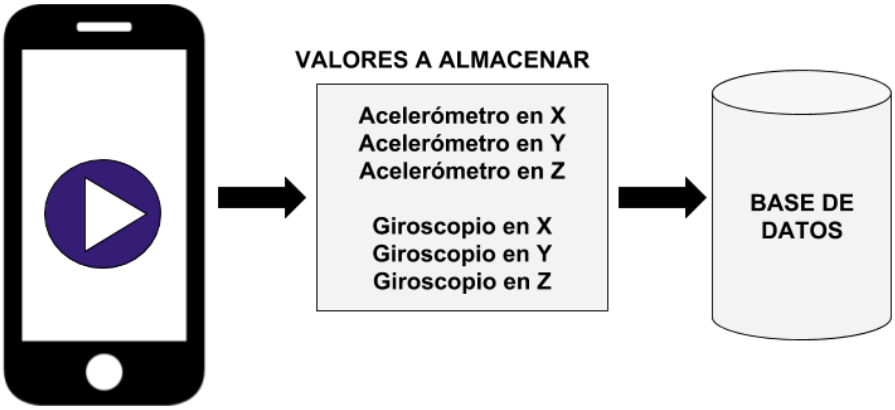


FIGURA 3.1: Recolección de datos, con intervalo de un segundo.

Para la captura de datos se usó un soporte para celular de parabrisas como se ve en la Figura 3.2; se realizó la captura en dos posiciones distintas (vertical y horizontal).

FIGURA 3.2: Soporte para celular de parabrisas, posición horizontal.

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 3.3 se aprecia un fragmento del conjunto de datos que se obtuvo en una captura.

3.2. 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

_id	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z
	Filter	Filter	Filter	Filter	Filter	Filter
1	-0.4353179...	9.60725402...	1.84175109...	9.15527343...	-0.0020294...	0.00770568...
2	-1.1603393...	9.73767089...	1.84773254...	-0.0500488...	0.02827453...	0.00379943...
3	-0.3013153...	9.57853698...	1.90994262...	0.00070190...	-0.0008697...	0.00354003...
4	-0.3479614...	9.59648132...	1.89201354...	0.00035095...	-0.0009918...	0.00372314...

FIGURA 3.3: Fragmento del conjunto de datos obtenido.

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 [**37**], para lograr este propósito se debe seguir un proceso que implica la selección, el pre-procesamiento y la transformación de datos.

3.2.1. Selección de datos

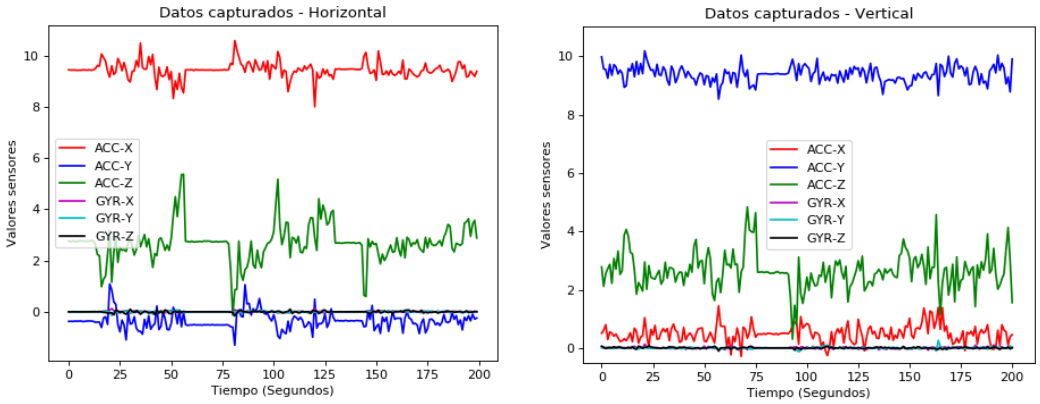
La selección de datos implica los siguientes pasos:

- Seleccionar solo un subconjunto de los datos disponibles.
- Derivar o simular algunos datos a partir de los datos diponibles, en caso de ser necesario.
- Excluir aquellos datos que no son reelevantes 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 3.1, esta captura se realizó, tanto de forma vertical como horizontal, a continuación se analizará las diferencias entre ellos.

En la Figura 3.4 se muestra fragmentos de las capturas obtenidas por el dispositivo móvil del manejo 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 suceptibles 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.



(A) Captura de datos en horizontal (B) Captura de datos en vertical

FIGURA 3.4: Gráfica de los sensores capturados en diferentes posiciones.

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.

3.3. 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 redución de los datos [**38**].

3.3.1. 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 encontrar valores ausentes, suavizar el ruido en los datos, reconocer valores atípicos y corregir inconsistencias.

Manejo de valores ausentes

Los valores ausentes en el conjunto de datos pueden ser llenados usando cualquiera de las técnicas que se mencionan a continuación.

- **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 nuestro 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 mas probable y preciso para los lugares faltantes, esto se puede lograr mediante regresión, formulación bayesiana e inducción por árboles de decisión.

Eliminar rudio (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 nuestros 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 nuestros datos, para poder evidenciar si existe ruido en los datos y/o valores atípicos.

En el presente proyecto se graficó histogramas de las frecuencias de cada característica de nuestro conjunto de datos y así visualizar de mejor manera si existe ruido o valores atípicos en el mismo. En la Figura 3.5 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 dá un indicio de posibles valores atípicos.

En la Figura 3.6 se presenta una tabla con estadísticas descriptivas de nuestro 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.

Con la información proporcionada en la Figura 3.6 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 0.02 y 0.79 y sin embargo la diferencia entre los valores mínimo y máximo son realmente grandes, por ejemplo para el Acelerómetro en X, la diferencia es 16 aproximadamente (Valor mínimo: -1.998 y valor máximo: 14.31), esto quiere decir que nuestro conjunto de datos

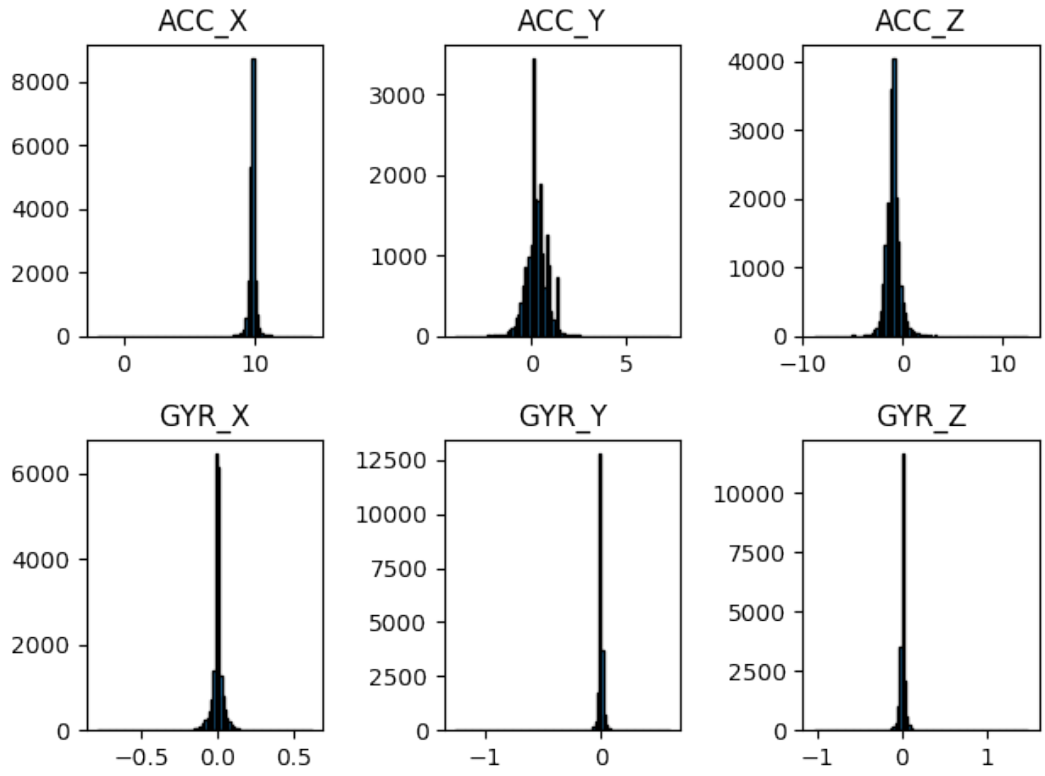


FIGURA 3.5: Histograma de frecuencias del conjunto de datos

	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000
mean	9.744480	0.263008	-0.992000	0.000492	-0.000743	0.000096
std	0.449664	0.582752	0.786257	0.048442	0.023572	0.039129
min	-1.998215	-4.037933	-8.923569	-0.790817	-1.261871	-1.051468
25%	9.677216	-0.018173	-1.399807	-0.006760	-0.005203	-0.007908
50%	9.766968	0.215057	-1.005829	0.000000	-0.000046	-0.000015
75%	9.822723	0.555355	-0.725754	0.006653	0.003510	0.008057
max	14.312866	7.334473	12.555618	0.621399	0.599594	1.495499

FIGURA 3.6: Tabla de resultados estadísticos del conjunto de datos.

presenta mucho ruido, considerando como ruido o valores atípicos, aquellos valores muy alejados de la media.

Para eliminar el ruido que presenta nuestro 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 3.7 se muestra este concepto de forma esquemática.

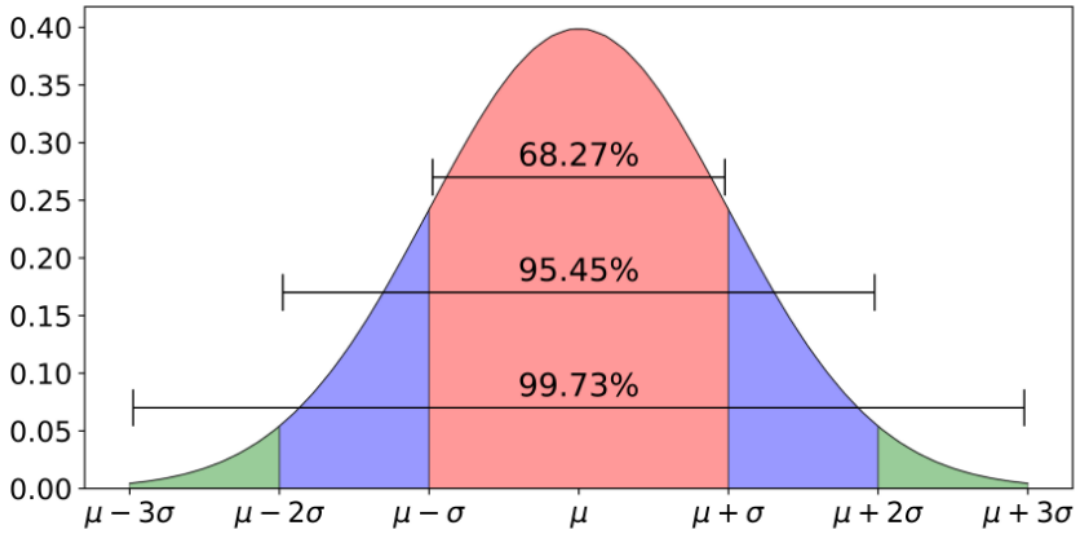


FIGURA 3.7: Regla 68-95-99.7

En la Figura 3.8, se puede observar como se comporta la regla 68-95-99.7 sobre los valores del sensor del acelerómetro en Y, por lo que para eliminar el ruido en nuestro conjunto de datos se tiene dos opciones, eliminar los valores después de tres desviaciones estándar de la media, en caso de considerar que nuestro conjunto de datos presenta pocas anomalías o valores atípicos, o eliminar los valores después de dos desviaciones estándar en caso de estar seguro que nuestro conjunto de datos presenta una gran cantidad de ruido en los datos, en nuestro caso la mayoría de nuestros datos pertenecen a comportamientos normales de conducción por lo que sólo eliminaremos los valores que se encuentran después de tres desviaciones estándar de la media.

