Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

Predecir las Preferencias o Necesidades del Conductor de un Vehículo para Ofrecer Confort y Seguridad Usando Aprendizaje Automático

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRA EN SISTEMAS COMPUTACIONALES

Presenta: **EUNICE ALEJANDRA IBARRA GALAVIZ**

Asesor **DR. RIEMANN RUÍZ CRUZ**

Tlaquepaque, Jalisco. agosto de 2019.

AGRADECIMIENTOS

Quiero agradecer a Josef, mi esposo, por haber recorrido este camino juntos. Tu amor, apoyo y ayuda en cada momento fueron fundamentales para llegar a este punto. Agradezco tu paciencia, que hayas compartido tus conocimientos y tu tiempo para obtener este grado juntos.

Agradezco a mi familia y amigos por todo su amor y apoyo. Este trabajo es parte del fruto del tiempo que sacrificamos de pasar juntos, su paciencia y ayuda durante estos dos años.

Agradezco a todos los profesores de las asignaturas que compartieron sus conocimientos y experiencias en cada clase. Ellos constituyeron la base para la adquisición de los conocimientos adquiridos durante estos dos años. Así mismo, agradezco a Riemann Cruz, su asesoría y conocimientos fueron de gran ayuda para la culminación de este trabajo. De igual forma, me gustaría agradecer a mis compañeros, por su ayuda y apoyo durante estos dos años.

Agradezco al CONACYT por la beca recibida número 487137, lo cual fue una contribución para las asignaturas de la Maestría en Sistemas Computacionales.

Agradezco al ITESO por el descuento que me ofreció y también me sirvió como contribución para las asignaturas. Así como por sus instalaciones que siempre me ofrecen un descanso mental, a pesar de la carga laboral y académica.

DEDICATORIA

Dedico esta tesis a Josef, mi familia y mis amigos que me apoyaron durante esta etapa tan ocupada de mi vida.

RESUMEN

Se presenta un sistema para configurar el ambiente de un vehículo que disminuya la carga cognitiva que representa esta tarea para los ocupantes. Dicho sistema tiene como objetivo principal desarrollar una nueva tecnología que mejora la experiencia de usuario dentro del vehículo e indirectamente aumenta la seguridad de los ocupantes del vehículo.

Esto se logra utilizando los siguientes métodos de aprendizaje automático: un sistema de recomendación, un *PageRank* y una red neuronal que componen dicho sistema; tomando como base de conocimiento el comportamiento y preferencias previas de dicho usuario para poder anticipar sus preferencias. En este trabajo se implementaron dichos métodos y se corrieron diferentes experimentos y pruebas con ellos para obtener las ventajas y desventajas del sistema y sus subsistemas.

Como parte de los resultados, este trabajo describe cuál es el rendimiento de cada método utilizado desde el punto de vista de consumo de memoria, tiempo de entrenamiento fuera de línea y tiempo de ejecución del cálculo de la preferencia de un usuario. Estos factores serían determinantes al elegir uno o más modelos que se quisieran implementar hasta una etapa de producción.

Con base en los resultados obtenidos en este trabajo, se puede continuar investigando y desarrollando métodos óptimos y adecuados para las restricciones de la industria automotriz. En el trabajo a futuro se proponen temas, como utilizar dos o más modelos simultáneamente para mejorar la predicción de la preferencia del usuario, para complementar este trabajo.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS Y ABREVIATURAS

AC	Air Conditioner
AFR	Air Fuel Ratio
AIDE	Adaptive Integrated Driver-Vehicle Interface
DVE	Driver Vehicle Environment
DVI	Driver Vehicle Interface
ECU	Electronic Control Unit
ESoP	European Statement of Principals on HMI
EV	Electric Vehicle
GID	Global Identifier
GPS	Global Positioning System
HASTE	Human Machine Interface And the Safety of Traffic in Europe
HMI	Human Machine Interface
I/O	Input/Output
ICA	Interaction and Communication Assistant
IoT	Internet of Things
IoV	Internet of Vehicles
LCD	Liquid Crystal Display
LS-SVM	Least Squares Support Vector Machines
MANET	Mobile Ad-Hoc Network
MAPE	Mean Absolute Percentage Error
MLP	MultiLayer Perceptron
NHTSA	National Highway Traffic Safety Administration
OEM	Original Equipment Manufacturer
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
RFID	Radio Frequency Identification
RMS	Root Mean Square
RoSPA	Royal Society for the Prevention of Accidents
SIoT	Social Internet of Things
SIoV	Social Internet of Vehicles
SR	Sistema de Recomendación
SVM	Support Vector Machines
TOG	Trabajo de Obtención de Grado
UI	User Interface
V2H	Vehicle to Human
V2R	Vehicle to Road
V2S	Vehicle to Sensor
V2V	Vehicle to Vehicle
VANET	Vehicular Ad-Hoc Network
WiMax	Worldwide Interoperability for Microwave Access

WLAN	Wireless Local Area Network

1. INTRODUCCIÓN

En la actualidad los vehículos generan una gran cantidad de datos que están directamente relacionados con las preferencias y necesidades del conductor. Analizar esta información para poder proveer más confort mejoraría la experiencia de usuario. Algunos ejemplos de uso son: personalización del sistema de entretenimiento, diversas aplicaciones en vehículos autónomos, creación de perfiles del conductor y ofrecer sugerencias al conductor para mejor su experiencia de usuario.

El Trabajo de Obtención de Grado (TOG) que se propone es la implementación de un sistema de aprendizaje automático que mediante el análisis de diferentes salidas de los sensores de un automóvil pueda aprender las preferencias y costumbres del conductor para proveer sugerencias acertadas, personalizar el automóvil y mejorar la experiencia de usuario.

El TOG propuesto se alinea con las líneas de investigación y tendencias de desarrollo de la industria automotriz. El resultado del TOG provee una nueva tecnología para mejorar la experiencia de usuario al transportarse y una base para el procesamiento y análisis óptimo de la información que proveen los distintos sensores del vehículo.

1.1. Antecedentes

Una de las metas actuales de las principales compañías de sistemas electrónicos automotrices es contribuir en el desarrollo de vehículos autónomos, los cuáles vienen a revolucionar la transportación. La transformación que se espera abarca distintas áreas, como seguridad y experiencia de usuario. Estos objetivos serán alcanzados mediante el desarrollo de nuevas tecnologías como el aprendizaje automático.

El desarrollo del aprendizaje automático y su aplicación en diferentes áreas se ha extendido rápidamente. Su desarrollo e implementación para aplicaciones automotrices se ha adoptado, aunque aún de forma limitada y experimental. Pero hay una tendencia creciente a su aplicación, a pesar de que la seguridad es un factor crítico para los sistemas automotrices [1].

1.2. Justificación

En la actualidad, la transportación se ha convertido en una necesidad básica de los seres humanos. Conforme las ciudades han crecido, las distancias a recorrer se han hecho más largas y las personas usan un tiempo considerable de su día exclusivamente para este propósito. Además, con el desarrollo de la tecnología, los vehículos proveen más opciones de entretenimiento para que los ocupantes tengan un viaje más placentero. Sin embargo, las interfaces de humano-máquina que existen para las diferentes funcionalidades de los autos no han evolucionado paralelamente. Lo que significa, que para habilitar o deshabilitar una funcionalidad, el usuario tiene que presionar unos cuantos botones. Por lo que, si un usuario quiere personalizar varias funcionalidades cada vez que va a comenzar un viaje, esto le puede tomar un tiempo.

En el mejor de los casos, si un usuario toma su tiempo, podrá personalizar distintas funcionalidades invirtiendo solo un poco de su tiempo antes de comenzar el viaje. Pero en el peor caso, esto se puede convertir en un riesgo, ya que el usuario se distrae del camino para poder lograr la personalización deseada. Por otro lado, existe el caso en el que las condiciones climatológicas, ambientales o inclusive personales van cambiando a lo largo del viaje, lo cual demanda al usuario la modificación de ciertos parámetros como por ejemplo la temperatura del aire acondicionado o bajar el volumen del radio.

De acuerdo con un reporte de la Administración Nacional de Seguridad del Tráfico en las Carreteras (NHTSA por sus siglas en inglés) de Estados Unidos [2], para conducir un vehículo de forma segura, los conductores deben realizar solo tareas primarias. Usualmente los conductores están realizando tareas secundarias o innecesarias mientras conducen que representan distracciones significativas y previenen de la conducción segura de un vehículo. Ya sea que las distracciones ocurran dentro o fuera del vehículo, o requieran de las manos, ojos o mente del conductor; estás actividades roban la atención necesaria del conductor para operar el vehículo de forma segura.

Por otro lado, además de la seguridad que esto provee, el confort que se le puede ofrecer a los usuarios de un vehículo es un sustento mercadológico para el desarrollo de tecnología que mejore la experiencia de usuario dentro de un vehículo.

1.3. Problema

Los vehículos modernos ofrecen diferentes funcionalidades para proveer confort a los ocupantes. Algunos ejemplos son el ajuste de la temperatura del aire acondicionado y calefacción, los calentadores de asientos con ajuste de temperatura, conectividad Bluetooth con el teléfono móvil, diferentes fuentes de música, control automático de los limpiaparabrisas, encendido y apagado automático de las luces del vehículo y diferentes vistas en el tablero dependiendo de la información que se quiera mostrar.

Para controlar cada uno de estos sistemas, existen diferentes controles electrónicos y/o mecánicos que ajustan la configuración deseada por el usuario. Por lo que, para hacer cualquier ajuste, el usuario debe accionar el control y su atención es desviada de la conducción.

Por otro lado, esta variedad de controles provee mayor flexibilidad al momento de configurar los sistemas del vehículo y cubren específicamente la demanda de cada uno de los usuarios, justificando el gran número de botones y controles dentro de un auto. Por tanto, el problema es desarrollar nuevas tecnologías que

permitan simplificar la interfaz humano-máquina dentro de un vehículo para crear un diferenciador de valor respecto a los otros vehículos en el mercado. De esta forma, las compañías que desarrollen y produzcan vehículos usando esta tecnología tendrán mayor ventaja competitiva.

En términos técnicos, se necesita desarrollar el algoritmo para que con base en la información que proveen los diferentes sensores del vehículo, un sistema de sugerencias sea capaz de aprender de la rutina de diferentes usuarios y recomiende configuraciones personalizadas de acuerdo con las condiciones del viaje actual. En forma simple, la entrada del usuario se limitará a un "sí" o "no" para establecer diferentes configuraciones; minimizando la distracción del camino y logrando el máximo confort posible.

1.4. Hipótesis

Dada la cantidad de datos que genera un vehículo por medio de sus sensores, sistemas electrónicos y configuraciones de los usuarios; la información es suficiente para predecir las configuraciones preferidas de un usuario dado el estado actual del vehículo al utilizar métodos de aprendizaje automático.

1.5. Objetivos

1.5.1. Objetivo General

El objetivo de este proyecto es desarrollar una nueva tecnología que mejora la experiencia de usuario dentro de un vehículo y que indirectamente aumenta la seguridad de los ocupantes del vehículo.

1.5.1. Objetivos Específicos

- Identificar cuál es la información que proveen diferentes sensores dentro de un vehículo que es útil para aprender la rutina de los usuarios.
- Implementar un algoritmo que procese la información de los sensores y aprenda la rutina de los usuarios.
- Definir cuál será el estado inicial de un vehículo después de salir de producción y antes de entregar al usuario final.

1.6. Novedad Científica, Tecnológica o Aportación

El gran diferenciador de este trabajo respecto a lo que existe en la actualidad es la innovación respecto a la interfaz humano-máquina. Aunque los controles y botones de los vehículos han evolucionado con los años, aún siguen siendo relativamente similares a lo que existía décadas atrás.

Además, este trabajo proveerá mayor confort a los usuarios de los vehículos y mejorará la calidad del tiempo que una persona usa para transportarse diariamente.

2. ESTADO DEL ARTE O DE LA TÉCNICA

2.1. Interfaz Humano-Máquina para las Configuraciones de Confort de un Vehículo

Actualmente, las configuraciones de confort dentro de un vehículo se realizan a través de uno o más controles. Cada uno de los controles tiene su función específica, por ejemplo, para la temperatura y modo del aire acondicionado, intensidad del calentador de asientos, volumen del audio, fuente del audio, vidrios, etcétera. Además de tener los controles de configuración de confort, los vehículos cuentan con controles para otras configuraciones del vehículo, ya sean de seguridad o control; por ejemplo, control de las luces, asistentes de conducción como el control de crucero, posición de espejos, seguros, etcétera.

Los tipos de controles que existen en los vehículos actuales son:

- Los botones de encendido/apagado son probablemente los más comunes y simples de los controles, pero a la vez su funcionalidad está limitada.
- Los botones giratorios pueden ser multifuncionales, lo que los hace un poco más complejos y por tanto requieren más atención del usuario para ser configurados.
- Las pantallas táctiles son cada vez más populares en los vehículos ya que proveen versatilidad, sin embargo, eso añade complejidad.
- El control por gestos está asociado en su mayoría con las pantallas táctiles.
- El control por voz es menos popular que los anteriores.

Un vehículo tiene al menos diez botones para realizar distintas configuraciones. En las Figura 2.1, Figura 2.2, y Figura 2.3 se pueden observar distintos interiores de vehículos de gama alta que están actualmente en el mercado.

Figura 2.1 Mercedes Maybach 2018 [3]

Figura 2.2 Tesla Model X [1]

Figura 2.3 Audi A8 L W12 [4]

Según los resultados del trabajo de tesis de Janus Yuan en [5], un conductor recibe información del entorno, el vehículo y medios sociales; por lo que una visualización efectiva de la información de conducción y entretenimiento es necesaria para que el conductor pueda enfocar su atención en el camino. Al mismo tiempo se debe tener una interfaz humano-máquina optimizada para mapear todas las funcionalidades de un vehículo accesiblemente como se muestra en la Figura 2.4.