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.

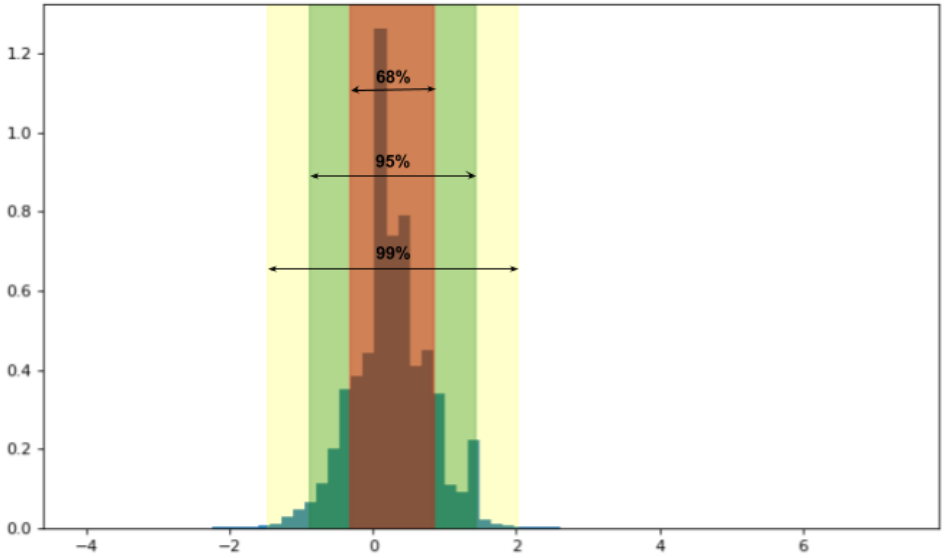


FIGURA 3.8: Regla 68-95-99.7 aplicada a los valores de los sensores del acelerómetro en Y.

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 datos vagos, además puede ser adecuada cuando se tiene menos características pero aún contienen información oculta para extraer.
- **Normalización / Estandarización:** La normalización o estandarización se define como el proceso de reescalar datos originales sin cambiar su comportamiento o naturaleza. Se define un nuevo límite (generalmente entre 0 y 1) y se convierte los datos en consecuencia. Esta técnica es útil en algoritmos de clasificación que involucran redes neuronales o algoritmos basados en la distancia (por ejemplo, KNN, K-means). Algunas técnicas de normalización son:

- *Normalización Min-Max:* En este método, cada entrada se normaliza entre unos límites definidos:

$$x_{normalizado} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

  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.

$$x_{normalizado} = \frac{x - x_{media}}{x_{desvSt}} \tag{3.2}$$

  Éste 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 interquartil (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 [**39**].

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, Richard Bellman en [**40**] 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 importantesz al mismo tiempo conserva las partes más valiosas (o componentes principales[1]) 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 entrenamiendo, 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*<=*d*).
7. Construir una matriz de proyección **W** a partir de los "top"*k* vectores propios.
8. Transformar el conjunto de entrenamiento de entrada *d*-dimensional **X** utilizando la matriz de proyección **W** para obtener el nuevo subespacio de característica *k*-dimensional.

División del conjunto de datos

Dado que se cuenta con un conjunto de datos para generar el modelo de Aprendizaje Automático, este se divide en tres partes: conjunto de entrenamiento[2], desarrollo[3] y prueba[4], sin embargo esto no es una tarea trivial; ya que si no se hace correctamente el resultado puede ser desastroso.

En [**41**] se dice que la división del conjunto de datos tiene un gran impacto en la productividad , por lo cual es importante que al elegir todos nuestros subconjuntos estos deben tener la **misma distribución** y deben ser elegidos aleatoriamente del conjunto de datos.

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

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

[3] **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.

[4] **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.

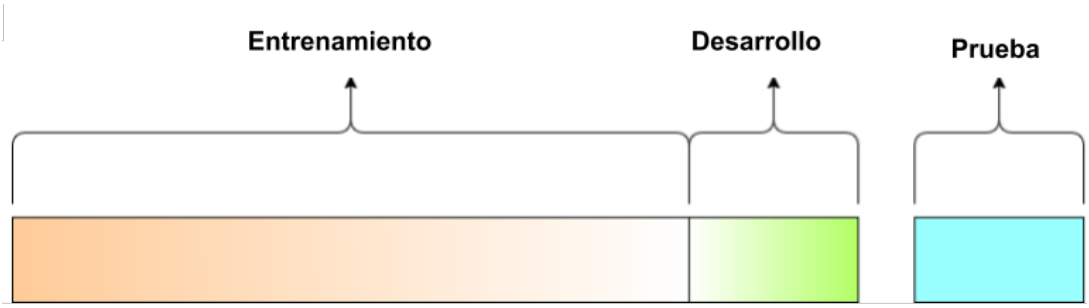


FIGURA 3.9: División del conjunto de entrenamiento.

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

CUADRO 3.1: Tabla de división del conjunto de datos.

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 nuestro conjunto de datos cuenta con 30000 ejemplos, por lo que la división de nuestro conjunto de datos será la que se observa en el Cuadro 3.1.

Estandarizar / Normalizar el conjunto de datos

En la sección 3.3.1 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 3.10 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.

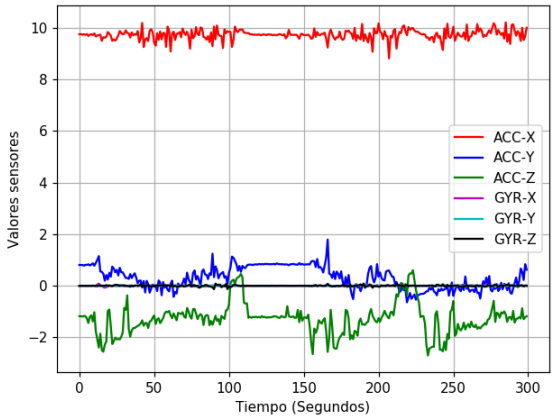


FIGURA 3.10: Visualización de los parámetros de conducción capturados.

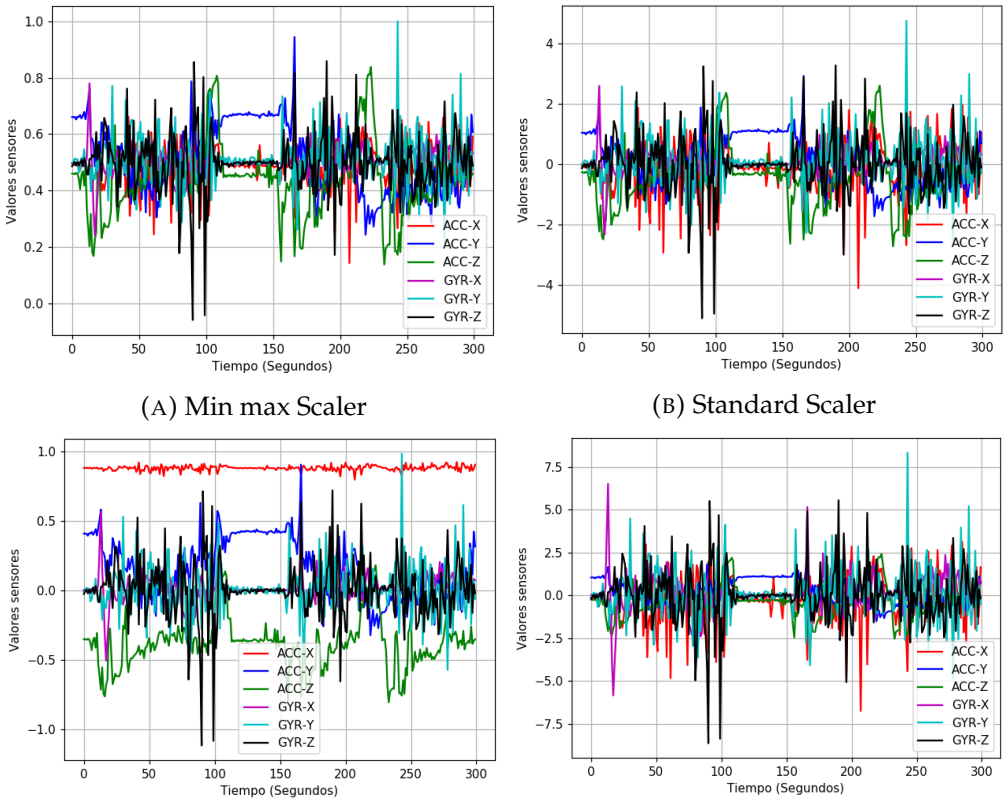
Para probar los diferentes tipos de escalado, se los ajusto con los datos que ya no presentan ruido o valores atípicos del conjunto de entrenamiento.

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 3.11a, donde se puede apreciar que los valores del acelerómetro, en sus tres ejes, no se veen 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 nuestros 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 3.11b, 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 3.11c. Esta técnica presenta los peores resultados, ya que no deja nuestro conjunto de datos en un mismo rango lo cual complica su trabajo.

La última técnica aplicada es el **escalado robusto**, su resultado puede ser observado



(A) Min max Scaler

(B) Standard Scaler

(C) Max Scaler

(D) Robust Scaler

FIGURA 3.11: Gráfica resultante de aplicar diferentes tipos de normalizaciones a un conjunto de datos.

en la figura 3.11d, el cual puede ser considerado bastante similar a las dos primeras técnicas, con diferencia de que este escalado pronuncia mucho más las fluctuaciones de los valores de los giroscopios.

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 estánda baja, por lo cual este puede ser un parámetro de decisión importante para elegir el tipo de escalado más adecuado.

En la tabla 3.2 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 todas oscilan entre 0.166759 y 0.169528, lo cual se adecúa muy bien para el análisis de componentes principales, que es lo que se busca en este punto.

		ACC X	ACC Y	ACC Z	GYR X	GYR Y	GYR Z
Min-Max	Media	0.494631	0.504011	0.500105	0.497873	0.499490	0.499922
	Desv. Est.	0.169528	0.168058	0.166759	0.167592	0.167005	0.166876
	Mínimo	-3.932480	-0.736326	-1.182118	-2.239772	-8.435568	-3.984707
	Máximo	2.216958	2.543331	3.373446	2.645991	4.752865	6.877400
Estándar	Media	-0.053019	0.006007	0.035521	0.013953	-0.007687	-0.001008
	Desv. Est.	1.956964	1.116028	1.270322	1.537762	1.591346	1.525546
	Mínimo	-51.157948	-8.230720	-12.779184	-25.105641	-85.147607	-40.998624
	Máximo	19.828881	13.548560	21.923844	19.724263	40.521651	58.300659
Valor Máximo	Media	0.879073	0.132640	-0.296123	0.003375	-0.010434	0.000820
	Desv. Est.	0.040565	0.293892	0.234706	0.332638	0.330862	0.333426
	Mínimo	-0.180264	-2.036399	-2.663783	-5.430323	-17.712144	-8.959693
	Máximo	1.291199	3.698900	3.747990	4.266974	8.416149	12.743336
Robusto	Media	-0.159037	0.083165	0.027062	0.039046	-0.083081	0.007354
	Desv. Est.	3.178981	1.031395	1.181458	3.848120	2.807440	2.582457
	Mínimo	-83.176214	-7.528934	-11.891205	-62.820606	-150.286234	-69.393756
	Máximo	32.138026	12.598724	20.384211	49.362424	71.418446	98.700906

CUADRO 3.2: Tabla con estadísticas descriptivas de los datos escalados con diferentes técnicas.

Aplicar PCA al conjunto de datos

Aunque al inicio de esta subsubsección 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 debe decidir la cantidad correcta de componentes principales para nuestro conjunto de datos, para ello haciendo uso de esta clase se calcula la varianza de cada atributo y posteriormente se grafica los resultados (Ver Figura 3.12).

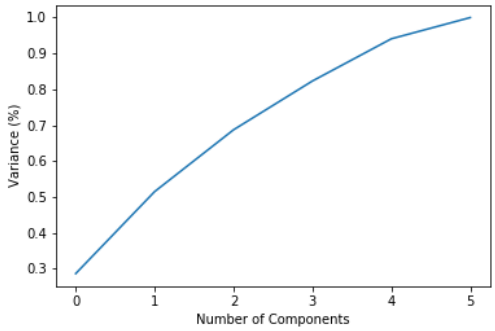


FIGURA 3.12: Gráfico de la varianza vs el número de componentes.

El Gráfico 3.12 indica que seleccionando 4 componentes se puede preservar alrededor del 95 % de la varianza total de los datos y que seleccionando 3 componentes se conserva alrededor del 80 % de la varianza de los datos, usar cualquiera de esas cantidades de componentes tiene sentido, ya que no se usará el 100 % de la varianza, porque denota todos los componentes y sólo se desea usar los principales.

3.4. Resumen del capítulo

En este capítulo se ha descrito la forma en la que se realizó la captura del conjunto de datos de conducción y la preparación del conjunto de datos, mediante distintas técnicas como la selección, limpieza, reducción y transformación de datos; con el fin de que estos valores sean una mejor entrada para el algoritmo de aprendizaje automático que se aplicará en el siguiente capítulo.

Capítulo 4

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 [**32**]. 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.

4.1. 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.

4.1.1. 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.

$$Y = f(X) \tag{4.1}$$

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.

4.1.2. 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).

4.1.3. 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 díficil 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 esta 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 [**33**].

En muchas aplicaciones, incluyendo la del presente trabajo de grado, 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** que nos ayude 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.

4.2. Modelos Generativos y Descriptivos

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 **[42]**.

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 nos ayudan a especificar la incertidumbre de un modelo, algunos ejemplos de modelos generativos son: Gaussian Mixture Model, Hiden Markov Model, Restricted Bolzmann Machine, Generative Adversial 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. Debido a esto y a que la tarea que se tiene como objetivo es bastante compleja, en la presente investigación se optará por el 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 grado.

4.3. Redes Neuronales Artificiales

La Red Neuronal Artificial o ANN[1] 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.

4.3.1. 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 4.2).

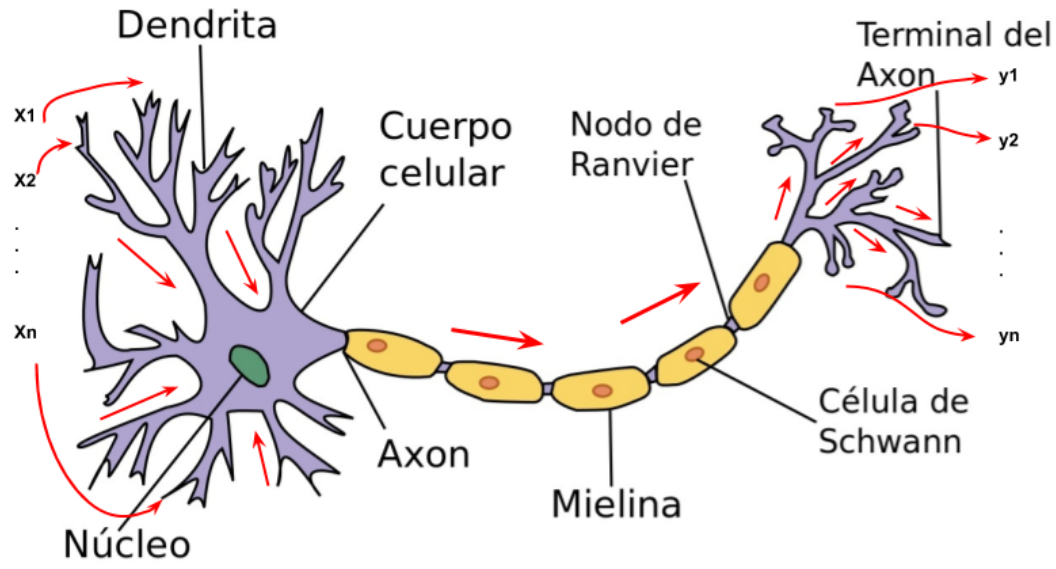


FIGURA 4.1: Gráfico de una neurona biológica.

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

La conexión entre neuronas se llama **sinapsis**, esta no es una conexión física, sino que existe 2 mm. de separación entre neronas. 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.

[1] **ANN**, Artificial Neural Network (Red Neuronal Artificial)

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

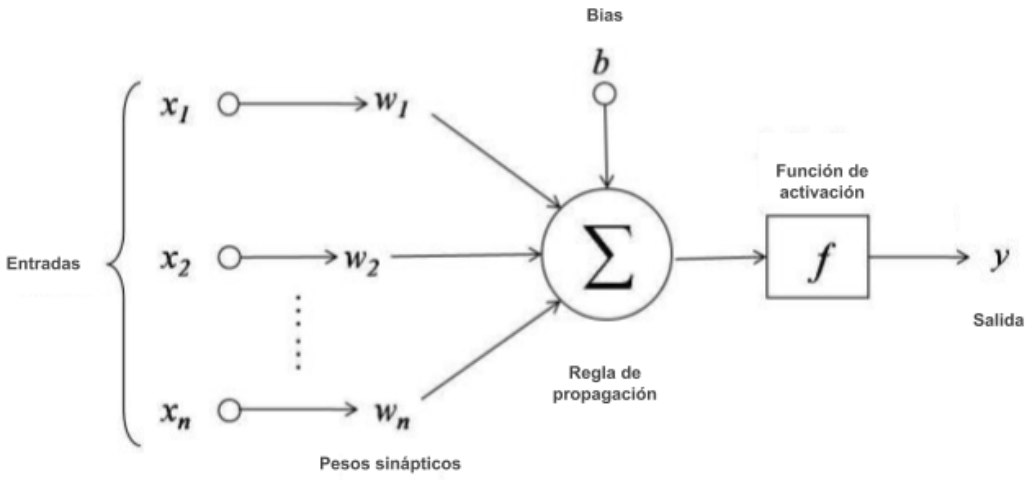
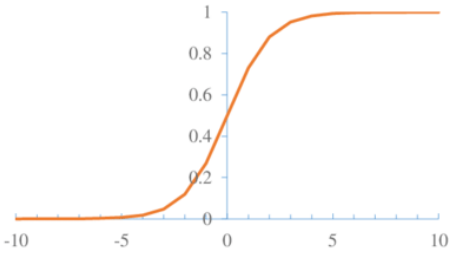


FIGURA 4.2: Gráfico de una neurona artificial.

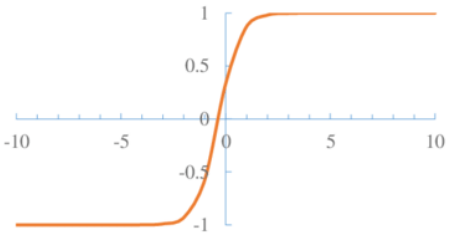
- **Las entradas** que reciben los datos de otras neuronas, estas entradas serían las dendritas de una neurona biológica.
- **Los pesos sinápticos** w_{ij}. En una neurona artificial a aquellas entradas que vienen de otras neuronas se les asigna un peso (factor de importancia). Este peso es un valor 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 más comunes es sumar las entradas, pero teniendo en cuenta la importancia (peso sináptico) de cada una; esta operación se llama *suma ponderada* 4.2, sin embargo otras operaciones también son posibles. Otra regla de propagación que es habitual es la distancia euclidea.

$$h_i(t) = \sum_j w_{ij} x_j \tag{4.2}$$

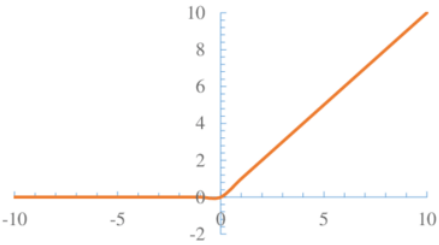
- **Función de activación**. El valor obtenido con la regla de propagación, se filtra a través de una función conocida como *función de activación* y es la que da la salida de la neurona. La función de activación es importante debido a que es la que decide si una neurona debe activarse o no, además si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



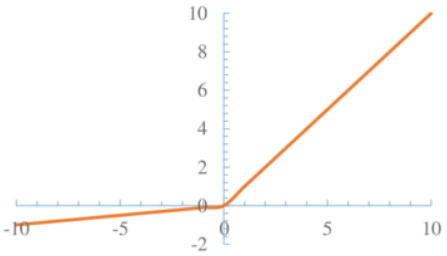
(A) Función Sigmoide



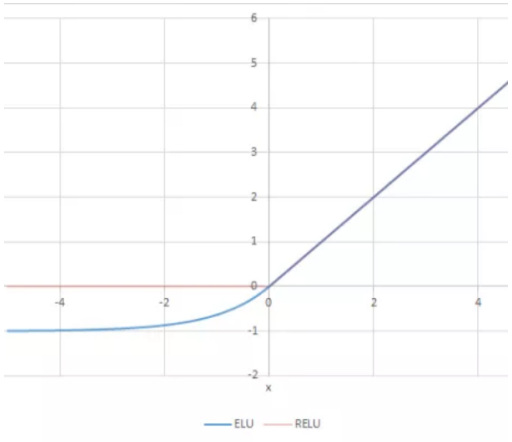
(B) Función Tanh



(C) Función ReLU



(D) Función Leaky ReLu



(E) Función ELU

FIGURA 4.3: Funciones de activación.