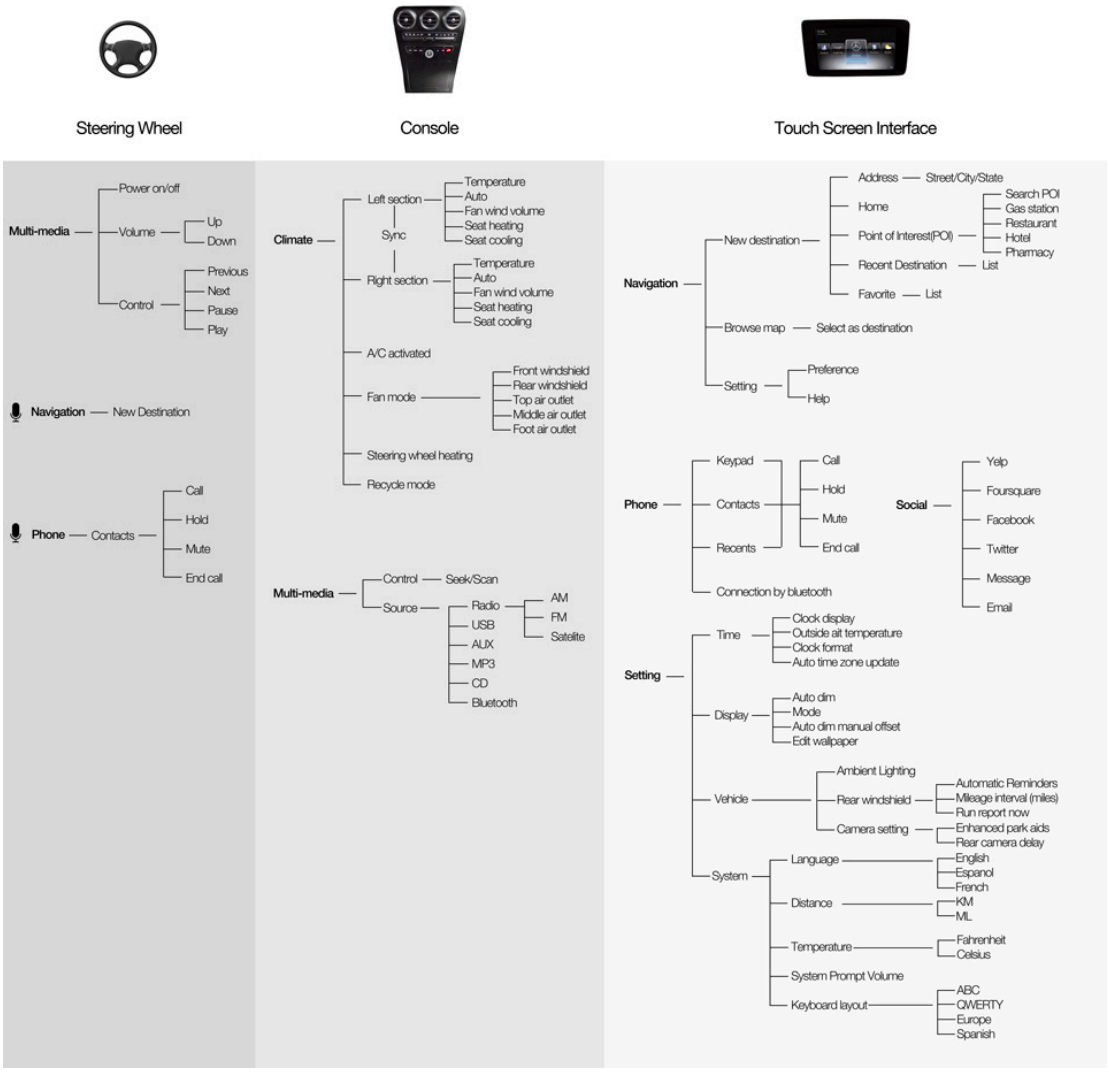

Figura 2.4 Functions in Your Car – Content Mapping [5]

Como resultado de una mesa redonda organizada por *Adaptive Integrated Driver-Vehicle Interface* (AIDE) en el 2008 [6], algunos de los mayores cambios con respecto a la interfaz humano-máquina en los vehículos fueron los controles en el volante, pantallas táctiles, consola central, *head-up display* y control de voz. Los mayores retos para los siguientes años son la integración de múltiples sistemas de asistencia, distracciones del conductor, sobrecarga de información para el conductor, verificación de la seguridad y mejora del confort. Por lo que algunas de las áreas de investigación con mayor potencial son las necesidades del conductor, mejoras en la tecnología usada en las interfaces humano-máquina y manejo de los recursos de las interfaces humano-máquina.

2.2. Investigación de la Interfaz Humano-Máquina para Vehículos Autónomos

Según Elbanhawi et al. en [7], los avances tecnológicos han habilitado el desarrollo de sistemas de interfaz conductor vehículo (*Driver Vehicle Interface* DVI) con el propósito de mejorar la experiencia de viaje en vehículos. El objetivo de DVI es mejorar el confort del pasajero y limitar la distracción del operador.

La divergencia de atención del conductor incrementa el riesgo de choque. La distracción es causada cuando el conductor presta atención a otras actividades, tales como los sistemas multimedia en el vehículo o el uso de dispositivos móviles. La retroalimentación entre el conductor y vehículo y alteraciones del diseño mostraron una influencia positiva en el comportamiento del conductor. La retroalimentación visual del tráfico que se aproxima ha llevado a una mejora del rendimiento del vehículo. La investigación actual en aplicaciones de realidad aumentada proveerá posiblemente métodos para el futuro de DVI.

Investigaciones actuales sugieren que el modo de transportación tiene un efecto significativo en los sentimientos de pasajero. Adicionalmente, el entretenimiento en el vehículo, como la música, ha demostrado que mejora el rendimiento y confort del conductor. El significado del rol que juegan los sistemas DVI en los vehículos aumentará conforme las actividades de control de los humanos se minimicen. En vez de prevenir distracciones, se predice que la DVI actual evolucionará para ser utilizada para aumentar la productividad de los pasajeros y proveer retroalimentación de las maniobras actuales, para mejorar el confort del pasajero.

Existe una carencia de investigación en el tema de confort de los pasajeros con miras en los coches autónomos. El estado del arte actual se resume en los siguientes puntos [7]:

- Se necesita evaluar el confort de los pasajeros en coches autónomos.
- Muchos métodos consideran explícitamente la planeación del confort. Estos se limitan a la investigación de la generación de fuerzas resultantes y el impacto en los pasajeros. Las fuerzas se originan de la trayectoria que el vehículo sigue. La mayoría de los investigadores no considera el mareo por movimiento, seguridad o caminos naturales como factores del confort humano.
- La pérdida del control aumentará la susceptibilidad a mareo por movimiento.
- Se espera que la retroalimentación DVI disminuya el mareo por movimiento.
- Las trayectorias continuas contribuirán a la prevención del mareo por movimiento y el sentimiento natural de los caminos.
- Se requieren suficientes pruebas de los consumidores para identificar el comportamiento ideal del vehículo.
- El aprendizaje automático provee una avenida para la generación de caminos más humanos y el comportamiento del vehículo.
- Se podrá requerir el rediseño de los interiores del vehículo. Eso mejorará la experiencia del viaje conforme disminuye el control humano.
- Los estudios de factores humanos con sujetos de prueba serán esencial para validar modelos de comportamiento y la planeación de algoritmos.

2.3. Estándares para la Implementación de la Interfaz Humano-Máquina de un Vehículo

De acuerdo con la RoSPA [8], un buen sistema HMI va a prevenir una sobrecarga del conductor, es decir, una situación en la que el conductor tenga que responder a muchas cosas a la vez. Dicha sobrecarga puede ser de índole biomecánica o cognitiva. Un conductor debe ser capaz de tomar buenas decisiones basándose en el uso previo de sistemas similares de un vehículo distinto. Si el conductor malinterpreta la retroalimentación del vehículo, es muy posible que sucedan accidentes. Esta es la razón por la cual los

estándares son necesarios. Los estándares pueden ser cualitativos, basados en la opinión y asesoría de un experto como ESoP o cuantitativos, tales como el régimen desarrollado por el proyecto HASTE.

Como resultado de varios subproyectos, discusiones y foros de AIDE, se realizó un reporte [9] en el que se incluyen estándares y guías relevantes para el desarrollo de HMI. Como parte de este reporte se incluye una arquitectura funcional de referencia cuya característica principal es habilitar la administración adaptiva de las interacciones del conductor en términos de la implementación de estrategias adaptivas basadas en el estado del conductor-vehículo-ambiente. Tomando en cuenta que la situación actual y el estado del conductor son la base para diseñar la interacción.

En la Figura 2.5 se presenta la estructura funcional principal de la arquitectura de referencia propuesta por AIDE.

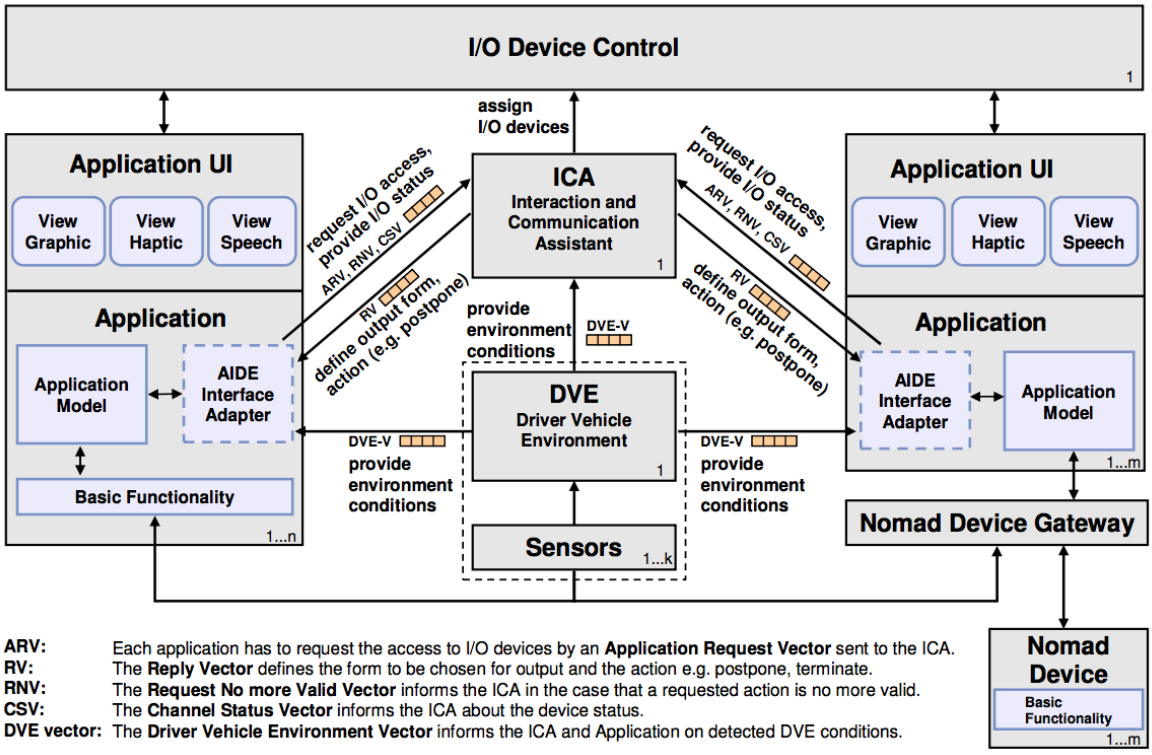

Figura 2.5 Arquitectura funcional de referencia [9]

La arquitectura contiene los siguientes componentes:

- Aplicaciones – estos componentes ofrecen funcionalidades específicas al usuario. La aplicación debe ser lo más independiente posible y debe trabajar independiente de las funciones de administración de la HMI.
- Dispositivos de entrada/salida – estos incluyen todos los dispositivos de entrada/salida como *displays* LCD, head-up displays y bocinas, pero también unidades de pre- y post-procesamiento.
- *Interaction and Communication Assistant* (ICA) – este es el componente que tiene la responsabilidad central de las funciones de administración de la HMI, es decir, contiene las reglas que determinan el comportamiento sistemático global hacía el usuario.

- Módulos *Driver-Vehicle-Environment* (DVE) – este componente es el responsable de proveer retroalimentación del estado DVE, la cual es usada por el ICA para adaptar la interacción conductor-sistema. También es usado por la aplicación para adaptar funcionalidades específicas de la misma.
- Puertas de dispositivos nómadas – esta puerta constituye el enlace entre dispositivos nómadas y los sistemas del vehículo, con el propósito de intercambiar información entre ellos.

De acuerdo a AIDE, esta arquitectura es exitosa para implementar HMI adaptivas para varias aplicaciones. AIDE recomendó que esta solución se utilizara para futuras guías de desarrollo de HMI adaptivas.

2.4. Información Generada en un Vehículo

El uso de sistemas electrónicos en el diseño y desarrollo de vehículos ha incrementado en los últimos años. La tendencia actual indica que seguirá incrementando, ya que los sistemas electrónicos de control son más baratos, ligeros y rápidos de implementar en comparación con un sistema mecánico o hidráulico que provee la misma funcionalidad. Lo cual significa que la cantidad de unidades de control electrónico, ECU por sus siglas en inglés, dentro de un vehículo seguirá aumentando.

Un ECU dentro de un vehículo genera información correspondiente a la funcionalidad que implementa y dicha información es compartida con otros ECUs. Un vehículo contiene una o más redes de ECUs interconectados, a través de esas redes se intercambian la información que genera cada uno. La información solo es compartida con aquellos ECUs que la necesitan y dicha dependencia se define desde el diseño de la red.

La información transmitida por un ECU puede ser el resultado de procesamiento de datos internos o provenir de una fuente externa, como un sensor. La siguiente lista nombra algunos ejemplos de información que se genera en un vehículo:

- Temperatura dentro del auto
- Temperatura fuera del auto
- Modo del AC/calefacción (cara, pies, desempañante)
- Temperatura del AC
- Temperatura de calefacción
- Temperatura/nivel de calefacción de asientos
- Llamada de emergencia
- Velocidad del auto
- Nivel de batería
- Nivel de combustible
- Control de crucero (velocidad, habilitado)
- Presión de las llantas
- Configuración del clúster
- Información de GPS
- Sistema multimedia: volumen, fuente de audio, mute activado/desactivado, reproducción aleatoria activado/desactivado, etcétera.

De acuerdo con el reporte de McKinsey & Company de movilidad con valor agregado presentado en [10], para el 2030 se espera que el negocio de información generada por vehículo y movilidad compartida crezca a 1.5 trillones de dólares. Un vehículo se comunicará con diferentes entidades: conductor, pasajeros, proveedores de servicios, proveedores de servicios móviles, autoridades, infraestructura, OEM, otros vehículos, casa, lugar de trabajo y "gigantes de la tecnología". Además de la gran cantidad de información que se genera día a día, este reporte muestra que la mayoría de las personas están dispuestas a compartir información de su movilidad en comparación con la información que comparten con otro tipo de aplicaciones y redes sociales como se muestra en la Figura 2.6.

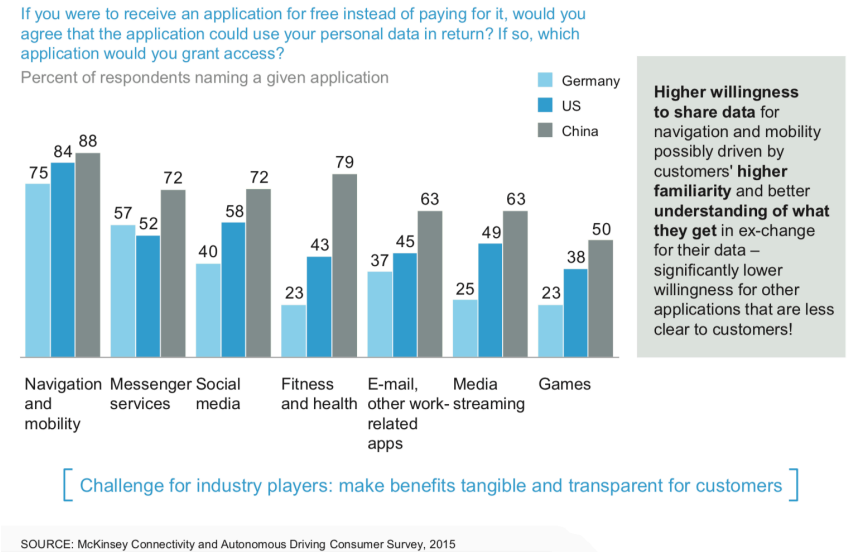

Figura 2.6 Clientes de distintas áreas geográficas están dispuestos a compartir su información de navegación. [10]

Los beneficios de compartir los datos generados por los vehículos se dividen en cuatro áreas principales:

- Seguridad – llamadas de emergencia, prevención de accidentes y advertencias tempranas acerca de peligros en el camino.
- Conveniencia – mantenimiento preventivo, servicios de conserje en el vehículo y mejor experiencia de usuario.
- Tiempo – reducir tiempo de navegación y estacionamiento.
- Costo – reducción de costos de seguros, infraestructura automatizada y reducción de costo de movilidad.