4.3.2. 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 4.3a), 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:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

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. Ésta función está centrada en 0 y su rango se encuentra entre -1 y 1 (Ver Figura 4.3b), por lo tanto, su salida esta definida por:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.4}$$

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 [**43**] y tareas de reconocimiento del habla [**44**].

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 [**46**], 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 4.3c), por lo que ReLU esta definida por:

$$f(x) = max(0, x) = \begin{cases} \text{si } x_i \geq 0 & x_i \\ \text{si } x_i < 0 & 0 \end{cases} \tag{4.5}$$

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 [**45**]. 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 [**44**]. 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 4.3d) se calcula como:

$$f(x) = \alpha x + x = \begin{cases} \text{si } x_i > 0 & x_i \\ \text{si } x_i \leq 0 & \alpha x_i \end{cases} \tag{4.6}$$

Función de Unidad Lineal Exponencial (ELU)

La función ELU[2] tiende a converger el costo a cero más rapido 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 4.7.

$$f(x) = \begin{cases} \text{si } x_i > 0 & x_i \\ \text{si } x_i \leq 0 & \alpha * (e^{x_i} - 1) \end{cases} \tag{4.7}$$

4.3.3. 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.

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

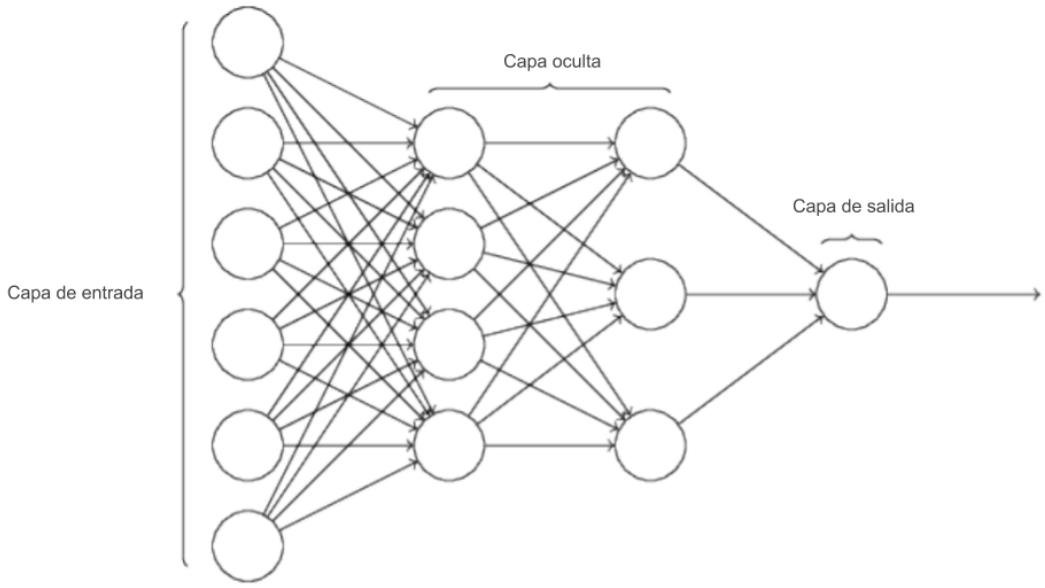


FIGURA 4.4: Arquitectura de una neurona artificial.

La figura 4.4 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 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.

4.3.4. 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.

4.4. Tipos de Redes Neuronales

4.4.1. 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 4.5. Los autoencoders suelen ser usados para reducción de dimensionalidad y aprendizaje de características.

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 $h = encoder(x)$, 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, $\hat{x} = decodificador(h)$.

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.

4.4.2. Redes neuronales convolucionales

Para algunos tipos de datos, especificamente para imágenes, las redes neuronales convencionales no están bien adaptadas; lo cual conlleva que en **[48]** propongan las redes neuronales convolucionales (CNN[3]) para solucionar ese problema. Las CNN han

[3] **CNN,** Convolutional Neural Network

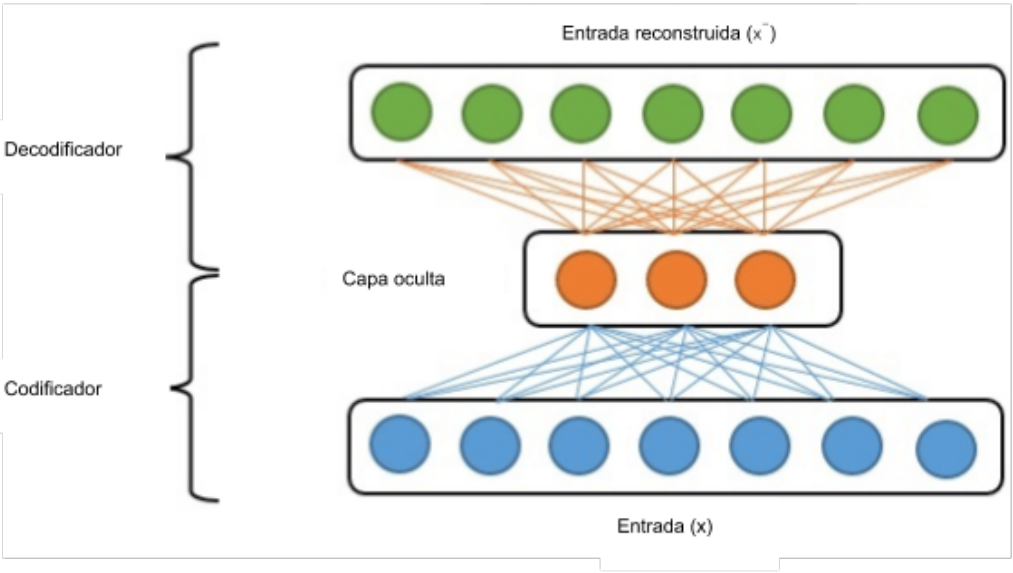


FIGURA 4.5: Gráfico de un Autoencoder.

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 **[49]**.

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 4.6 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.

4.4.3. 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.

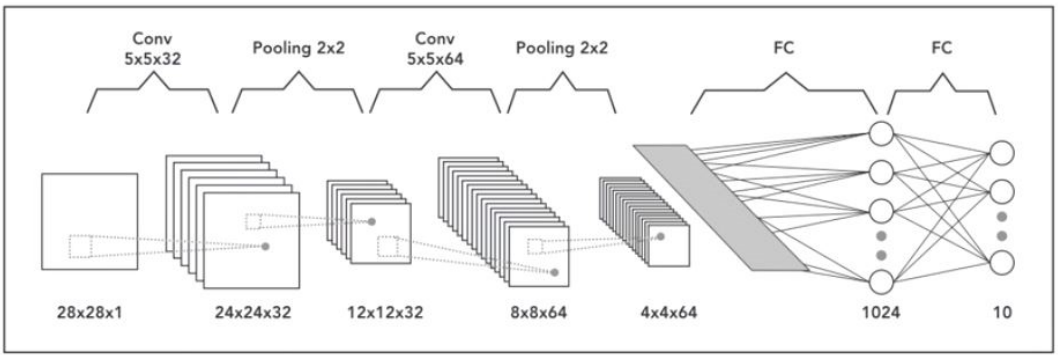


FIGURA 4.6: Arquitectura de una Red Neuronal Convolucional (CNN).

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 (RNN[4]), las cuales son un tipo de red neuronal artificial propuesta en los años 80 ([**34**], [**35**], [**36**]) 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.

En la Figura 4.7 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 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 (BPTT[5]). 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 [**50**]. Para superar

[4] **RNN,** Recurrent Neural Network
[5] **BPTT,** Backpropagation Through The Time

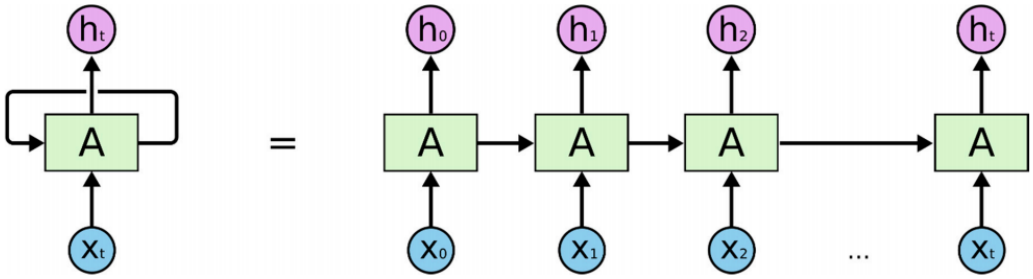


FIGURA 4.7: Procesamiento secuencial en una red neuronal recurrente (RNN) [**53**].

los problemas de gradiente de desaparición y explosión que tienen las RNN estándar, se pueden usar LSTM y GRU [**51**].

4.4.4. LSTM

LSTM[6] es una evolución de RNN, fue introducida por Hochreiter y Schmidhuber en [**52**] 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 4.8.

Una típica red LSTM esta 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 sigmoideas.

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 utiliza puertas para controlar el proceso de memorización.

4.4.5. GRU

GRU[7] fue diseñado por primera vez por Kyunghyun Cho en [**55**]. Esta estructura de RNN sólo contiene dos puertas. La puerta de actualización controla la información

[6] **LSTM,** Long Short-Term Memory
[7] **GRU,** Gated Recurrent Unit

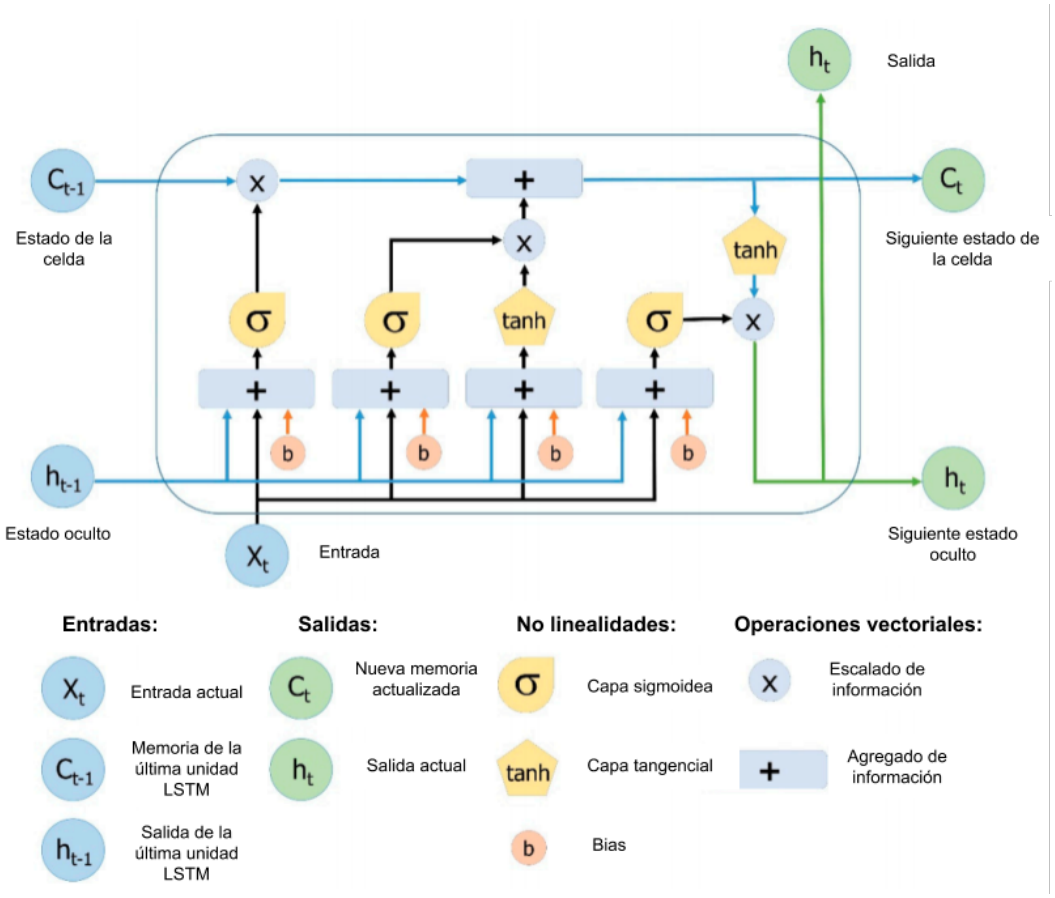


FIGURA 4.8: Estructura de LSTM. Reproducido desde Yan [**54**].

que fluye hacia 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.

4.5. 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.

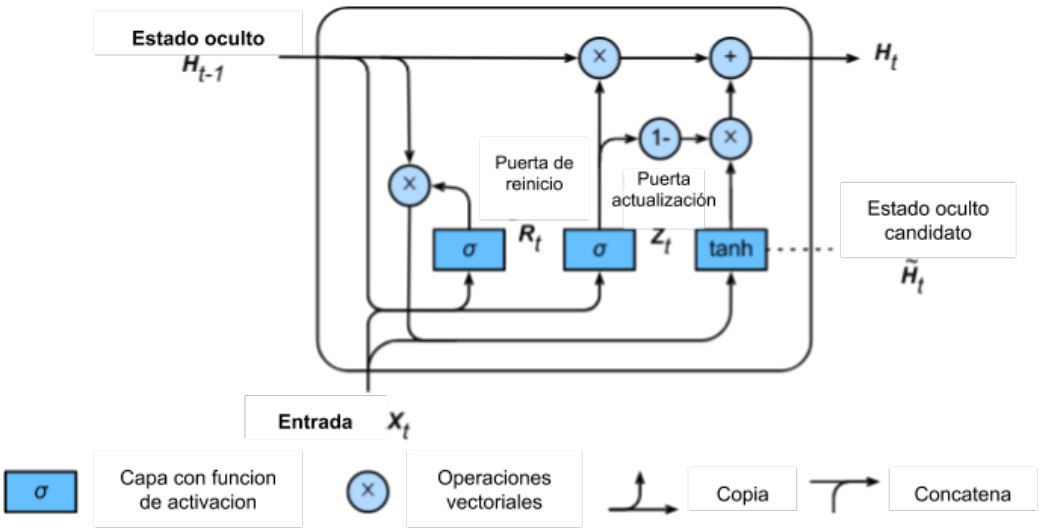


FIGURA 4.9: Estructura de GRU. Reproducido desde [**56**].

4.5.1. One-Class SVM

One-Class SVM[8] (OC-SVM) es un enfoque ampliamente utilizado para descubrir anomalías de forma no supervisada [**59**]. 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 4.10.

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

4.5.2. 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 et al. y Williams et al. fueron los primeros que propusieron autocodificadores para la detección de anomalías [**57**], [**58**]. Desde entonces, la capacidad de los autoencoders para detectar

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

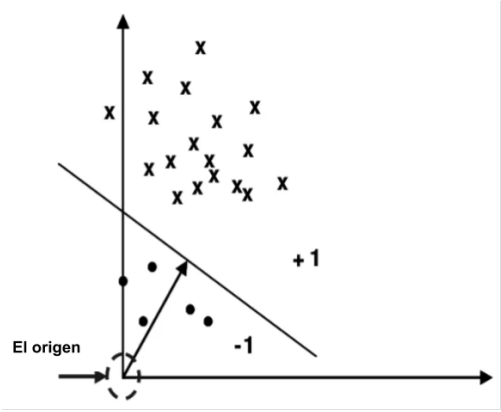


FIGURA 4.10: One-Class SVM

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 podr'a 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 determinar si los datos presentados para la prueba son anómalos **[47]** (Ver Figura 4.11). La ecuación 4.8 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.

$$C(z) = \begin{cases} \text{si } S_z \leq \text{Umbral} & \text{Comportamiento normal} \\ \text{si } S_z > \text{Umbral} & \text{Anomalía} \end{cases} \tag{4.8}$$

4.5.3. Isolation Forest

Este modelo se usa en un escenario similar al de una clase SVM, 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

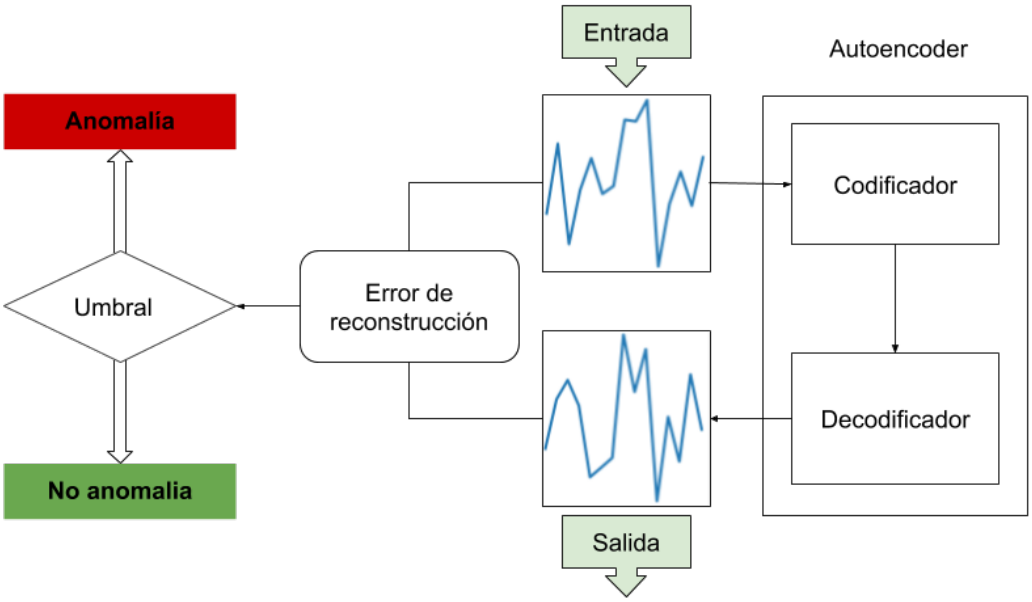


FIGURA 4.11: Detección de anomalías con autoencoder

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.

La Figura 4.12 ilustra el algoritmo y cómo se espera que un punto de datos anómalo se aísle rápidamente, mientras que se espera que un punto de datos normal necesite más particiones.

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.

4.6. 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.

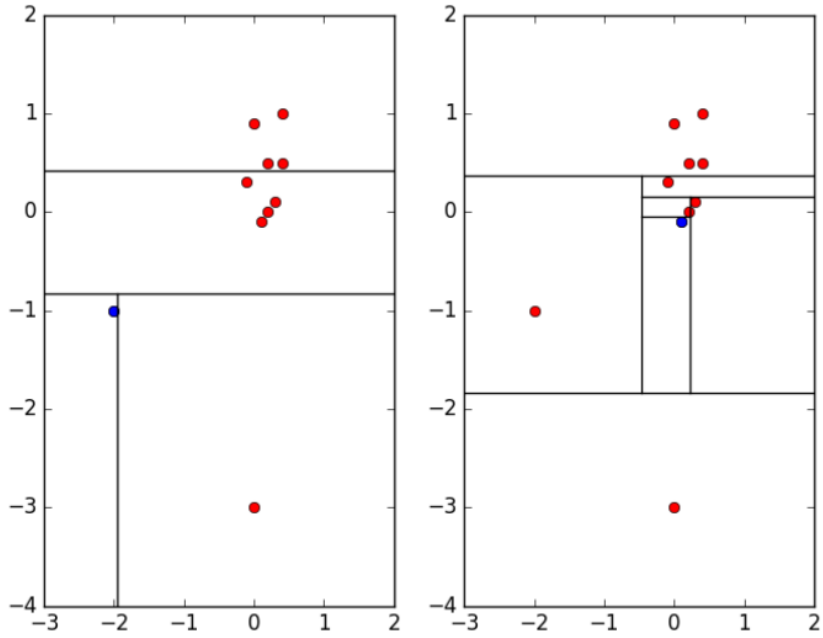


FIGURA 4.12: La figura de la izquierda muestra el aislamiento de una anomalía, que requiere solo tres particiones. A la derecha, el aislamiento de un punto normal requiere seis particiones [**60**].

4.6.1. 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.

$$\text{Precisión} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones realizadas}} \tag{4.9}$$

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.

4.6.2. 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

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

CUADRO 4.1: Matriz de confusión, para una clasificación binaria.

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:

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * log(p_{ij}) \tag{4.10}$$

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.