McKinsey estima que una población de 1.2 billones de personas pasa 50 minutos en su vehículo y esa cantidad es mayor en ciudades con mayor cantidad de tráfico. Por lo que se puede concluir que los vehículos modernos generan suficiente información para ser utilizada con distintos propósitos.

2.5. Internet de los Vehículos

Según Sadiku et al. en [11], con el aumento de la población urbana y la expansión rápida de las ciudades, se ha incrementado la cantidad de vehículos. También ha habido un incremento de vehículos eléctricos (EV), ambos totalmente eléctricos e híbridos. Existe una necesidad de mejores comunicaciones e interconectividad entre estos vehículos a causa de su movilidad. Conforme los vehículos han evolucionado

de un simple medio de transporte a entidades inteligentes con capacidad de censado y comunicación, se han convertido en parte integral de las ciudades inteligentes.

Los vehículos inteligentes tienen cinco características: autónomos, conducción segura, conducción social, vehículos eléctricos y aplicaciones móviles.

El Internet de las cosas (IoT por sus siglas en inglés) es una red global que conecta objetos inteligentes y los habilita para comunicarse entre ellos. Cuando esos objetos inteligentes interconectados por Internet son exclusivamente vehículos, entonces IoT se convierte en Internet de los Vehículos (IoV por sus siglas en inglés). Por lo que, IoV es una aplicación extendida de IoT en el área de transportación inteligente. Su visión es servir como sensor de datos esenciales y plataforma de procesamiento para los sistemas de transportación inteligente. Un vehículo será una plataforma de censado, absorbiendo información del ambiente, otros vehículos, el conductor y usándola para una navegación segura, control de contaminación y manejo de tráfico.

IoV consiste en vehículos que se comunican entre sí, así como con dispositivos de los peatones, unidades en los caminos y las redes públicas usando interconectividad V2V (vehículo-a-vehículo), V2R (vehículo-al-camino), V2H (vehículo-a-humano) y V2S (vehículo-a-sensor) para crear una red social dónde los participantes son objetos inteligentes en vez de seres humanos. Esto lleva a la convergencia del Internet Social de los Vehículos (SIoV por sus siglas en inglés). SIoV es esencialmente una instancia vehicular del IoT Social (SIoT). IoV se puede considerar como un conjunto de la Red Vehicular Ad-hoc (VANET por sus siglas en inglés) el cuál se originó de la Red Móvil Ad-hoc (MANET por sus siglas en inglés). IoV extiende la escala, estructura y aplicación de VANET.

Según Chowdhary et al. en [12], se estima que para el 2025 los vehículos podrán conducirse autónomamente mejor que los humanos, con menor tiempo de espera, menos contaminación y mayor confort. La comunicación IoV se ha establecido con respecto a cuatro aspectos definidos por la variable X a continuación. La información se disemina entre vehículos-a-X dónde X es vehículos, unidades en el camino, humanos e internet. Basados en los valores X, se definen tres tipos diferentes de redes: intra-vehículos, inter-vehículos e internet-vehículos como se muestra en Figura 2.7. Los vehículos en IoV emplean identificación de radio frecuencia (RFID) y sensores para recolectar una cantidad enorme de datos y analizarlos. La comunicación del vehículo en IoV se deriva de las VANETs. Adicionalmente, las unidades a bordo y los teléfonos inteligentes se han agregado y trabajan en colaboración con la infraestructura de VANET para monitorear el tráfico, compartir información de los vehículos y carga de vehículos eléctricos.

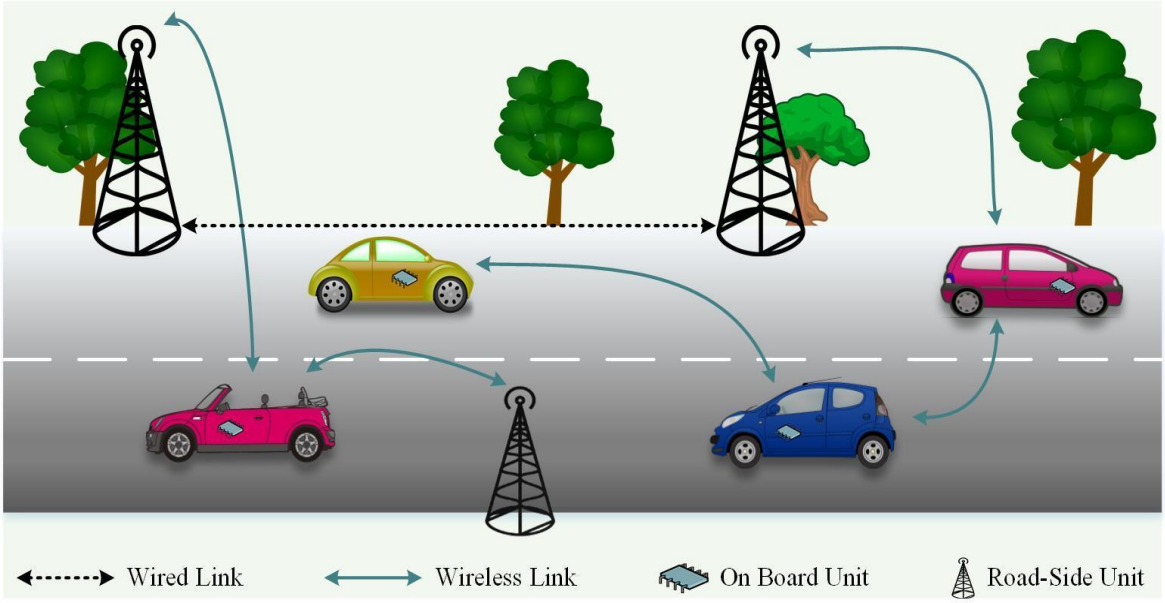

Figura 2.7 Escenario típico de IoV [12]

2.5.1. Arquitectura de IoV

Según Sadiku et al. en [11], una arquitectura típica de IoV consiste en tres capaz como se muestra en la Figura 2.8:

1. Capa de percepción. Esta capa contiene todos los sensores dentro de un vehículo para juntar datos ambientales y detectar eventos de interés específicos como patrones de conducción, condiciones ambientales, etc. Además, tiene identificación de radiofrecuencia (RFID), posicionamiento satelital, ambiente de los caminos, posición del vehículo, percepción de otros carros y objetos, etc.
2. Capa de red. Esta es la capa de comunicación que asegura la conectividad a redes de comunicación como GSM, G5, WiMax, WLAN, Wi-Fi y Bluetooth. Puede soportar diferentes modos de comunicación inalámbrica como V2V, V2I, V2P y V2S.
3. Capa de aplicación. Esta capa incluye herramientas estadísticas, soporta almacenamiento e infraestructura de procesamiento. Es la responsable de almacenar, analizar, procesar y tomar decisiones acerca de las situaciones de riesgo como un congestionamiento de tráfico, mal clima, etc. Esta capa representa aplicaciones inteligentes, seguridad de tránsito, eficiencia y multimedia basada en infoentretenimiento.

Para la presencia en línea de los vehículos, cada vehículo debe tener un identificador único de Internet. Una terminal ID global (GID por sus siglas en inglés) es el núcleo de IoV. El GID provee a los vehículos de una cyber-placa o cyber-ID.

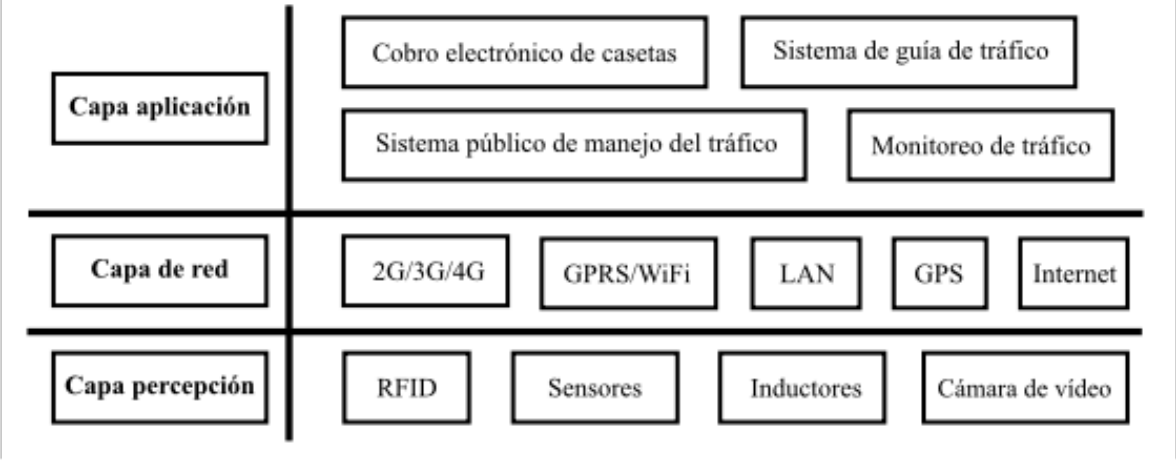

Figura 2.8 Arquitectura de IoV [11]

2.5.2. Aplicaciones de IoV

Según Sadiku et al. en [11], IoV tiene diversas aplicaciones, entre las cuáles se incluyen:

1. Conducción segura: esto se refiere a los sistemas cooperativos para la prevención de choques que usan sensores para detectar colisiones inminentes y advierten al conductor. Esta aplicación involucra mensajes de estatus periódicos y mensajes de emergencia. Un mensaje de emergencia se dispara por un evento de emergencia como una congestión de tráfico, accidente o mala condición del camino.
2. Control de tráfico: IoV provee cambios fundamentales para el manejo de los congestionamientos urbanos, transportación y logística, tránsito vehicular y un estilo de vida colectivo.
3. Respuesta en accidentes: los vehículos conectados pueden enviar información automáticamente y en tiempo real de accidentes y su ubicación a equipos de emergencia. Esto podría salvar vidas al acelerar el tiempo de respuesta ante una emergencia.
4. Servicios convenientes: la habilidad de acceder remotamente a un vehículo hace que servicios como abrir remotamente la puerta o recuperar un vehículo robado estén disponibles. La tecnología de los vehículos conectados puede proveer a las agencias de transportación información en tiempo real del tráfico, tránsito e información de estacionamientos; permitiendo que el manejo de sistemas de transportación sea más eficiente y se reduzca el tráfico y congestionamientos vehiculares.
5. Infoentretenimiento: los vehículos conectados pueden proveer opciones en línea para entretenimiento en el tablero del vehículo.

Otras aplicaciones pueden ser casetas electrónicas, sistemas de guía de tráfico para navegación segura, vehículos autónomos, control inteligente del vehículo, prevención de accidentes, monitoreo del flujo vehicular y autonomía de un vehículo.

2.6. Aprendizaje Automático

Según Adomavicius et al. en [13], existe gran diversidad de algoritmos y métodos de aprendizaje automático en la actualidad. Estos son utilizados en una amplia variedad de aplicaciones como comercio

electrónico, entretenimiento, médicas, bancarias, investigación, vehículos autónomos, procesamiento de lenguaje natural, visión por computadora, seguridad, etc.

Actualmente en el área de comercio electrónico y entretenimiento, el problema de recomendación se define en su forma más simple como la estimación de calificación de artículos o productos que el usuario no ha probado aún. Intuitivamente esta estimación se basa en las calificaciones dadas por el mismo usuario a otros artículos o productos y alguna información extra. Una vez que se ha estimado la calificación de los artículos o productos no calificados, se puede recomendar al usuario aquellos artículos con la calificación estimada más alta.

Según Khan et al. en [14], en años recientes ha surgido el interés por descubrir conocimiento a través del procesamiento de lenguaje natural utilizando técnicas de aprendizaje automático. La intención del procesamiento de lenguaje natural es estudiar métodos y algoritmos para construir modelos computacionales que sean capaces de analizar lenguajes naturales para realizar tareas comunes como habilitar la comunicación entre humanos y máquinas, mejor la comunicación entre humanos o simplemente lograr el procesamiento de texto a voz. El conocimiento lingüístico contiene ambigüedades. Para lograr la desambiguación se utilizan métodos de aprendizaje automático.

Según Zhen et al. en [15], las plataformas en línea como redes sociales juegan un rol importante en el intercambio de información y conocimiento en la actualidad. Sin embargo, hay usuarios maliciosos que realizan acciones fraudulentas como spam, rumores y vandalismo; por lo que son una amenaza de seguridad para los usuarios legítimos. Por lo que muchas plataformas en línea han implementado herramientas y mecanismos para proteger a los usuarios legítimos. Esto ha atraído la atención de la comunidad de investigadores para desarrollar mecanismos que detectan comportamientos fraudulentos.

Según Mirsky et al. en [16], las redes neuronales se han convertido en una solución muy popular para los sistemas de detección de intrusos en una red. Su capacidad de aprender patrones complejos y comportamientos las hace una solución óptima para diferenciar entre tráfico normal y ataques en una red. Sin embargo, un problema con las redes neuronales es la gran cantidad de recursos que necesitan para ser entrenadas. Además, muchos dispositivos de red, como los *routers*, no tienen suficientes recursos para entrenar o correr dichos modelos. Aún más importante, las redes neuronales existentes son entrenadas con supervisión. Esto quiere decir que un experto tiene que etiquetar el tráfico y actualizar periódicamente el modelo de forma manual.

2.6.1. Sistemas de Recomendación

Según McKinsey & Company en [17], los SR en aplicaciones web y de comercio electrónico son utilizados para sugerir distintos productos a los usuarios. Las preferencias de todos los usuarios de dichas plataformas contribuyen al refinamiento de las sugerencias para que estas puedan ser acertadas y lleven al usuario a un nivel de satisfacción, que dicho de otra forma motivan al usuario a seguir utilizando la plataforma ya sea para comprar productos, elegir contenidos o utilizar un servicio.

El uso de SR es crítico para algunas de las plataformas web más exitosas de la actualidad. Ya en el 2013, el 35% de las compras que se realizan en Amazon y el 75% del contenido visto en Netflix es producto de algoritmos sofisticados que analizan las transacciones de su base de datos y tendencias de medios digitales.