4.6.3. 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 **[61]**. 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 las clasificación y de ellas se derivan diferentes medidas estadísticas.

En el Cuadro 4.1 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 4.11. Este valor es una medida de clasificación como un todo, ya que indica la probabilidad de que una muestra se clasifique correctamente.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.11}$$

La exactitud no es una medida adecuada para la evaluación de algoritmos de detección de valores atípicos, debido a que la mayoria 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.

4.6.4. Área bajo la curva (AUC)

Área bajo la curva (AUC[9]) es una de las métricas más utilizadas para la evaluación. Se utiliza para problemas de clasificación binaria.

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

Tasa de Verdaderos Positivos (TPR)

La tasa de verdaderos positivos (TPR[10]), 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 4.12.

$$Sensibilidad = \frac{VP}{FN + VP} \tag{4.12}$$

Tasa de Falsos Positivos (FPR)

La tasa de falsos positivos (FPR[11]), también conocida como especificidad, corresponde a la proporción de puntos de datos negativos que se consideran erróneamente como positivos, con respecto a todos los puntos de datos negativos. Se define según la ecuación 4.13.

$$Especificidad = \frac{FP}{FP + VN} \tag{4.13}$$

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 umbral 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

[9] **AUC,** Area Under Curve
[10] **TPR,** de sus siglas en inglés True Positive Rate, también conocido como Sensitivity.
[11] **FPR,** de sus siglas en inglés False Positive Rate, también conocido como Specificity.

AUC, que es el área bajo la curva ROC. En la Figura 4.13 se muestra un ejemplo de la curva AUC-ROC.

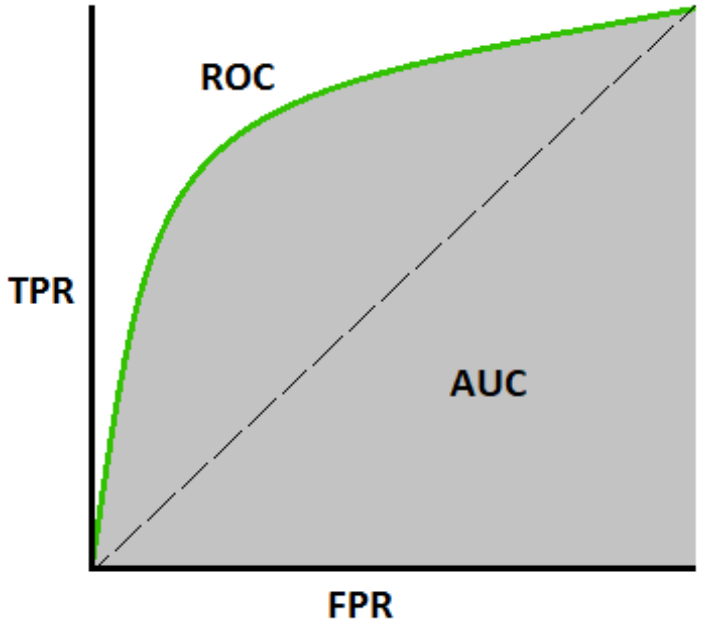


FIGURA 4.13: Ejemplo de un curva AUC-ROC **[62]**.

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

4.6.5. 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.

$$Precision = \frac{VP}{VP + FP} \tag{4.14}$$

Recuperación (Recall)

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

$$Recall = TPR = \frac{VP}{VP + FN} \tag{4.15}$$

Por lo tanto, F1 Score se expresa matemáticamente como:

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4.16}$$

4.7. Resumen del capítulo

Este capítulo permite complementar los conocimientos necesarios para abordar el desarrollo del método de detección de anomalías de conducción mediante el uso de Aprendizaje Automático. En primer lugar, se decribió los diferentes paradigmas de aprendizaje que existen y los diferentes enfoques de modelos, luego, se realizó una descripción del funcionamiento de las redes neuronales y se mostró sus diferentes tipos, así como también diferentes técnicas de detección de anomalías y por útimo se presenta diferentes métricas de evaluación de modelos de aprendizaje automático. El siguiente capítulo detalla la generación del modelo de detección de anomalías de manejo.

Capítulo 5

Generación del modelo de detección de anomalías de conducción

Este capítulo describe el proceso seguido para la generación del modelo de detección de anomalías de manejo. Según lo repasado en el capítulo anterior, en el presente trabajo, se propone un método de detección de anomalías de conducción siguiendo un enfoque semi-supervisado; sin embargo, antes de profundizar en el método propuesto se debe realizar un repaso del conjunto de datos con el que se cuenta en la investigación.

5.1. Conjunto de datos normales y anómalos

En el Capítulo 3 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**, los cuales corresponden a aquellos datos que no cuentan con una etiqueta.

A pesar de contar con una gran cantidad de datos normales se debe recolectar datos que correspondan a anomalías con el objetivo de poder validar el método que se propone en este proyecto. Por lo que se debió proceder a la captura de **datos anómalos**, el cual está conformado según el Cuadro 5.1.

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

Tipo de anomalía	Nro anomalías	Nro de datos
Frenos en seco	5	100
Giros a la derecha e izquierda a alta velocidad	5	450
Giros en U a alta velocidad	5	300

CUADRO 5.1: Tabla del conjunto de anomalías.

5.1.1. Generación de series temporales

Para la generación del modelo detector de anomalías se optó por algoritmos de aprendizaje automático enfocados en series de tiempo, dado que los datos capturados por el dispositivo móvil, dependen del tiempo en el que fueron capturados; por lo cual el primer paso a realizar es la generación de pequeñas fracciones de series temporales, esto permitirá que el modelo propuesto vaya más allá de una simple detección de anomalías puntuales y pueda detectar anomalías contextuales o colectivas.

En el Cuadro 5.2 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 uno y dos pasos; debido a que no son lo suficientemente descriptivos. En cuanto a las series de tiempo restantes no es posible definir aún cual es la cantidad correcta de pasos, por lo cual al igual que la cantidad de componentes principales, será un parámetro a optimizar en los diferentes experimentos que se realizará en las siguientes secciones. Cabe recalcar que el dominio de ésta variable estará entre 3 a 5 pasos y el de la cantidad de componentes principales estará entre 3 y 4.

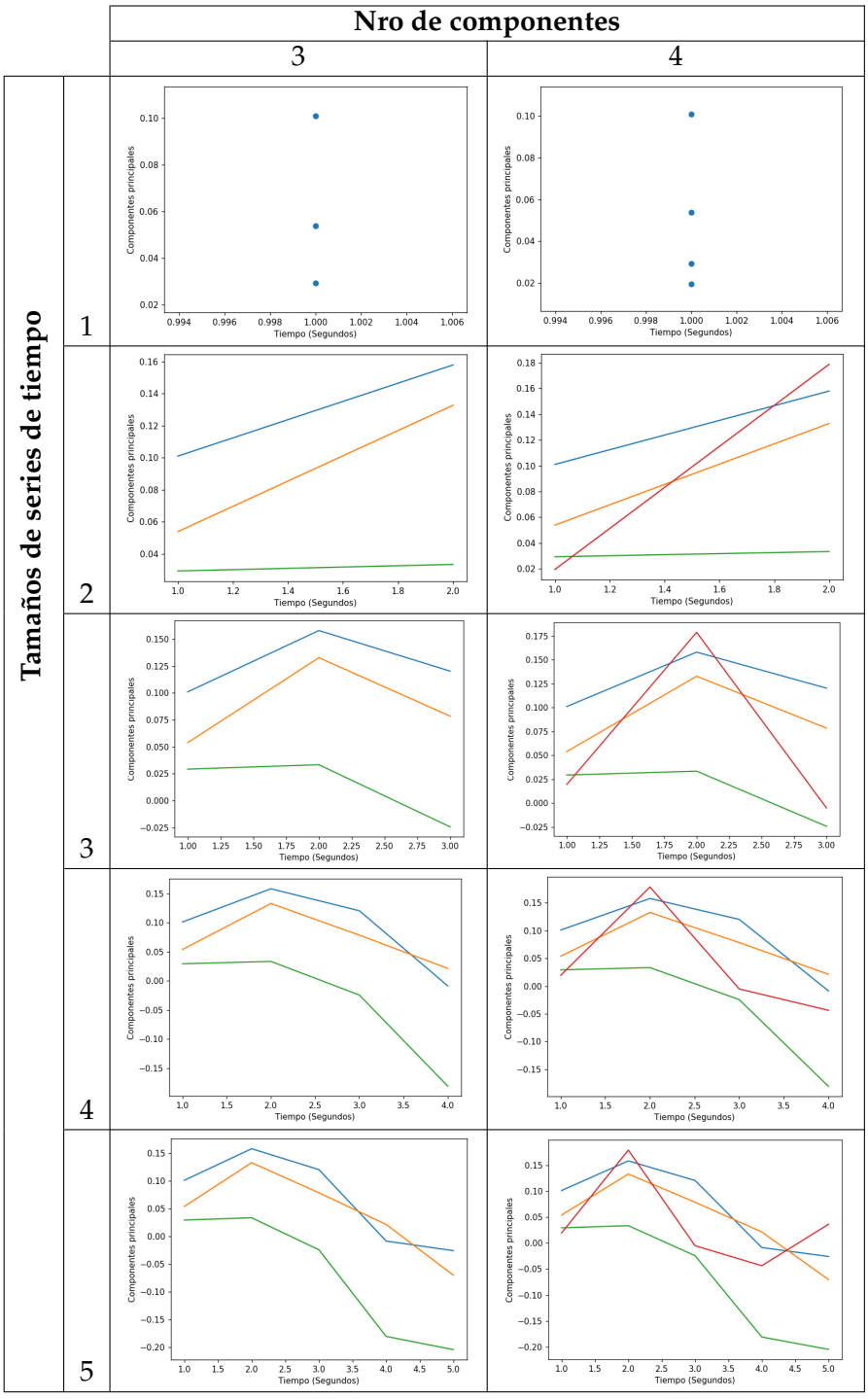
5.2. Modelo de detección de anomalías

Como se mencionó previamente en la presente investigación se propone un método de detección de anomalías de conducción siguiendo un enfoque semi-supervisado, el cual consta de dos componentes: un **modelo ajustado al comportamiento normal** de manejo de un agente y un **método de umbral para la detección de valores atípicos**.

Para la presente investigación se hará la comparación entre 3 diferentes métodos de detección, y según el rendimiento de cada uno se elegirá la mejor opción. En el Cuadro 5.3 se presenta los tres diferentes métodos que serán comparados y como se puede observar para todos los casos se usa un autoencoder como modelo del comportamiento normal, por lo que las siguientes secciónes se enfocarán en la descripción del entrenamiento del autoencoder, para posteriormente probar los 3 diferentes tipos de detección de anomalías.