Existen distintas razones [18] por las cuales los proveedores de servicios implementan SR:

- Incrementar el número de artículos vendidos.
- Vender artículos más diversos.
- Incrementar la satisfacción de usuario.
- Incrementar la fidelidad de un usuario.
- Comprender mejor que quieren los usuarios.
- Estudios muestran que los usuarios pierden interés después de 60 a 90 segundos de búsqueda, ya que los humanos somos muy malos escogiendo de una variedad amplia, el reto del SR es que el usuario encuentre algo atractivo que ver y entienda porque lo quiere ver [19].

Según Aggarwal et al. en [20], los SR son aplicados en distintos dominios, como ventas, contenido de música, contenido de vídeo, búsquedas web, consulta de datos y anuncios publicitarios. Algunos de estos dominios requieren métodos especializados para adaptar los SR. Todas las aplicaciones en los distintos dominios son centrados en la web. Un aspecto importante de los SR es que se asume que existe un mecanismo eficiente de identificación del usuario para poder seguir e identificar sus intereses a largo plazo. En muchos dominios web, no siempre se cuenta con un mecanismo de identificación de usuario eficiente. En esos casos, las recomendaciones directas por usuario no están disponibles. Además, nuevos elementos (como anuncios publicitarios) continuamente aparecen y desaparecen del sistema, por lo que hay que buscar un método apropiado para esos casos.

Algunos ejemplos de los SR más populares son el de Netflix, Amazon, Youtube, Facebook [18], LinkedIn [18], etc.

Netflix utiliza una colección de algoritmos para crear la experiencia Netflix completa: clasificación personalizada de los vídeos, vídeos top, tendencias actuales, continuar viendo, recomendaciones con base en lo visto anteriormente y variedad para cuentas compartidas [19].

El SR de Amazon elige una cantidad pequeña de artículos, de su catálogo de cientos de millones de artículos, que el usuario disfrutará basándose en su contexto y comportamiento pasado [21].

El objetivo del SR de Youtube es presentar recomendaciones personalizadas al usuario que le permitan encontrar vídeos de alta calidad relevantes a sus intereses. Para mantener entretenidos a los usuarios, es imperativo que las recomendaciones se actualicen regularmente y reflejan su actividad reciente en el sitio [22].

Algunas redes sociales, como Facebook y LinkedIn, usan SR basados en confianza para explotar las relaciones de confianza dentro de la red social del usuario para implementar nuevos algoritmos de recomendación [18].

2.6.2. PageRank

Según Leskovec et al. en [23], uno de los mayores cambios en nuestras vidas en la década que le siguió al cambio de siglo es la disponibilidad de búsquedas eficientes y acertadas en la Web, a través de motores de búsqueda como Google. Aunque Google no fue el primer motor de búsqueda en el mercado, si fue el primero que pudo derrotar a los *spammers* quienes hacían las búsquedas casi inútiles. Sobre todo, la innovación que proveyó Google fue un avance tecnológico no trivial llamado *PageRank*.

Había muchos motores de búsqueda antes que existiera Google. En su mayoría, funcionaban monitoreando la *Web* y listando los términos que se encontraban en las páginas *web* (palabras o cadenas de caracteres distintos a espacio) en un índice invertido. Un índice invertido es una estructura de datos que facilita encontrar todos los lugares donde se encontraban los términos.

Cuando hacía una búsqueda de un término, se extraían todos los sitios *web* que contenían esos términos del índice invertido y eran ponderados según la frecuencia de aparición de dicho término.

Conforme la gente comenzó a usar los motores de búsqueda para encontrar sitios en la *Web*, algunas personas con poca ética vieron la oportunidad de engañar a los motores de búsqueda para que accedieran a sus sitios *web*. Independientemente si el sitio se relacionaba con el término de la búsqueda o no, su único interés era que sus sitios web tuvieran visitas.

La técnica que engañaba a los motores de búsqueda para que creyeran que un sitio *web* es acerca de un tema distinto se le llama *spam* de término. Dada la implementación de esta técnica, los motores de búsqueda se volvieron prácticamente inútiles. Para combatir esto, Google introdujo dos innovaciones:

1. *PageRank* se usaba para simular dónde los navegadores de la *Web*, iniciando desde un sitio web aleatorio, tenderían a converger si seguían algunos enlaces externos aleatoriamente desde la página que se encontraban actualmente e iteraban muchas veces en este proceso. Los sitios *web* que tenían mayor cantidad de visitas son consideradas más importantes con respecto a las que casi no serían visitadas. Google prefiere desplegar los sitios web más importantes como respuesta de una petición de búsqueda.
2. El contenido de un sitio web no solo es juzgado por los términos que aparecen en él, sino que se usan los términos en los enlaces dentro y cercanos. Nótese que, aunque es fácil para un *spammer* añadir términos falsos en un sitio *web* que controlan, es difícil añadir términos falsos en los sitios web que tienen un enlace a su sitio, si es que ellos no controlan esos sitios.

Es razonable preguntarse por qué una simulación de navegadores aleatorios nos permite aproximarnos a la noción de importancia de un sitio *web*. Hay dos motivos que inspiran esta aproximación.

- Los usuarios de la *Web* "votan con los pies". Se tiende a poner enlaces a sitios web que son buenos o útiles para mirar, en vez de enlaces a sitios malos o inútiles.
- El comportamiento de un navegador aleatorio indica cuáles sitios de la *Web* son más probables a ser visitados. Es más probable que un usuario visite páginas útiles que inútiles.

Independientemente de la razón, el algoritmo *PageRank* ha probado empíricamente que funciona.

2.6.3. Predicción de Series de Tiempo con Redes Neuronales

Según Simeone et al. en [26], las series de tiempo financieras son aquellas que registran eventos relacionados con finanzas. Los cambios de valores en las series de tiempo financieras son el resultado de un fenómeno complejo. El efecto de dicho fenómeno es visible como una mezcla de pendientes y saltos en una gráfica de la serie de tiempo. Las variaciones más notables son la tendencia, las variaciones cíclicas y periódicas, y las variaciones día-a-día. La tendencia es una variación gradual identificada a largo plazo en la serie del tiempo. Usualmente esta se predice con una extrapolación. Las variaciones cíclicas y periódicas siguen patrones por temporada o ciclos de negocio en la economía. Las variaciones diarias y a

término corto aparentemente son aleatorias y difíciles de predecir, pero estas son usualmente la fuente para pérdidas y ganancias de transacciones financieras.

En el sector financiero, las series de tiempo se basan usualmente en una extrapolación y cualquier desviación por más pequeña que sea, puede llevar a tomar decisiones incorrectas, perdiendo oportunidades y dinero que pueden no solo afectar a un individuo, sino a un sistema económico completo. Las predicciones suelen ser acertadas solo por un periodo corto de tiempo, sin embargo, los sistemas requieren estrategias a largo plazo para tomar decisiones pronto.

Tradicionalmente, se han usado modelos de predicción estocásticos, basados en análisis estadístico de las series del tiempo. recientemente la atención se ha enfocado en métodos de redes neuronales para la predicción de series de tiempo. Esto es porque las deficiencias de los modelos lineales eran bien conocidas. Las redes neuronales tienen la ventaja de ser fáciles de usar.

En los 90's había mucho escepticismo acerca de si las redes neuronales podrían o no mejorar los modelos lineales y técnicas tradicionales. Ahora se ha acumulado suficiente evidencia para superar el escepticismo.

El método de series de tiempo también se ha utilizado en aplicaciones automotrices como la predicción del comportamiento de un sistema automotriz basándose en la información extraída de algunos sensores del vehículo: temperatura del refrigerante del motor, relación aire-combustible (AFR) de la combustión interna y el voltaje de la batería del automóvil. Se utilizan modelos como el perceptrón multicapa (MLP), máquina de vectores de soporte (SVM) y máquinas de vectores de soporte *least squares* (LS-SVM). Mostrando como resultado que la combinación de modelos es una aproximación prometedora en el contexto de datos automotrices [24].

Además, en el área de transportación, las series de tiempo se utilizan para analizar la complejidad del tráfico. Este estudio es útil para entender el sistema de transporte y su evolución, para predicción y control [25]. En [25] el autor plantea como trabajo a futuro encontrar una red que permita obtener una guía eficiente para el tráfico con el propósito de controlar efectivamente las congestiones de tráfico y mejorar su capacidad significativamente.

3. MARCO TEÓRICO/CONCEPTUAL

3.1. Aprendizaje Automático

El término de aprendizaje automático se introdujo como una alternativa al método convencional de ingeniería para el diseño de soluciones algorítmicas. Como se ilustra en Figura 3.1, en (a), el flujo convencional de diseño ingenieril comienza con la adquisición del conocimiento del dominio: El problema en cuestión es estudiado en detalle, produciendo un modelo matemático que captura la física del objeto de estudio. Basado en el modelo, un algoritmo optimizado es producido que ofrece garantía de desempeño bajo la premisa que el modelo basado en la física es una representación acertada de la realidad.

Como ejemplo, el diseño de un algoritmo de decodificación para un canal inalámbrico intermitente bajo el método convencional de ingeniería va a requerir que el desarrollo, o selección, de un modelo para el canal conectando el transmisor y receptor. La solución se obtendría encontrando una optimización del problema y ofrece garantía de ser óptimo solo para el modelo del canal específico.

En contraste, en su forma más básica, el modelo de aprendizaje automático sustituye el paso de adquirir conocimiento del dominio con la tarea potencialmente más sencilla de recolectar una muestra lo suficientemente grande de ejemplos en los cuáles esté representado el comportamiento deseado para el algoritmo en cuestión. Estos ejemplos constituyen el conjunto de entrenamiento. Como se ilustra en Figura 3.1, en (b), los ejemplos del conjunto de entrenamiento alimentan un algoritmo de aprendizaje para producir una "máquina" entrenada que ejecuta la tarea deseada. El aprendizaje es posible al seleccionar un conjunto de posibles "máquinas", también conocidas como clase de hipótesis, de la cual el algoritmo de aprendizaje hace una selección durante el entrenamiento. Un ejemplo de una clase de hipótesis es una arquitectura de una red neuronal con pesos sinápticos aprendidos. Los algoritmos de aprendizaje están basados generalmente en la optimización del criterio de desempeño que mide que tan bien la "máquina" se comporta con los datos disponibles.

Para el problema de diseñar un decodificador de canal, el método de aprendizaje automático puede operar aun cuando no se ha definido un modelo acertado del canal. De hecho, es suficiente con tener una cantidad suficientemente grande de ejemplos de las señales recibidas – las entradas de la máquina decodificadora – y los mensajes recibidos – las salidas deseadas de la máquina de decodificación – para ser usados en el entrenamiento de la clase de funciones de decodificación.

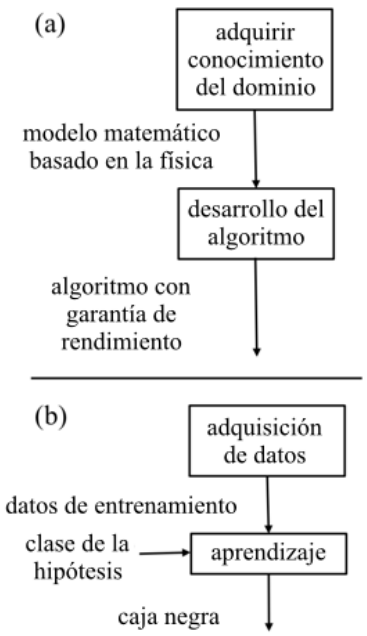

Figura 3.1 (a) Flujo convencional de diseño de ingenieril; (b) metodología base de aprendizaje automático [26]

Según Simeone et al. en [26], además de la formulación básica descrita en los párrafos anteriores, las herramientas de aprendizaje automático pueden integrar el conocimiento del dominio disponible en el proceso de aprendizaje. Esta es la clave del éxito en las herramientas de aprendizaje automático de muchas aplicaciones. Un ejemplo notable es el procesamiento de imágenes, en el que el conocimiento de la invariancia traslacional de las características visuales se refleja en la adopción de redes neuronales convolucionales para el entrenamiento de la clase de hipótesis.

Existen tres clases principales de técnicas de aprendizaje automático.

- Aprendizaje supervisado: el conjunto de entrenamiento consiste en pares de entradas y salida deseada y el objetivo es que en el aprendizaje se crea una relación entre los espacios de entradas y salida. En las aplicaciones se incluye el decodificador de canal discutido anteriormente, así como la clasificación de correo no deseado en el correo electrónico con base en los ejemplos de correo deseado y no deseado.
- Aprendizaje no supervisado: el conjunto de entrenamiento consiste en entradas no etiquetadas, esto es las entradas sin la asignación de una salida esperada. El aprendizaje no supervisado tiene como objetivo descubrir propiedades del mecanismo que genera los datos. Entre sus aplicaciones se incluye el agrupamiento de documentos con temas similares.
- Aprendizaje por refuerzo: este se encuentra entre el aprendizaje supervisado y el no supervisado. A diferencia del aprendizaje no supervisado, existe cierta supervisión, pero esta no es la especificación de la salida deseada para cada entrada. Al contrario, un algoritmo de aprendizaje por refuerzo recibe retroalimentación del ambiente solo después de haber seleccionado una salida para una entrada dada. La retroalimentación indica el grado con el cuál la salida, conocido como

aprendizaje por refuerzo, cumple las metas del aprendiz. El aprendizaje por refuerzo se aplica a problemas de toma de decisión secuencial en los cuáles el aprendiz interactúa con el ambiente ejecutando acciones secuencialmente – las salidas – con base en sus observaciones – las entradas – mientras que recibe retroalimentación de cada una de las acciones seleccionadas.

La mayoría de las aplicaciones actuales de aprendizaje automático se encuentra en la categoría de aprendizaje supervisado, por lo cual se enfocan en aprender los patrones que existen entre las entradas y las salidas. El aprendizaje supervisado y sus beneficios implementados en herramientas algorítmicas se entiende relativamente bien en un nivel teórico. El aprendizaje no supervisado hasta el momento ha enfrentado un tratamiento teórico unificado. Sin embargo, posee un problema fundamentalmente práctico: el reto de aprender por observación directa sin ninguna retroalimentación explícita. El aprendizaje por refuerzo ha encontrado una extensa aplicación en problemas que se caracterizan por tener señales de retroalimentación claras, como salidas de ganar o perder en juegos, y eso desencadena una búsqueda de posibles finales en grandes árboles de acción-observación.

3.1.1. Sistemas de Recomendación

Los SR son herramientas de software y técnicas que proveen sugerencias para ser usadas por los usuarios. Las sugerencias se relacionan con distintos procesos de toma de decisiones como qué producto comprar, qué música escuchar o qué noticias leer [18]. Los SR están diseñados para ayudar a usuarios que no tienen la experiencia o conocimiento suficiente para tomar una decisión, así que se utiliza la retroalimentación de muchos otros usuarios para determinar una sugerencia con alta probabilidad de que también les guste a otras personas.