Método	**Descripción**
AE_T	Método de detección basado en autoencoders y umbralización (Thresholding)
AE_IF	Método de detección basado en autoencoders y la aplicación de Isolation Forest sobre la codificación obtenida por el autoencoder
AE_OC-SVM	Método de detección basado en autoencoders y la aplicación de One-Class SVM sobre la codificación obtenida por el autoencoder

CUADRO 5.3: Tabla de los métodos comparados.



CUADRO 5.2: Tabla con diferentes tamaños de series de tiempo para 3 y 4 componentes principales.

5.2.1. Modelo del comportamiento normal

Esta etapa es una de las partes más importantes de éste trabajo debido a que el rendimiendo 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 manejo. 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 solo se basa el uso de capas densas, 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 6 diferentes tipos de combinaciones de entrada, como se indica en el Cuadro 5.2, es decir, se probó una diferente cantidad de pasos (entre 3 y 5) por cada cantidad de componentes principales (3 y 4). Por lo tanto se realizaron 18 diferentes experimentos, de los cuales por cada tipo de red sobresalio uno (usando la precisión de las redes como tipo de evaluación para desarrollar las comparaciones).

En el Cuadro 5.4 se presenta la mejor red de todas las Redes Densas que se probaron, esta red corresponde a la red que fue alimentada con 3 componentes principales en 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 cuadrada, 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 utiliza *tanh*, esto se debe a que el conjunto de datos, posterior a la obtención de componentes principales, se encuentran en el rango $(1, -1)$.

		Arquitectura Densa			
		NN_33			
		Tipo	**Salida**	**Activación**	**# Parámetros**
PCA	**3**	Input	(3,3)		0
		Flatten	9		0
		Dense	6	elu	60
		Dense	3	elu	21
		Dense	6	elu	24
		Dense	9	tanh	63
		Reshape	(3,3)		0

CUADRO 5.4: Arquitectura densa para una secuencia de 3 pasos y 3 componentes principales

Por otra parte el Cuadro 5.5 se presenta la mejor red de todas las Redes Convolucionales que se probó, esta red al igual que la anterior corresponde a la red que fue alimentada con 3 componentes principales en secuencias de 3 pasos. La arquitectura de esta red consta de una capa de entrada (Input), una combinacion de capas de convolución de una dimensión y agrupación (MaxPooling1D) hasta comprimir los datos a una dimensión de (1,4), luego un conjunto de capas convolucionales y de muestra ascendente (Upsampling1D) para decodificar la información compresa. Cabe recalcar que esta red también usa *tanh* como función de activación de 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 5.5: Arquitectura convolucional para una secuencia de 3 pasos y 3 componentes principales

En el Cuadro 5.6 se presenta la mejor red de todas las Redes Recurrentes probadas, corresponde a la red que tiene como entrada una secuencia de 3 pasos para 3 componentes principales, como en los anteriores casos. Esta 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 5.6: Arquitectura recurrente para una secuencia de 3 pasos y 3 componentes principales

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

Evaluación de autoencoders

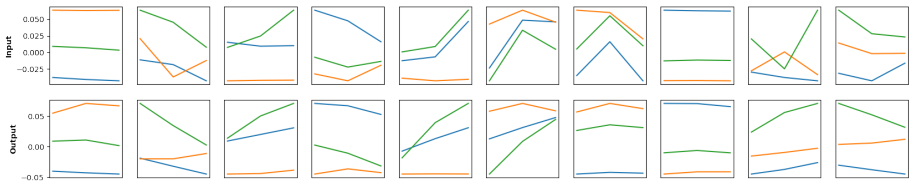
En la anterior sección se presentó los mejores representantes por tipo de red; ahora se procederá a la evaluación y comparación de estos 3 autoencoders para elegir la arquitectura que se ajusta mejor al comportamiento normal de manejo.

En el Capítulo 4 se mostró los diversos tipos de evaluación que existen, para 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 manejo que correspoden a una sola clase "Normal"); por lo tanto a continuación se presenta el Cuadro 5.7 con la precisión de cada red autoencoder, el tiempo de ejecución de cada una y la pérdida logarítmica de cada modelo, en este cuadro se puede apreciar que la red densa **NN_33** tiene una precisión muy similar a la de **RNN_33** y también presentan un valor de pérdida relativamente bajos en comparación a la red **CNN_33**.

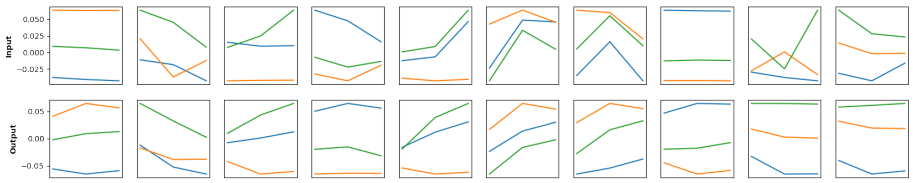
Red	**Precisión**	**Loss**	**Tiempo ejecución**
NN_33	0.85511	0.00764	27us/step
CNN_33	0.81399	0.00971	37us/step
RNN_33	0.85022	0.00635	188us/step

CUADRO 5.7: Evaluación de las redes NN_33, CNN_33 y RNN_33

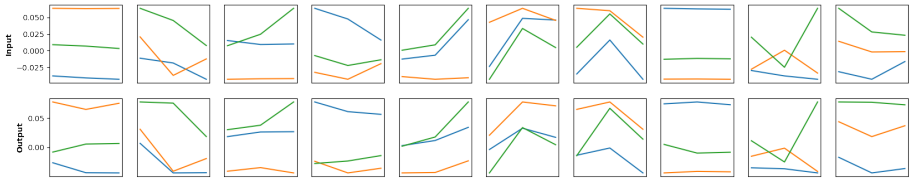
Por otra parte la diferencia más grande que presentan las redes **NN_33** y **RNN_33** es el tiempo de ejecución ya que de la primera es de tan solo 27 segundos/paso y de la segunda 188 segundos/paso, debido a estas similitudes entre ambas redes, se tomó como un paso necesario verificar visualmente las reconstrucciones de cada tipo de red. En



(A) Resultados de la red **NN_33**



(B) Resultados de la red **CNN_33**



(C) Resultados de la red **RNN_33**

FIGURA 5.1: Resultados.

la Figura 5.1 se presenta los resultados de las redes autoencoders para diez secuencias tomadas del conjunto de prueba aleatoriamente, en la parte superior de cada figura se encuentra la secuencia de entrada y en la inferior la reconstrucción del modelo.

Como se puede observar las redes **NN_33** y **CNN_33** obtienen resultados más suavisados de las secuencias, debido a ello muchas veces la reconstrucción no se parece a la secuencia de entrada, por otra parte la red **RNN_33** a pesar de tener un procesamiento más lento obtiene resultados mucho más similares a los de las entradas, lo cual es justo lo que se necesita en el presente trabajo. Por lo tanto, la red **RNN_33** será el modelo ajustado al comportamiento normal de manejo.

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

5.2.2. 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

Para la aplicación de umbralización se usará la reconstrucción del error
, en cuanto para los bosques aleatorios y SVM de una clase se usará el resultado del codificador de los autoencoders para su entrenamiento.

5.3. Evaluación del modelo de detección de anomalías

Capítulo 6

Conclusiones y trabajos futuros

Después de haber realizado el procedimiento descrito en los anteriores capítulos para generar un mecanismo (modelo) capaz de detectar anomalías de conducción, se puede decir que se ha cumplido a cabalidad con los objetivos propuestos en el proyecto. A continuación se presentará las conclusiones a las que se llego 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 de grado.

6.1. Conclusiones

La presente investigación se ha dedicado al

6.2. Limitaciones y recomendaciones

Una vez concluido el trabajo de grado, se considera interesante investigar sobre aspectos para 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, así como también de las anomalías.
- Utilizar tensorflow 2.0 en lugar del framework keras

usar datos que son invriantes al movimiento del celular
realizar mas preubas con otras anomalias

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**
Redes Neuronales - 3 componentes principales	**NN_33**	Input	(3,3)		0
		Flatten	9		0
		Dense	6	elu	60
		Dense	3	elu	21
		Dense	6	elu	24
		Dense	9	tanh	63
		Reshape	(3,3)		0
	NN_43	Input	(4,3)		0
		Flatten	12		0
		Dense	8	elu	104
		Dense	4	elu	36
		Dense	8	elu	40
		Dense	12	tanh	108
		Reshape	(4,3)		0
	NN_53	Input	(5,3)		0
		Flatten	15		0
		Dense	10	elu	160
		Dense	5	elu	55
		Dense	10	elu	60
		Dense	15	tanh	165
		Reshape	(5,3)		1

CUADRO A.1: Arquitectura densa para 3 componentes principales

A.1.2. Redes densas para 4 componentes

		Arquitecturas Densas			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Neuronales - 4 componentes principales	**NN_34**	Input	(3,4)		0
		Flatten	12		0
		Dense	8	elu	104
		Dense	4	elu	36
		Dense	8	elu	40
		Dense	12	tanh	108
		Reshape	(3,4)		0
	NN_44	Input	(4,4)		0
		Flatten	16		0
		Dense	10	elu	170
		Dense	5	elu	55
		Dense	10	elu	60
		Dense	16	tanh	176
		Reshape	(4,4)		0
	NN_54	Input	(5,4)		0
		Flatten	20		0
		Dense	12	elu	252
		Dense	6	elu	78
		Dense	12	elu	84
		Dense	20	tanh	260
		Reshape	(5,4)		1

CUADRO A.2: Arquitectura densa para 4 componentes principales

A.2. Redes convolucionales

A.2.1. Redes convolucionales para 3 componentes

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

CUADRO A.3: Arquitectura convolucional para 3 componentes principales

A.2.2. Redes convolucionales para 4 componentes

		Arquitecturas Convolucionales			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Conv - 4 componentes principales	**CNN_34**	Input	(3,4)		0
		Conv1D	(3,2)	elu	26
		MaxPooling1D	(2,2)		0
		Conv1D	(2,4)	elu	28
		MaxPooling1D	(1,4)		0
		Conv1D	(1,6)	elu	54
		UpSampling1D	(3,6)		0
		Conv1D	(3,4)	tanh	76
	CNN_44	Input	(4,4)		0
		Conv1D	(4,3)	elu	39
		MaxPooling1D	(2,3)		0
		Conv1D	(2,4)	elu	40
		MaxPooling1D	(1,4)		0
		Conv1D	(1,6)	elu	54
		UpSampling1D	(4,6)		0
		Conv1D	(4,4)	tanh	52
	CNN_54	Input	(5,4)		0
		Conv1D	(5,2)	elu	26
		MaxPooling1D	(3,2)		0
		Conv1D	(3,4)	elu	28
		MaxPooling1D	(2,4)		0
		Conv1D	(2,5)	elu	65
		Reshape	(5,2)		0
		Conv1D	(5,4)	tanh	28

CUADRO A.4: Arquitectura convolucional para 4 componentes principales

A.3. Redes recurrentes

A.3.1. Redes recurrentes para 3 componentes

		Arquitecturas Recurrentes			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Rec - 3 componentes principales	**RNN_33**	Input	(3,3)		0
		LSTM	(3,9)	elu	468
		LSTM	6	elu	3840
		Reshape	(3,2)		0
		LSTM	(3,3)	elu	72
		LSTM	(3,9)	elu	468
		TimeDistributed(Dense(3))	(3,3)	tanh	30
	RNN_43	Input	(4,3)		468
		LSTM	(4,9)	elu	384
		LSTM	6	elu	0
		Reshape	(3,2)		216
		LSTM	(3,6)	elu	576
		LSTM	(3,9)	elu	40
		TimeDistributed(Dense(3))	(3,4)	tanh	0
		Reshape	(4,3)		0
	RNN_53	Input	(5,3)		468
		LSTM	(5,9)	elu	384
		LSTM	6	elu	0
		Reshape	(3,2)		72
		LSTM	(3,3)	elu	468
		LSTM	(3,9)	elu	50
		TimeDistributed(Dense(3))	(3,5)		0
		Reshape	(5,3)	tanh	0

CUADRO A.5: Arquitectura recurrente para 3 componentes principales

A.3.2. Redes recurrentes para 4 componentes

		Arquitecturas Recurrentes			
		Tipo	**Salida**	**Activacion**	**# Parametros**
Redes Rec - 3 componentes principales	**RNN_34**	Input	(3,4)		0
		LSTM	(3,9)	elu	468
		LSTM	6	elu	3840
		Reshape	(3,2)		0
		LSTM	(3,3)	elu	72
		LSTM	(3,9)	elu	468
		TimeDistributed(Dense(3))	(3,4)	tanh	30
	RNN_44	Input	(4,4)		0
		LSTM	(4,4)	elu	144
		LSTM	4	elu	144
		Reshape	(4,1)		0
		LSTM	(4,2)	elu	32
		LSTM	(4,4)	tanh	112
	RNN_54	Input	(5,3)		0
		LSTM	(5,15)	elu	1200
		LSTM	10	elu	1040
		Reshape	(5,2)		0
		LSTM	(5,3)	elu	468
		LSTM	(5,6)	elu	240
		TimeDistributed(Dense(4))	(5,4)	tanh	28

CUADRO A.6: Arquitectura recurrente para 4 componentes principales

Bibliografía

[1] DANG-NHAC, L., DUC-NHAN, N., THI-HAU, N., y HA-NAM, N. Vehicle mode and driving activity detection based on analyzing sensor data of smartphones, *Sensors* v. 18, 4 (2018), 1036.

[2] FERREIRA, J., CARVALHO, E., FERREIRA, B., DE SOUZA, C., SUHARA, Y., PENTLAND, A., y PESSIN, G. Driver behavior profiling: An investigation with different smartphone sensors and machine learning, *PLoS One* v. 12, 4 (2017), e0174959.

[3] WHO-LEE, K., SIK-YOON, H., MIN-SONG, J., y RYOUNG-PARK, K. Convolutional Neural Network-Based Classification of Driver's Emotion during Aggressive and Smooth Driving Using Multi-Modal Camera Sensors, *Sensors* v. 18, 4 (2018), 957.

[4] JOHNSON, D., y TRIVEDI, M. Driving Style Recognition Using a Smartphone as a Sensor Platform. En: Intelligent Transportation Systems (ITSC), *14th International IEEE Conference*, 2011. pp. 1609–1615.

[5] KRIDALUKMANA, R., YAN-LU, H., y NADERPOUR, M. An Object Oriented Bayesian Network Approach for Unsafe Driving Maneuvers Prevention System, *12th International IEEE Conference*, 2017.

[6] BHOYAR, V., LATA, P., KATKAR, J., PATIL, A., y JAVALE, D. Symbian Based Rash Driving Detection System. Int. J. Emerg. Trends Technol. Comput. Sci. 2013; 2:124–126.

[7] CHEN, Z., YU, J., ZHU, Y., CHEN, Y., y LI, M. D3: Abnormal Driving Behaviors Detection and Identification Using Smartphone Sensors, *Proceedings of the 12th Annual IEEE International Conference on Sensing, Communication, and Networking*; Seattle, WA, USA. 22–25 de Junio 2015; pp. 524–532.

[8] EREN, H., MAKINIST, S., AKIN, E., y YILMAZ, A. Estimating Driving Behavior by a Smartphone, *Proceedings of the Intelligent Vehicles Symposium*; Alcalá de Henares, España. 3–7 de Junio 2012; pp. 234–239.

[9] BOONMEE, S., y TANGAMCHIT, P. 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 de Mayo 2009; pp. 412–415.

[10] KOH, D.-W., y KANG, H.-B. Smartphone-Based Modeling and Detection of Aggressiveness Reactions in Senior Drivers; *Proceedings of the IEEE Intelligent Vehicles Symposium*; Seoul, Korea. 28 de Junio – 1 de Julio 2015; pp. 12–17.

[11] ZALDIVAR, J., CALAFATE, C.T., CANO, J.C., y MANZONI, P. 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 de Octubre 2011; pp. 813–819.

[12] ARAUJO, R., IGREJA, A., DE CASTRO R., y ARAUJO, R. Driving coach: A smartphone application to evaluate driving efficient patterns; *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV)*; Alcala de Henares, España. 3–7 de Junio 2012; pp. 1005–1010.

[13] SHAI, S., y SHAI, B. Understanding Machine Learning: From Theory to Algorithms (2014).

[14] XUE, Z., SHANG, Y., y FENG, A. Semi-supervised outlier detection based on fuzzy rough C-means clustering. *Mathematics and Computers in Simulation*, vol. 80, no. 9, pp. 1911–1921.

[15] RUMELHART, D. E., HINTON, G. E., y WILLIAMS, R. J. (1986). Learning representations by backpropagating errors. *Nature*, 323(6088), 533–536.

[16] ELMAN, J. Finding structure in time. Cognitive Science (1990), 14(2), 179–211.

[17] WERBOS, P. J. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* (1988), 1(4), pp. 339–356.

[18] PYLE, D. Data preparation for data mining (1999), pp. 90.

[19] SUAD, A., WESAM, S. Review of Data Preprocessing Techniques in Data Mining. *Journal of Engineering and Applied Sciences* 12(16): 4102-4107. 2017.

[20] BISHOP, M. C. Pattern recognition and Machine Learning (2006), pp. 561.

[21] BELLMAN, R. E. Dynamic programming (2003). Courier Dover Publications. *ISBN* 978-0-486-42809-3.

[22] MOINDROT, O. y GENTHIAL, G. Splitting into train, dev and test sets. 24 de Enero en 2018. Disponible en: `https://cs230-stanford.github.io/train-dev-test-split.html`.

[23] HSU, A. y GRIFFITHS, T. Effects of generative and discriminative learning on use of category variability.

[24] DAUPHIN, Y. N., FAN, A., AULI, M., y GRANGIER, D. Language Modeling with Gated Convolutional Networks, *arXiv*, 2017.

[25] MASS, A., HANNUN, A., y NG, A., Rectifier Nonlinearities Improve Neural Network Acoustic Models. *International Conference on Machine Learning (icml)*, 2013.

[26] ZEILER, M. D., RANZATO, M.,MONGA, R.,MAO, M.,YANG, K., LE, Q. V., y HINTON, G. E. On rectified linear units for speech processing. *International Conference on Acoustics, Speech and Signal Processing. IEEE*, 2013, pp. 3517–3521, IEEE.

[27] LECUN, Y., BENGIO, Y., y HINTON, G. Deep learning. *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] GUO, Y., LIAO, W., WANG, Q., YU, L., JI, T. y LI, P. Multidimensional Time Series Anomaly Detection: A GRU-based Gaussian Mixture Variational Autoencoder Approach. *Proceedings of Machine Learning Research* 95:97-112, 2018.

[29] LECUN, Y., JACKEL, L., BOSER, B., DENKER, J., GRAF, H., GUYON, I., HENDERSON, D., HOWARD, R., y HUBBARD, W. Handwritten digit recognition : Applications of neural networks chips and automatic learning. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*. 2014.

[31] BENGIO, Y., SIMARD, P., FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw*. 1994, 5, 157–166.

[32] PASCANU, R., MIKOLOV, T., BENGIO, Y. On the difficulty of training recurrent neural networks. *In Proceedings of the International Conference on Machine Learning*, Atlanta, GA, USA, 16–21 Junio de 2013; pp. 1310–1318.

[33] HOCHREITER, S., SCHMIDHUBER, J. Long short-term memory. *Neural Comput*. 1997, 9, 1735–1780.

[34] OLAH, C. Understanding LSTM Networks. Disponible en la página web: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` (Último acceso en 11 de octubre de 2019).

[35] YAN, S. Understanding LSTM and Its Diagrams. Disponible en la página web: `https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714` (Último acceso en 11 de octubre de 2019).

[36] CHO, K., MERRIËNBOER, B. V., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., y BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv* preprint arXiv:1406.1078, 2014a.

[37] ZHANG, A., LIPTON, Z., LI, M., y SMOLA, A. Dive into Deep Learning, 25 de Septiembre de 2019. Disponible en la página web: `https://en.d2l.ai/d2l-en.pdf` (Último acceso en 11 de octubre de 2019).

[38] HAWKINS, S., HE, H., WILLIAMS, G. y BAXTER, R. Outlier Detection Using Replicator Neural Networks. *International Conference on Data Warehousing and Knowledge Discovery*. pp. 170–180. No. September, Springer Berlin Heidelberg (2002).

[39] WILLIAMS, G. y BAXTER, R. A comparative study of RNN for outlier detection in data mining. *IEEE International Conference on Data Mining* (December 2002), 1–16 (2002).

[40] SCHÖLKOPF, B. y SMOLA, A. J. Support vector machines, regularization, optimization, and beyond. *MIT Press*, pp. 656:657, 2002.

[41] WOLPHER, M. Anomaly Detection in Unstructured Time Series Data using an LSTM Autoencoder. Disponible en la página web: http://www.diva-portal.org/smash/get/diva2:1225367/FULLTEXT01.pdf (Último acceso en 11 de octubre de 2019).

[42] SMITS, P., DELLPIANE, S., y SCHOWENGERDT, R. 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 (1999), pp.1461–1486.

[43] ÖZLER, H. Accuracy Trap! Pay Attention to Recall, Precision, F-Score, AUC. Disponible en la página web: https://medium.com/datadriveninvestor/accuracy-trap-pay-attention-to-recall-precision-
(Último acceso en 16 de octubre de 2019).