La importancia del Internet para realizar transacciones electrónicas y de negocios ha permitido el desarrollo de la tecnología para los SR. Uno de los catalizadores principales es la facilidad con lo que se puede proveer retroalimentación como “Me gusta”, “No me gusta” o calificación con estrellas. La idea básica de los SR es utilizar estas fuentes de información para inferir los intereses de los usuarios.

En un SR, la entidad a quién se refiere la recomendación se denomina “usuario” y el producto que se recomienda es conocido como “elemento”. Por lo cual, el análisis de las recomendaciones se basa frecuentemente en interacciones previas entre los usuarios y elementos, porque las preferencias históricas son buenos indicadores de preferencias futuras [20].

Los SR se clasifican en los siguientes tipos [13]:

- Recomendaciones con base en contenido. A los usuarios se les recomiendan artículos similares a los elegidos anteriormente.
- Recomendaciones colaborativas. A los usuarios se les recomiendan artículos que personas con gustos similares eligieron en el pasado.
- Recomendaciones híbridas. Es una mezcla de recomendaciones con base en contenido y colaborativas.

El principio básico de las recomendaciones es la existencia de una dependencia significativa entre la actividad del usuario y los elementos. En muchos casos, las categorías de elementos pueden presentar correlaciones significativas para hacer recomendaciones más acertadas; sin embargo, las dependencias

también se pueden presentar entre elementos individuales. Las dependencias se pueden aprender de una matriz de calificaciones y el modelo resultante se usa para predecir las preferencias de diferentes usuarios. Cabe mencionar que entre más información se tenga de un usuario, será más probable hacer predicciones robustas del comportamiento futuro de dicho usuario.

Los modelos de SR más avanzados utilizan información contextual como información del tiempo, conocimiento externo, información de localización, información social o información de la red.

3.1.1.1. Modelado de un SR

Se debe considerar la aplicación en la que será utilizada el SR, ya que esta determina cuál es la aproximación algorítmica que se tomará [18]. En [27] se clasifican algunos de los sistemas existentes en sus aplicaciones más comunes:

- Entretenimiento – películas, música y contenido de televisión.
- Contenido – periódicos personalizados, documentos, sitios web, aprendizaje en línea y filtros de correo electrónico.
- E-Comercio – recomendaciones de productos.
- Servicios – viajes, consultas expertas o casas en renta.

Dado que los SR son evaluados por el asertividad de sus recomendaciones, lo cual es subjetivo al gusto y estado actual de una persona, no existe una regla para el diseño de un SR. Se debe realizar un análisis de las necesidades del sistema, las circunstancias que influencian las recomendaciones y las recomendaciones esperadas para determinar cuál es el algoritmo óptimo para resolver ese problema.

El propósito del modelo basado en similitud de usuarios es encontrar usuarios con preferencias similares de acuerdo con la calificación que dan a diferentes artículos.

El primer paso de este método es calcular primero el promedio de las calificaciones de un usuario utilizando la ecuación (3.1) [20].

$$\mu_u = \frac{\Sigma_{k \in I_u} r_{uk}}{|I_u|} \quad \forall u \in \{1 \dots m\} \tag{3.1}$$

Donde μ_u es el promedio de calificaciones del usuario u, r_{uk} es la calificación del usuario u e I_u es el conjunto de muestras de entrada del usuario u.

A continuación, en la ecuación (3.2) [20] se calcula la similitud entre los diferentes usuarios de la población completa utilizando la correlación de Pearson. Esto determina cuáles usuarios tienen gustos similares para poder utilizar sus calificaciones como referencia para hacer recomendaciones.

$$Sim(u,v) = \frac{\Sigma_{k \in I_u \cap I_v}(r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\Sigma_{k \in I_u \cap I_v}(r_{uk} - \mu_u)^2} \cdot \sqrt{\Sigma_{k \in I_u \cap I_v}(r_{vk} - \mu_v)^2}} \tag{3.2}$$

Donde $Sim(u,v)$ es la similitud entre la muestra u y v, μ_u es el promedio de calificaciones del usuario u, r_{uk} es la calificación del usuario u, μ_v es el promedio de calificaciones del usuario v, r_{vk} es la calificación del usuario v, I_u es el conjunto de muestras de entrada del usuario u e I_v es el conjunto de muestras de entrada del usuario v.

Para obtener los resultados descartando las diferentes escalas utilizadas por los usuarios, es decir, las calificaciones centradas en el promedio se usan para el cálculo de la recomendación mostrado en la ecuación (3.3) [20].

$$s_{uj} = r_u - \mu_u \quad \forall u \in \{1 \dots m\} \tag{3.3}$$

Donde s_{uj} es la calificación centrada en el promedio del usuario u, μ_u es el promedio de calificaciones del usuario u, r_u es la calificación del usuario u.

Finalmente, usando los resultados parciales de los pasos anteriores, la recomendación para un usuario se calcula utilizando la ecuación (3.4) [20].

$$\hat{r}_{uj} = \mu_u + \frac{\Sigma_{v \in P_u(j)} Sim(u,v) \cdot s_{vj}}{\Sigma_{v \in P_u(j)} |Sim(u,v)|} \tag{3.4}$$

Donde $\hat{r}_{uj}$ es la recomendación para el usuario u, μ_u es el promedio de calificaciones del usuario u, $Sim(u,v)$ es la similitud entre la muestra u y v y s_{vj} es la calificación centrada en el promedio del usuario v.

3.1.1.2. Características de un SR

En esta sección se definen algunas de las características principales que se deben analizar para poder determinar cuál es el algoritmo más apropiado para modelar un SR. Dependiendo de la aplicación específica, se puede seleccionar una o varias de las siguientes [18]:

1. Preferencias de usuario: Las preferencias de los usuarios se refieren a encontrar cuáles son los SR utilizados en aplicaciones similares que los usuarios consideran más acertados. De esta forma se puede tener una noción de que algoritmo es el más adecuado.

2. Exactitud de la predicción: Medir la exactitud de la predicción es un tema que varía en la literatura ya que es muy subjetivo y no existe una característica que lo relacione uno a uno. Los tres métodos más utilizados son: medir los ratings de la exactitud de las predicciones, medir el uso de las predicciones y medir los rankings.

3. Cobertura: La cobertura puede medirse con base en distintos parámetros: espacio, usuarios y comienzo. La cobertura de espacio se refiere a la proporción de artículos del catálogo completo pueden recomendarse. La cobertura de usuarios se refiere a la proporción de usuarios puede satisfacerse exitosamente. La cobertura del comienzo se refiere a la proporción de artículos o usuarios sin historial que se puede satisfacer.

4. Confianza: La confianza se mide con respecto al usuario, es decir, que tanto confían los usuarios en las predicciones del SR.

5. Fidelidad: La fidelidad se define a la seguridad que tiene un sistema al hacer recomendaciones o predicciones. La medición de fidelidad más común es la probabilidad de que un valor predicho sea cierto.

6. Novedad: La novedad es la capacidad de un sistema de recomendar artículos que el usuario no conoce. La forma más fácil de medirlo es preguntarle al usuario, sin embargo, se puede medir la novedad con estudios fuera de línea.

7. Diversidad: Diversidad se define como lo contrario de similitud. Hay muchos casos de uso en los que es deseado tener diversidad en los resultados, por qué esto permite al usuario conocer distintos artículos más rápidamente.

8. Utilidad: La utilidad se mide con respecto al generador de recomendaciones o dueño de la aplicación. Por ejemplo, en el caso del comercio electrónico, se utilizan SR para aumentar las ganancias de la compañía.

9. Riesgo: En algunos casos, las recomendaciones están directamente ligadas a un potencial de riesgo como en el caso de inversiones. La forma más común de medir el riesgo es considerando la utilidad y varianza asociada a las recomendaciones.

10. Robustez: La robustez es la estabilidad de las recomendaciones generadas cuando se recibe información falsa como entrada. Ya que hay personas que dependen de las recomendaciones generadas por el sistema, este debe tener robustez para soportar distintos ataques.

11. Privacidad: En el caso de los SR colaborativos, los usuarios están dispuestos a contribuir con sus preferencias a cambio de una buena recomendación, sin embargo, no quieren que su información sea compartida con terceros. La privacidad de la información debe ser considerada en el diseño de un SR.

12. Adaptabilidad: Dada lo volatilidad de artículos en algunas aplicaciones, como las noticias, hay sistemas que deben adaptarse rápidamente a los cambios y al mismo tiempo seguir generando recomendaciones acertadas.

13. Escalabilidad: Existe una alta probabilidad que la cantidad de datos crezca conforme para el tiempo, por lo que es importante que el SR sea diseñado para seguir dando recomendaciones correctas, aunque cuando se escale la cantidad de datos.

3.1.1.3. Métodos de Pre-Procesamiento de la Información de Entrada de un SR

Frecuentemente, los datos que son tomados como entrada en un SR necesitan ser pre-procesados previamente a pasar por el algoritmo del SR. Dependiendo del diseño del sistema, los datos necesitarán ser limpiados, filtrados o transformados [18].

Existen distintas técnicas de pre procesamiento para los datos de un sistema de aprendizaje automático. Algunas de las características principales que se busca destacar con el pre procesamiento de datos son [18]:

- Encontrar la similitud entre los datos para mapear las relaciones entre los gustos de diferentes usuarios posteriormente.
- Dada la gran cantidad de información que se genera en la actualidad, se debe encontrar cuál es la frecuencia de muestreo ideal para obtener información suficiente y no filtrar eventos significativos.
- Para optimizar la cantidad de recursos utilizados por el SR, es deseable tener la mayor cantidad de información en el menor espacio posible y en el formato óptimo para que el algoritmo la procese en el menor tiempo posible.
- Los datos deben de proveer información certera, por lo que se debe remover cualquier tipo de ruido.

3.1.1.4. Prueba de un SR

Según Ricci et al. en [18], ya que el modelado de un SR depende de la aplicación específica en que se va a aplicar, las pruebas y métricas de efectividad dependerán de la implementación de dicho sistema.

Uno de los métodos de prueba que existen son experimentos fuera de línea. Esto significa que se utiliza información previamente recolectada, asumiendo que el comportamiento de los usuarios cuando dicha información va a ser recolectada es similar cuando el sistema sea lanzado. Uno de los problemas con este método es que no se puede medir la influencia que tiene el sistema en lo toma de alguna decisión. Por tanto, los experimentos fuera de línea se usan principalmente para seleccionar el algoritmo correcto y posteriormente solo se realizan calibraciones finas.

Otro método de prueba son los estudios de usuarios. Para estos estudios se elige una población de sujetos de prueba y se les pide que realicen diversas acciones que requieran interactuar con el SR. Estos sujetos son observados durante el experimento y se recolectan distintas mediciones cuantificables, como que porción de la tarea fue completada, el asertividad del resultado o el tiempo que tomó realizar la tarea. Además, se pueden realizar preguntas cualitativas antes, durante o después del experimento.

Por último, se utilizan evaluaciones en línea para medir el efecto que tiene un SR. Estas evaluaciones dependen de distintos factores como la intención del usuario, el contexto y la interfaz que presenta la recomendación. Frecuentemente, usuarios seleccionados al azar son redireccionados al sistema en prueba en vez del sistema que normalmente genera las recomendaciones. Este método es el más eficiente dado que la interacción es con usuarios reales, sin embargo, se recomienda pasar por los métodos anteriores para refinar el sistema antes de causar mayor insatisfacción en los usuarios aleatorios.

Para evaluar el SR, se hace un análisis de los resultados de las pruebas utilizando técnicas como valores P, resultados paramétricos, resultados no paramétricos, pruebas múltiples o intervalos de confianza.

3.1.2. PageRank

Según Leskovec et al. en [23], *PageRank* es una función que asigna un número real a cada sitio en la *Web* (o al menos de la *Web* que ha sido explorada y sus enlaces descubiertos). La intención es que entre más alto el *PageRank* de un sitio, más importancia tiene. No hay ningún algoritmo fijo para asignar el *PageRank* de un sitio y de hecho variaciones de la idea básica pueden alterar el *PageRank* relativo entre

dos sitios. Se comienza por definir el básico, idealizado *PageRank*, y se prosigue con modificaciones que son necesarias para resolver problemas reales de la estructura de la *Web*.

La *Web* se puede describir como un grafo dirigido, dónde los sitios son los nodos, y hay una arista entre el sitio *p1* y el sitio *p2* si hay uno o más enlaces de *p1* a *p2*. La Figura 3.2 muestra un ejemplo de una versión diminuta de la *Web* que tiene solo 4 sitios. El sitio *A* tiene enlaces a cada uno de los otros tres sitios; el sitio *B* tiene enlaces a *A* y *D* solamente; el sitio *C* tiene solo un enlace a *A*, y el sitio *D* tiene enlaces a *B* y *C* solamente.

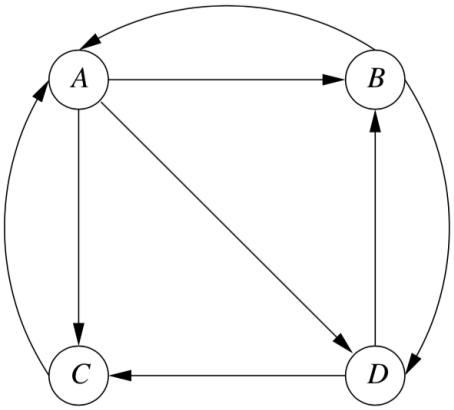

Figura 3.2 Ejemplo hipotético de la Web [23]

Si suponemos que un navegador aleatorio comienza en el sitio *A* de la Figura 3.2. Hay enlaces a los sitios *B*, *C* y *D*, entonces el navegador puede estar en esos sitios con probabilidad de 1/3 y tiene probabilidad de 0 de estar en *A*. Un navegador aleatorio en *B* tiene, en el siguiente paso, probabilidad de 1/2 de estar en *A*, 1/2 de estar en *D* y 0 de estar en *B* o *C*.

En general, se puede definir la matriz de transición de la *Web* para describir que paso con los navegadores aleatorios en el siguiente paso. La matriz *M* tiene *n* filas y columnas, donde *n* es el número de sitios. El elemento m_{ij} en la fila *i* y la columna *j* tienen valor de 1/k si el sitio *j* tiene *k* aristas de salida y una de ellas es el sitio *i*. Sino $m_{ij}=0$.

La matriz de transición para Figura 3.3 es

$$M = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Figura 3.3 Matriz de transición de la Web [23]

En esta matriz el orden de los sitios es el natural, *A*, *B*, *C* y *D*. Por lo que, la primera columna expresa el hecho descrito previamente, que un navegador en el sitio *A* tiene una probabilidad de 1/3 de estar en cualquiera de los otros sitios en el siguiente paso. La segunda columna expresa el hecho que un navegador en el sitio *B* tiene una probabilidad de 1/2 de estar en *A* o *D* en el siguiente paso. La tercera columna dice

que un navegador que está en *C* estará en *A* en el siguiente paso. La última columna dice que un navegador en *D* tiene una probabilidad de 1/2 de estar en *B* o *C* en el siguiente paso.

La distribución de probabilidades para la ubicación de un navegador aleatorio está descrita por un vector columna en el que el componente *j-ésimo* es la probabilidad de que el navegador esté en la página *j*. Esta probabilidad es (idealmente) la función *PageRank*.

Si se supone que un navegador aleatorio comienza en cualquiera de las *n* páginas de la *Web* con la misma probabilidad. Entonces el vector inicial $\mathbf{v_0}$ tendría $1/n$ para cada componente. Si *M* es la matriz de transición de la Web, después de un paso, la distribución del navegador sería $M\mathbf{v_0}$, después de dos pasos sería $M(M\mathbf{v_0}) = M^2\mathbf{v_0}$ y así sucesivamente. En general, multiplicar el vector inicial $\mathbf{v_0}$ por *M* un total de veces *i* dará la distribución del navegador después de *i* pasos.

Para demostrar porque un vector de distribución $\mathbf{v}$ multiplicado *M* veces da una distribución $\mathbf{x} = M\mathbf{v}$ en el siguiente paso, se razonó de la siguiente manera. La probabilidad $\mathbf{x}_i$ que un navegador aleatorio esté en el nodo *i* en el siguiente paso, es $\Sigma_j m_{ij}\mathbf{v}_j$. Aquí m_{ij} es la probabilidad que un navegador en el nodo *j* se mueva al nodo *i* en el siguiente paso y $\mathbf{v}_j$ es la probabilidad que el navegador estuviera en el nodo *j* en el paso previo.

Este comportamiento es un ejemplo de la conocida teoría de procesos de Markov [28] [29]. Es conocido que la distribución del navegador se aproxima a una distribución límite $\mathbf{v}$ que satisface $\mathbf{v} = M\mathbf{v}$, si las siguientes dos condiciones se cumplen:

1. El grafo está conectado fuertemente; esto significa, es posible llegar de un nodo a cualquier otro.
2. No hay finales muertos: nodos que no tienen aristas de salida.

El límite se alcanza cuando se multiplica al multiplicar la distribución por *M* otra vez, ya no hay cambios en la distribución. En otros términos, el límite de $\mathbf{v}$ es un eigenvector de *M* (un eigenvector de una matriz *M* es un vector $\mathbf{v}$ que satisface $\mathbf{v} = \lambda M\mathbf{v}$ para un eigenvalor constante λ). De hecho, dado que *M* es estocástica, lo que significa que cada una de sus columnas suma 1, $\mathbf{v}$ es el eigenvector principal (su eigenvalor asociado es el mayor de todos los eigenvalores). Nótese también que, dado que *M* es estocástica, su eigenvalor asociado con el eigenvector principal es 1.

El eigenvector principal de *M* dice dónde es más probable que el navegador esté después de muchos pasos. Recordando que el principio detrás del *PageRank* es que entre más probabilidades haya de que un navegador esté en una página, más importante es dicha página. Se puede computar el eigenvector principal de *M* iniciando con el vector inicial $\mathbf{v_0}$ y multiplicando por *M* algunas veces, hasta que el cambio en el vector sea despreciable entre cada ronda. En la práctica, para la *Web*, 50 a 75 iteraciones son suficientes para converger dentro del límite de error de aritmética de doble precisión.

Como ejemplo, si se aplica el proceso descrito arriba a la matriz *M* de Figura 3.2. Ya que existen cuatro nodos, el vector inicial $\mathbf{v_0}$ tiene cuatro componentes de 1/4 cada uno. La secuencia de aproximaciones al límite que se obtiene de multiplicar cada paso por *M* es:

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix}, \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix}, \ldots, \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

Figura 3.4 Probabilidades límite

Se puede notar en este ejemplo, que las probabilidades de *B*, *C* y *D* son iguales. Es fácil ver que *B* y *C* siempre van a tener los mismos valores en cualquier iteración, porque sus filas en *M* son idénticas. Se puede utilizar una prueba con el método inductivo para mostrar que sus valores son los mismos que *D*. Dado que los últimos tres valores del vector límite tienen que ser los mismos, es fácil descubrir el límite de la secuencia en Figura 3.4. La primera fila de *M* muestra que la probabilidad de *A* debe ser 3/2 de las otras probabilidades, por lo que el límite tiene la probabilidad de *A* igual a 3/9, o 1/3, mientras que la probabilidad para los otros tres nodos es de 2/9.

La diferencia en probabilidad no es significativa. Pero en la *Web*, con miles de millones de nodos con variaciones importantes, la verdadera probabilidad de estar en un nodo como www.amazon.com es ordenes de magnitud mayor que en los nodos típicos.

3.1.3. Predicción de Series de Tiempo con Redes Neuronales

En general, una serie de tiempo es una colección de datos de alguna variable que cambia en el tiempo y es registrada, típicamente en intervalos regulares de tiempo, por ejemplo, la caída de lluvia mensual en una región o la producción anual de una granja.

La capacidad de predicción refleja el nivel de entendimiento de los procesos que se desarrollan en el ambiente. El método científico induce relaciones causales del análisis cuantitativo de las observaciones. Tal análisis tiene como objetivo establecer la relación causal entre las cantidades observadas (variable dependiente o de salida) y otras cantidades observadas (variables independientes o entradas) del proceso. Las relaciones causales encontradas con este procedimiento forman el modelo del proceso observado. Al no tener modelos confiables, no hay forma de obtener los siguientes valores de la serie hasta que el evento ocurra, por lo que métodos empíricos que permiten la predicción de series de tiempo se usan abundantemente.

Actualmente se utilizan modelos basados en redes neuronales para predecir series de tiempo. Las redes neuronales tienen la ventaja de ser fáciles de usar debido a que existen implementaciones de estas en diferentes lenguajes de programación. Es necesario valorar el rendimiento predictivo y rango de validez de una red neuronal antes de ser usada.

Las series de tiempo tienen componentes o variaciones características. Los cuatro componentes principales son: la tendencia, las variaciones periódicas, las variaciones cíclicas y las fluctuaciones a corto plazo. Se ilustran en Figura 3.5, y se describen de la siguiente manera:

- La tendencia (T) es la componente sutil a largo plazo. Se refiere a la figura general de toda la gráfica de la serie de tiempo.

- Las variaciones cíclicas (C) son las oscilaciones a mediano y largo plazo que se generan alrededor de la tendencia. Estos ciclos pueden ser periódicos o aperiódicos.
- Las variaciones periódicas o de temporada (S) son patrones cíclicos muy regulares.
- Las fluctuaciones a corto plazo (I) son variaciones de poco tiempo usualmente muy irregulares que parecen como ruido o saltos repentinos que se atribuyen a eventos aleatorios. Estos son los más difíciles de predecir.

Estos componentes no son completamente independientes uno del otro.

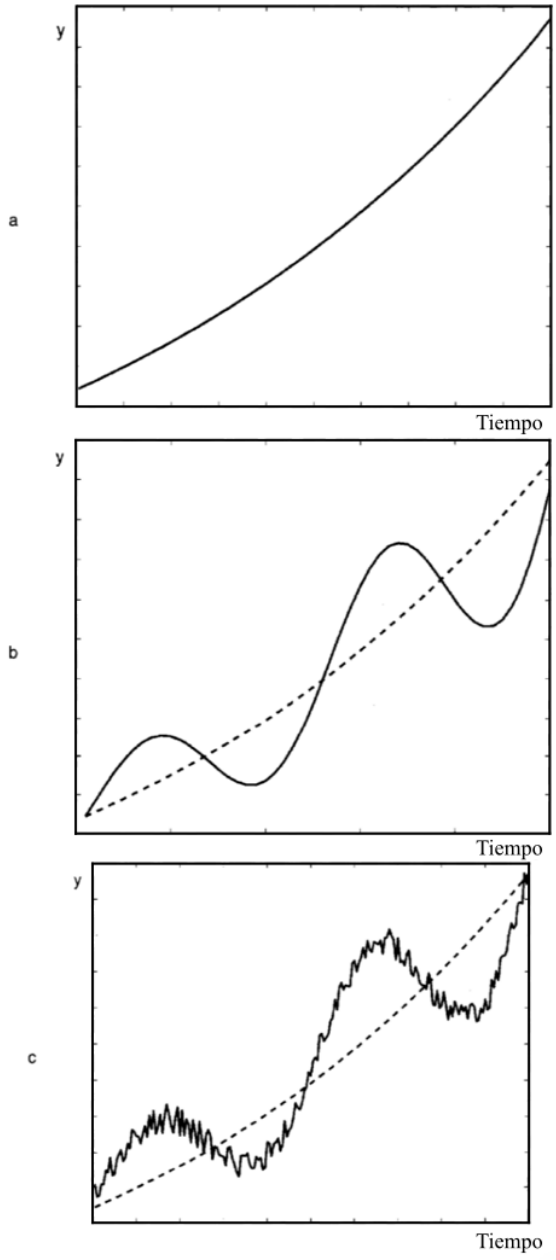

Figura 3.5 Componentes de las series de tiempo: (a) la tendencia, (b) las variaciones cíclicas y periódicas y (c) las variaciones a corto plazo [30]

Las series de tiempo se describen formalmente de dos maneras distintas que son ampliamente aceptadas; se basan en un ensamble de componentes de las series de tiempo que se suman o multiplican [30].

$$Y = T + C + S + I \quad (3.5)$$

$$Y = T \times C \times S \times I \quad (3.6)$$

Donde Y es la predicción de la serie del tiempo, T es la tendencia, C son las variaciones cíclicas, S son las variaciones periódicas e I son las fluctuaciones a corto plazo.

El propósito del análisis de una serie de tiempo es descomponer las series y separar el componente de interés para cualquier propósito de estudio.

Las redes neuronales son útiles por dos razones: primera, son constructos matemáticos genéricos con un número grande de parámetros, llamados pesos, que pueden ser ajustados para representar casi cualquier función multivariable en un amplio rango de sistemas dinámicos. Segunda, dado un conjunto de puntos de prueba de una función desconocida, existen algoritmos para encontrar los pesos de tal forma que la red neuronal se aproxime la función desconocida. Por lo que las redes neuronales son una técnica de modelado dependiente de datos.

Con el propósito ilustrativo se considera una red neuronal *feedforward* para la predicción de series de tiempo. En esta existen tres fases: la fase de entrenamiento, la fase de prueba y la fase de predicción fuera-de-muestras. Para la fase de entrenamiento, secciones de igual longitud de los datos de la serie de tiempo se introducen en la red neuronal, junto con la predicción esperada. Para la fase de prueba, se introduce a la red neuronal nuevas secciones de datos, y su predicción es comparada con el valor conocido de los datos. La etapa de prueba provee una medida de confianza de la predicción. Finalmente, en la fase de predicción fuera-de-muestras la ventana con los valores más recientes de la serie es introducida a la red neuronal para obtener la predicción de un valor futuro desconocido de la serie.

Las redes neuronales artificiales son constructos computacionales inspirados en hechos e hipótesis acerca de procesamiento de información de sistemas naturales. Estas consisten en muchos elementos computacionales simples que están interconectados y se llaman neuronas artificiales. Una neurona artificial tiene muchas entradas, cada una caracterizada por un factor de peso. El aprendizaje ocurre modificando los pesos de las entradas específicas.

Las neuronas artificiales están diseñadas para operar concurrentemente, por lo que las redes neuronales artificiales deben ser estructuras masivas de cómputo paralelo. A pesar de la predominante simulación en secuencias digitales para las redes neuronales artificiales, los diagramas de flujo de las señales en paralelos para redes neuronales artificiales son útiles para entender la parametrización de los algoritmos de una red neuronal. Se muestra una representación de un diagrama de flujo de una red neuronal en Figura 3.6.

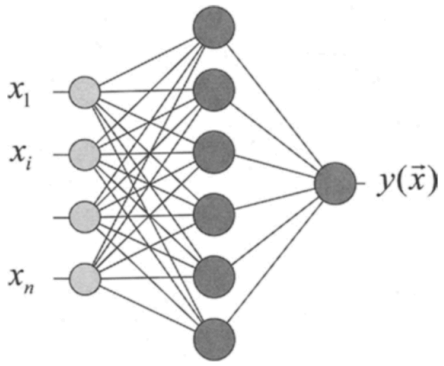

Figura 3.6 Representación de un diagrama de flujo de una red neuronal *feedforward* de muestra. El flujo de las señales va de izquierda (entradas) a derecha (salida). Los círculos más grandes representan neuronas. Cada línea entrante a una neurona tiene asociado un factor de peso. Los círculos más pequeños a la izquierda son los puntos de distribución de las señales de entrada [30]

Las neuronas son dispositivos de múltiples entradas y una salida. La neurona artificial más usada es la neurona McCulloch-Pitts que se muestra en Figura 3.7.

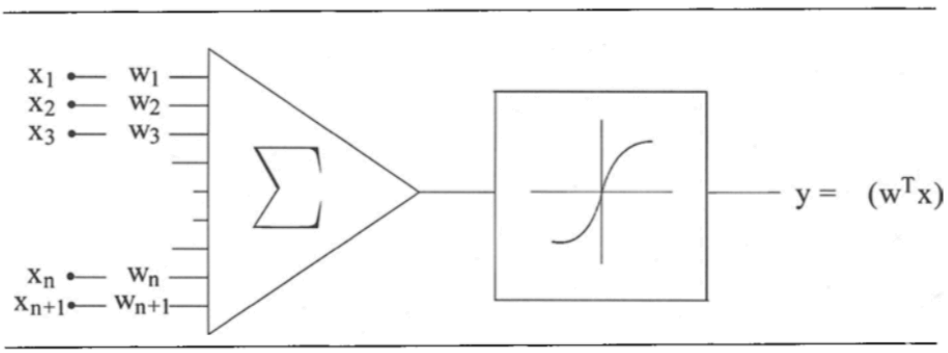

Figura 3.7 Diagrama de una neurona artificial genérica [30]

La neurona McCulloch-Pitts calcula la suma de los pesos se sus *m* entradas y después aplica una función sigmoidea (de forma s) σ(x) a la suma que produce de salida [30].

$$y(\vec{x}) = \sigma\left(\sum_{i=1}^{m} w_i x_i - b\right) = \sigma(\vec{w}^T\vec{x} - b) = \sigma\big(h(\vec{x})\big) \qquad (3.7)$$

Donde $y(x)$ es el vector de salida, $\vec{x}$ es un vector de números reales, $\vec{w}$ es un vector de los coeficientes de los pesos de cada neurona, b es la constante bias y $\sigma(x)$ es la función de activación.

La discriminante lineal es una transformación afín del espacio de entrada R^m al espacio de números reales R [30].

$$h(\vec{x}) = \sum_{i=1}^{m} w_i x_i - b = \vec{w}^T\vec{x} - b \qquad (3.8)$$

Donde $h(x)$ es el vector de neuronas ocultas, $\vec{x}$ es un vector de números reales, $\vec{w}$ es un vector de los coeficientes de los pesos de cada neurona y b es la constante bias.

Una función sigmoidea es una función $f: R \to R$ que está acotada e incrementa monotónicamente. En la práctica tanto la función sigmoidea logística, en la ecuación (3.9) [30], como la función sigmoidea de tangente hiperbólica, en la ecuación (3.10) [30], son usadas, dependiendo de si la salida tiene o no signo.

Otras funciones de activación que también se utilizan son *Rectified Linear Unit* (ReLU), en la ecuación (3.11) [30], y la *Parametric Rectified Linear Unit* (PReLU), en la ecuación (3.12) [30].

$$\sigma(h) = \frac{1}{1 + e^{-kh}} \quad (3.9)$$

$$\sigma(h) = \tanh(h) = \frac{1 - e^{-kh}}{1 + e^{-kh}} \quad (3.10)$$

$$\sigma(h) = \begin{cases} 0, h < 0 \\ h, h \geq 0 \end{cases} \quad (3.11)$$

$$\sigma(h) = \begin{cases} \alpha h, h < 0 \\ h, h \geq 0 \end{cases} \quad (3.12)$$

Donde $\sigma(x)$ es la función de activación, $h(x)$ es el vector de neuronas ocultas, k es una constante y α es el parámetro de la función PReLU.

La razón para elegir la función sigmoidea es que la derivada es fácil de computar con el valor de la función [30].

$$\sigma'(h) = k\sigma(h)\big(1 - \sigma(h)\big) \quad (3.13)$$

Donde $\sigma'(x)$ es la derivada de la función de activación, $\sigma(x)$ es la función de activación, $h(x)$ es el vector de neuronas ocultas y k es una constante.

Esto es útil cuando se entrenan las redes neuronales. Con la función de transferencia sigmoidea la salida de la neurona se convierte en una función no linear de sus entradas. El valor de salida de la neurona siempre está entre 0 y 1, o -1 y 1 en el caso de la función sigmoidea de tangente hiperbólica. La entrada de una neurona puede tener cualquier valor real, sin embargo, cuando viene de otra neurona estará limitado al rango de salida de la neurona previa.

La expresión en la ecuación (3.7) no involucra tiempo. Esta aproximación es válida aun para predecir series de tiempo. La razón es que los valores en la serie de tiempo corresponden a momentos discretos en el tiempo separados por intervalos de tiempo que son largos comparados con el tiempo que la neurona necesita para calcular el valor de salida.

Una perspectiva más realista es que existe un retraso de propagación de la señal a través de la neurona. Por lo que, una vez que la entrada está estable la salida se estabilizará en el valor descrito en la ecuación (

3.7) después de cierto tiempo de retraso Δt. Al medir el tiempo en múltiplos de Δt la salida de la neurona en el siguiente paso se describe [30]

$$y(t+1) = \sigma(\vec{w}^T \vec{x}(t) - b) \quad (3.14)$$

Donde $y(t+1)$ es la salida de la neurona, $\sigma(x)$ es la función de activación, $\vec{x}$ es un vector de números reales, $\vec{w}$ es un vector de los coeficientes de los pesos de cada neurona y b es la constante bias.

El comportamiento dependiente del tiempo de una neurona McCulloch-Pitts puede ser modelado más precisamente con una ecuación diferencial, que describe como cambia la salida continuamente en el tiempo. Sin embargo, no se requiere ese nivel de detalle para predicciones de series de tiempo.

Las redes neuronales como la que se muestra en Figura 3.6 son llamadas *feedforward* por qué el flujo de la señal siempre es una dirección, de las entradas a las salidas. Las redes neuronales *feedforward* están construidas de una o más capas de neuronas. Las salidas de la capa anterior son las entradas de la siguiente capa. Las salidas de una capa solamente alimentan la siguiente capa, nunca la misma capa o anteriores. Las capas antes de la capa de salida se llaman capas ocultas. La primera capa siempre recibe las entradas externas.

Las entradas deben ser siempre constante los suficientemente grandes para permitir que las señales se propaguen a través de la red y que las salidas se estabilicen en nuevos valores. El comportamiento del proceso transitorio no es de interés. Las redes *feedforward* representan un mapeo estático de las entradas hacía las salidas y no hay una referencia explícita del tiempo.

El tipo de red neuronal *feedforward* más usado frecuentemente es el perceptrón multicapa (*Multilayer Perceptron* MLP). La MLP consiste en al menos dos capas de neuronas McCulloch Pitts. El teorema universal de aproximación establece las capacidades matemáticas de una MLP con una capa oculta de neuronas sigmoideas y una neurona lineal en la capa de salida.

La red neuronal MLP implementa una expresión de la función escalar desconocida de p variables como una combinación lineal de un set de funciones base p-dimensional, que son las funciones sigmoideas [30]. Donde W es la matriz de coeficientes de peso.

$$y(\vec{x}) = \vec{v}^T \sigma\left(W^T \vec{x} - \vec{b}\right) \quad (3.15)$$

Donde $y(x)$ es la salida de la neurona, $\sigma(x)$ es la función de activación, $\vec{x}$ es un vector de números reales, $\vec{w}$ es un vector de los coeficientes de los pesos de cada neurona y b es la constante bias.

Por virtud del teorema una MLP nunca debe de tener más de una capa oculta para representar una función continua con el grado de precisión deseado. Además, la neurona en la capa de salida es una neurona lineal, esto significa, que la suma ponderada de la salida de las neuronas de la capa oculta no alimenta una función de transferencia sigmoidea.

La MLP debe validarse para saber su precisión. El primer paso para obtener la precisión de la salida de una red es separar algunas de las muestras de entrada-salida para probar el rendimiento de la red. Los datos del conjunto de prueba no se deben usar para el entrenamiento de la red. La situación deseable es que el error RMS de la salida de la red en el conjunto de pruebas y el error en el conjunto de entrenamiento sea

casi el mismo. Un error en el conjunto de pruebas que sea significativamente mayor que el error en el conjunto de entrenamiento indica que la red se ha ajustado en exceso a los valores particulares del entrenamiento. Una solución al problema de sobre entrenamiento es tener un tercer conjunto de los datos disponible, al que se llama conjunto de validación. Durante el entrenamiento, se monitorea el error en el conjunto de validación y el entrenamiento termina cuando el error en el conjunto de entrenamiento está por debajo del error en el conjunto de validación [30].

El algoritmo de retropropagación (*backpropagation*) es un algoritmo de aprendizaje para redes neuronales en el cuál la activación se propaga en la red y las unidades de activación en la salida se comparan con un vector de aprendizaje. Esto representa los pares de entrada/salida. La comparación de los pares de entrada/salida resulta en los errores, que se usan para propagar los cambios de regreso a través de las capas de pesos. Los pesos representan la fortaleza de las conexiones entre nodos y sus vecinos en una red neuronal, estos pueden ser positivo o negativos. Los pesos representan la "inteligencia" de la red y son la esencia de la capacidad de predicción [31]. El gradiente descendente se usa en el algoritmo de retropropagación para minimizar el error. Los pasos a continuación describen el algoritmo [32]:

- Crear una red neuronal *feedforward* con n_{in} entradas, n_{hidden} unidades ocultas y n_{out} unidades de salida.
- Se inicializan los pesos de la red con números aleatorios pequeños.
- Hasta que la condición de terminación se cumpla, se repiten los siguientes pasos:
    - Para cada $\langle \vec{x}, \vec{t} \rangle$ en el conjunto de entrenamiento, donde $\vec{x}$ es el vector de entrada de la red y $\vec{t}$ es el vector objetivo de salida de los valores de salida.
        - Propagar la entrada a través de toda la red: el valor de entrada será $\vec{x}$ para la red neuronal y computar la salida o_u para cada unidad u en la red.
        - Retropropagar los errores a través de la red. Para cada unidad de salida k, calcular el término de error δ_k [33]

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \quad (3.16)$$

        - Para cada unidad oculta h, calcular el término de error δ_h [33]

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\, \delta_k \quad (3.17)$$

        - Actualizar cada peso de la red w_{ij} [33]

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$
$$\text{Donde } \Delta w_{ji} = \eta \delta_j x_{ji} \quad (3.18)$$

Si los valores iniciales de los pesos de las neuronas de la red son muy pequeños, la mayoría de las señales tomarán valores muy pequeños. Por lo contrario, si los valores iniciales de los pesos de las neuronas de la red son muy grandes, la mayoría de las señales tomarán valores muy grandes y pasarán las funciones de activaciones de las capas posteriores. Por lo que la inicialización de Xavier ayuda a la generación óptima de los pesos iniciales de las neuronas, tal que las señales tiendan a valores muy pequeños, ni muy grandes.

La inicialización de Xavier para una distribución normal, cuando la función de activación es logística se muestra en la ecuación (3.19). Para una distribución normal, cuando la función de activación es la tangente hiperbólica, se utiliza la ecuación (3.20) [33].

$$W_l = random(n_l, n_{l-1}) * \sqrt{\frac{2}{n_l + n_{l-1}}} \quad (3.19)$$

Donde W_l son los valores iniciales de los pesos, n_l es la cantidad de neuronas de la capa y n_{l-1} es la cantidad de neuronas de la capa anterior [33].

$$W_l = random(n_l, n_{l-1}) * 4\sqrt{\frac{2}{n_l + n_{l-1}}} \quad (3.20)$$

Donde W_l son los valores iniciales de los pesos, n_l es la cantidad de neuronas de la capa y n_{l-1} es la cantidad de neuronas de la capa anterior.

Para derivar una regla de aprendizaje de pesos para unidades lineales, se comienza especificando una medida para el error de entrenamiento de una hipótesis (vector de pesos), relativos a los ejemplos de entrenamiento. Aunque hay muchas formas de definir este error, una medida común que es muy conveniente es la diferencia de los errores cuadrados que se muestra en la ecuación (3.21) [32]. Para el algoritmo del gradiente descendente se utiliza la suma de los errores cuadrados como se muestra en la ecuación (3.22) [32].

$$E = \frac{1}{2}(a - y)^2 \quad (3.21)$$

Donde *E* es el error de cada una de las muestras, *a* es la salida esperada y *y* es la salida de la red neuronal en entrenamiento.

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} E_i \quad (3.22)$$

Donde *J(w)* es el error de los pesos, *m* es la cantidad de muestras de entrenamiento y *E* es el error de cada una de las muestras.

El error de la suma de errores cuadrados que se retropropaga en el algoritmo de gradiente descendente se muestra en la ecuación (3.23) [32].

$$\frac{\partial E_i}{\partial z_k} = a - y \quad (3.23)$$

Donde E_i es el error de las muestras de entrenamiento, *a* es la salida esperada y *y* es la salida de la red neuronal en entrenamiento.

La minimización de la función de costo se utiliza para la determinación de los pesos del modelo. La consecuencia de esta acción es que los pesos se ajustan muy bien a los datos de entrenamiento y en muchos casos el modelo no funciona para casos generales. Esto hace que las redes neuronales sean susceptibles al sobre ajuste. Para resolver este problema se introduce el termino de regularización a la función de costo definida anteriormente y se genera una nueva función de costo que se muestra en la ecuación (3.24) [32].

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} E_i + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{n_{l+1}}\sum_{j=1}^{n_l} {w^2}_{lji} \quad \textbf{(3.24)}$$

$$\frac{\partial J}{\partial w_l} = \frac{\lambda}{m} w_l \quad \textbf{(3.25)}$$

Donde *J(w)* es la función de costo que depende del error de estimación y los pesos de la red neuronal, *m* es la cantidad de muestras de entrenamiento, λ es la constante de regularización, *w* son los pesos, *L* es la cantidad de capas de la red neuronal, n_l es la cantidad de neuronas de la capa y n_{l+1} es la cantidad de neuronas de la capa siguiente.

4. DESARROLLO METODOLÓGICO

La propuesta de este TOG es generar un sistema de recomendación para configuraciones de confort de un vehículo utilizando distintos modelos. Al asumir que cada uno de los modelos tiene un asertividad mayor para un caso de uso específico, el objetivo es cubrir los distintos perfiles de usuarios al combinar dichos modelos. La Figura 4.1 muestra los pasos que se siguieron para el desarrollo de este TOG.

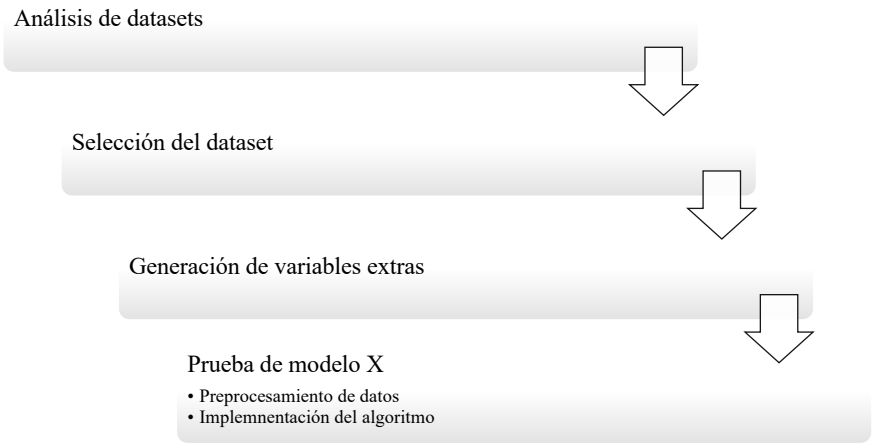

Figura 4.1 Proceso general de desarrollo del TOG

Los subcapítulos a continuación describen detalladamente cada uno de estos pasos.

4.1. Selección del *Dataset* y Generación de Datos

En este subcapítulo se cubren el análisis de los *datasets*, la selección del *dataset* y la generación de variables extras que se muestran en la Figura 4.1.

Después de analizar distintos *datasets* que se encontraron en Internet; para este trabajo se utilizó un *dataset* tomado de [34], el cual fue seleccionado debido a que cumplía con las siguientes características:

- Son 39 archivos distintos con aproximadamente 7500000 muestras distintas.
- Se realizó un análisis de calidad de los datos y se observó que las muestras son representativas para el desarrollo de este TOG.
- Los datos de viaje de cada archivo son independientes entre sí.

- Proveen información de confort como el estado del aire acondicionado, cantidad de pasajeros, nivel de la ventana y el volumen del radio.
- Contienen un campo que cuantifica la comodidad del conductor.

La Tabla 4.1 muestra los resultados del análisis de calidad del *dataset* utilizado en este TOG.

Tabla 4.1 Análisis de calidad del *dataset*

Variables	**Tipo**	**Valores faltantes**	**Valores presentes**	**Valores únicos**	**Valor mínimo**	**Valor máximo**	**Valor promedio**	**Desviación estándar**
'Time_inSeconds_'	'double'	0	7748753	189438	0.016	4306	1252.45419	896.655631
'Vehicle_sSpeed_inM_s_'	'double'	0	7748753	365115	-50.308	32.343	7.84973194	7.35632293
'ShiftNumber_0_IntermediatePosition_'	'double'	0	7748753	6	0	5	1.76037306	1.86754636
'TotalAcceleration_m_s_2_'	'double'	0	7748753	500250	-73.774	6.4855	0.15983073	0.79801346
'LateralAcceleration_m_s_2_'	'double'	0	7748753	704910	-5.9633	5.7768	-0.0138537	0.4297008
'PassengerCount_0_5_'	'double'	0	7748753	5	0	4	0.4389443	0.86445143
'AirConditioningStatus_0_4_'	'double'	0	7748753	4	0	4	0.12674362	0.41593538
'WindowOpening_0_10_'	'double'	0	7748753	7	0	6	0.75317112	1.40282375
'RadioVolume_0_10_'	'double'	0	7748753	9	0	8	3.96536023	2.01424317

'RainIntensity_0_10_'	'double'	0	7748753	4	0	3	0.32292899	0.72560861
'Driver_sWellbeing_0_10_'	'double'	0	7748753	7	2	8	5.82009634	1.45074747
'Driver_sRush_0_10_'	'double'	0	7748753	6	0	5	2.83301003	1.30011585

Aun cuando el *dataset* ya contenía algunas variables relacionadas con el confort de los ocupantes del vehículo, se decidió generar datos para 8 variables adicionales que estaban más relacionadas con el confort. Las variables que se generaron son:

1. Temperatura fuera del vehículo.
2. Temperatura dentro del vehículo.
3. Lectura del sensor de luz.
4. Estado de las luces exteriores del vehículo.
5. Estado de los limpia parabrisas.
6. Estado del freno.
7. Estado de conexión de dispositivo Bluetooth.
8. Estado del sistema de navegación (encendido o apagado).

Para cada variable fue elegida una estrategia diferente de generación de los datos debido a que cada variable tienen características físicas diferentes.

1. La estrategia para la generación de datos de la temperatura fuera del vehículo fue la siguiente: se toma el valor máximo del estado del aire acondicionado, i.e. 'AirConditioningStatus_0_4_'. Si el valor es 0, se genera una temperatura aleatoria entre 15º y 22º C. Si el valor va de 2 a 4, el rango de temperatura generado es entre 25º y 35º C.
2. La estrategia para la generación de datos de la temperatura dentro del vehículo fue la siguiente: se toma el valor generado de la temperatura fuera del vehículo, indirectamente relacionado con el estado del aire acondicionado. La temperatura dentro del vehículo se calculó con la ecuación (4.1).

$$temp_{in} = temp_{\ out} - (2 * AC_{status}) \quad (4.1)$$

   donde $temp_{in}$ es la temperatura dentro del vehículo, $temp_{\ out}$ es la temperatura fuera del vehículo y AC_{status} es el estado del aire acondicionado.

3. La estrategia para la generación de datos del sensor de luz fue la siguiente: se genera una hora aleatoria, si el valor va de 7:30 am a 7:00 pm el valor del sensor se genera entre 70% y 100%. De lo contrario, de 7:00 pm a 7:30 am el valor del sensor se genera entre 20% y 70%.

4. La estrategia para la generación de datos del estado de las luces externas del vehículo fue la siguiente: se toma el valor del sensor de luz, si el valor está entre 0% y 65%, el estado de las luces es prendido; de lo contrario, las luces están apagadas.
5. La estrategia para la generación de datos del estado de los limpia parabrisas fue la siguiente: se toma el valor de intensidad de la lluvia, que va de 0 a 10, y la escala va de 0 a 2. El valor se calculó con la ecuación (4.2).

$$wiper = {}^{rain_{intensity}}/_{3} \qquad (4.2)$$

donde $wiper$ es el estado del limpia parabrisas y $rain_{intensity}$ es el valor de la intensidad de la lluvia.

6. La estrategia para la generación de datos del estado de frenado fue la siguiente: se toma la aceleración total y cuando el valor es negativo, el valor de frenado es el valor absoluto de la aceleración. En caso de que el valor de aceleración sea positivo, el valor de frenado es 0.
7. La estrategia para la generación de datos del estado del dispositivo Bluetooth (conectado o desconectado) fue la siguiente: se elige aleatoriamente un tiempo y aleatoriamente se determina si el estado antes de ese tiempo es conectado o desconectado. El estado después de ese tiempo es el contrario.
8. La estrategia para la generación de datos del estado del sistema de navegación (encendido o apagado) fue la siguiente: se elige aleatoriamente un tiempo y aleatoriamente se determina si el estado antes de ese tiempo es encendido o apagado. El estado después de ese tiempo es el contrario.

La Tabla 4.2 muestra el análisis de calidad de los datos aplicada a las variables que se generaron para este TOG.

Tabla 4.2 Análisis de calidad de las variables generadas para el dataset

Variables	Tipo	Valores faltantes	Valores presentes	Valores únicos	Valor mínimo	Valor máximo	Valor promedio	Desviación estándar
'tempOutsideVehicle'	'double'	0	7748753	7748753	15.0071695	34.9763115	20.7413859	4.01321157
'tempInsideVehicle'	'double'	0	7748753	7748752	14.808811	35.1727698	20.4879324	3.59857322
'lightSensor'	'double'	0	7748753	7748753	24.546716	104.184677	69.342791	24.7638967
'extLights'	'double'	0	7748753	2	0	1	0.3999664 2	0.48989112
'wiperState'	'double'	0	7748753	2	0	1	0.0491854 6	0.21625508
'brakeState'	'double'	0	7748753	250599	0	73.774	0.1937658 1	0.46928566
'BTdeviceConnected'	'double'	0	7748753	2	0	1	0.3994575 6	0.48978694
'NavigationSysRunning'	'double'	0	7748753	2	0	1	0.4515073 6	0.49764294

4.2. Sistema de Recomendación

Para el sistema de recomendación basado en similitud de usuarios se implementó el modelo que se presenta en [20]. El propósito del modelo es calcular recomendaciones acertadas para cualquier usuario en un ambiente de producción que involucra una cantidad masiva de datos.

4.2.1. Preprocesamiento de los Datos

Un sistema de recomendación se obtiene del análisis de un conjunto de datos como base de conocimiento. En este trabajo se crearon tres subconjuntos de datos diferentes, de los cuales se generó un sistema de recomendación diferente para poder evaluar el desempeño de cada uno. El sistema de recomendación se probó con estos subconjuntos del *dataset* generado. Los valores de los subconjuntos se utilizaron sin ningún procesamiento previo, es decir, ninguna transformación de los datos.

1. Para los experimentos que se usaron todas las muestras, solamente se hizo una compilación de las muestras que estaban en distintos archivos; obteniendo solo las columnas a utilizar y guardando un archivo .mat.

2. Para los experimentos que se usaron solo 10% de las muestras, se tomaron todas las muestras del archivo .mat y se tomaron 10% de las muestras aleatoriamente. Es decir, se generaron números aleatorios entre 1 y el número total de muestras, y las muestras en dichas filas se compilaron para generar el subconjunto de datos en otro archivo .mat.
3. Para los experimentos que se usaron los centroides resultantes, se tomaron todas las muestras del archivo .mat y se corrió el algoritmo de *clustering k-means* con todas las muestras, k = [80, 130] y número máximo de iteraciones de 1000. Una vez que se tienen los clústeres, se evalúa cuál es la cantidad óptima de clústeres utilizando el criterio de Calinski-Harabasz y se toman los centroides como el subconjunto de datos, guardándolos en un archivo .mat.

4.2.2. Implementación del Algoritmo

Para este trabajo, el modelo considera cada muestra como un conjunto de variables. De este conjunto, *n* variables son conocidas y *m* son desconocidas. Por lo que, el usuario más cercano se define como las muestras con las *n* variables similares y las *m* variables se calculan independientemente con el procedimiento descrito a continuación y en la Figura 4.2.

1. Calcular el promedio de las muestras de entrada usando la ecuación (3.1) como se muestra en Figura 4.3.
2. Calcular la similitud entre las muestras de entrada y la muestra de prueba usando la ecuación (3.2) como se muestra en Figura 4.4.
3. Para descartar las diferentes escalas utilizadas en las muestras de entrada, se calcula las similitudes centradas en el promedio usando la ecuación (3.3) como se muestra en Figura 4.5.
4. Por último, se usan los resultados parciales de los pasos anteriores para calcular la recomendación para cada *m* variable usando la ecuación (3.4) como se muestra en Figura 4.6.

Para el cálculo de la recomendación descrito en el paso 4, se utilizaron tres métodos distintos para elegir la población representativa y definir las muestras similares. Se buscó obtener el mejor método como una combinación optimizada de costo computacional y asertividad para el modelo.

Las variables de prueba que se utilizaron para los experimentos fueron:

- Estado del aire acondicionado (columna 7)
- Apertura de la ventana (columna 8)
- Estado del sensor de luz (columna 15)

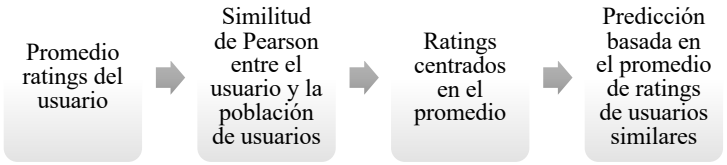

Figura 4.2 Modelo basado en similitud de usuarios

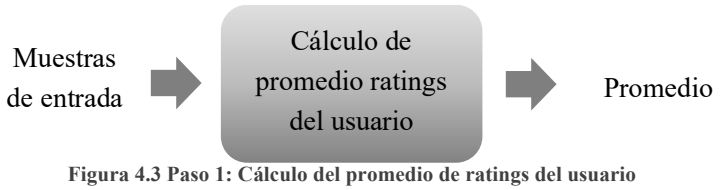

Figura 4.3 Paso 1: Cálculo del promedio de ratings del usuario

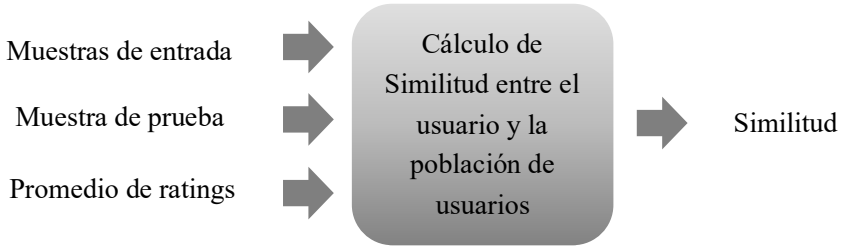

Figura 4.4 Paso 2: Cálculo de similitud entre el usuario y la población

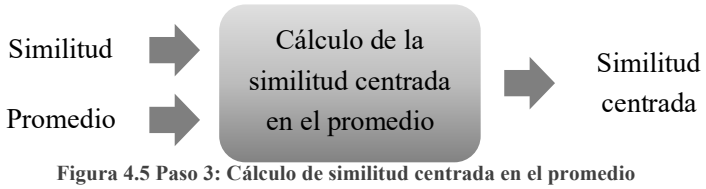

Figura 4.5 Paso 3: Cálculo de similitud centrada en el promedio

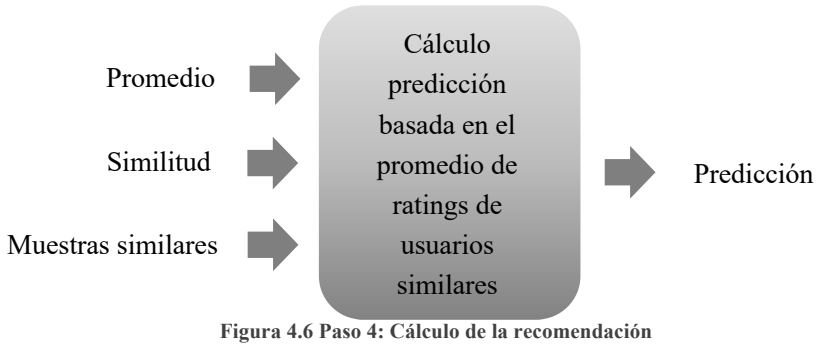

Figura 4.6 Paso 4: Cálculo de la recomendación

4.2.2.1. Métodos para Elegir la Población

El objetivo de un modelo basado en similitud de usuarios es calcular recomendaciones acertadas para cualquier usuario en un ambiente de producción que involucra una cantidad masiva de datos. Por ello es importante elegir la cantidad correcta de muestras que representan la población para computar recomendaciones acertadas en un periodo de tiempo aceptable y costo de computación aceptable.

Para los experimentos, se usaron tres métodos para seleccionar la población representativa: tomar todas las muestras, tomar los centroides resultantes de correr el algoritmo de *clustering* para todas las muestras y tomar 10% de muestras aleatorias de toda la población.

Para el método en el que se usaron todas las muestras, aplicadas como entradas en las ecuaciones (3.1), (3.2) y (3.3), no se necesitó ninguna etapa de pre procesamiento, pero el costo computacional es

proporcional a la cantidad de datos. Se debe considerar que en producción se generarían datos se continuamente, por lo que la cantidad de muestras a procesar puede volverse inmanejable.

Para el método en el que se usaron los centroides resultantes, se corrió el algoritmo de *K-means* donde todas las muestras se usaron como entrada usando un iterador de 1000. El número óptimo de clústeres es desconocido; por lo que el algoritmo de *K-means* se corrió seleccionando de 80 a 130 clústeres. Después de este paso, una evaluación de los resultados se ejecuta para determinar el número óptimo de clúster y se toman los centroides de esa corrida. Los centroides son usados en las ecuaciones (3.1), (3.2) y (3.3) como las muestras similares con base en las cuales se obtendrán las recomendaciones.

La etapa de pre procesamiento para este método consume mucho tiempo y demanda muchos recursos computacionales, lo cual representa una gran desventaja comparado con los otros dos métodos usados en los experimentos. En contraste, una vez que la población se ha seleccionado, la calidad de los datos es alto y la cantidad baja; lo que significa que los cálculos se computan más rápidos con la ecuación (3.2).

Para el método en el que se usaron 10% de todas las muestras, estas se seleccionan aleatoriamente para evitar un sesgo. La etapa de pre procesamiento para este método es simplemente tomar el 10% de muestras, i.e. el costo computacional puede ser discriminado. Al comparar este método con el que utiliza todas las muestras, este método es 90% más eficiente desde el punto de vista de tiempo de cómputo, sin embargo, sigue siendo proporcional al tamaño de la población.

4.2.2.2. Métodos para Elegir las Muestras Similares

Como se mencionó anteriormente, la selección de la población puede alterar el desempeño de los sistemas de recomendación. Una vez definiendo la población de análisis, existe un factor también determinante en la calidad de las predicciones; este factor es el número de muestras similares que se tomaran en cuenta para hacer las predicciones. Para optimizar los resultados del modelo basado en similitud de usuarios, se deben elegir el número correcto de muestras similares para usar en la ecuación (3.4). En los experimentos, se utilizaron tres métodos distintos e independientes para después determinar cuál es el mejor para el modelo final.

El primer método que se utilizo es elegir un número constante de muestras similares. Después de que se calcula la similitud entre la muestra de prueba y la población, los resultados se ordenan y un número constante de muestras con la correlación más alta son tomadas en la ecuación (3.4). En los experimentos el número de muestras se incrementó logarítmicamente, i.e. 1, 10, 100 hasta que el tamaño de la población lo permita; debido a que se quería mostrar si al aumentar el número de muestras similares tomadas en cuenta para hacer la predicción se tendrían mejoras en la exactitud de las predicciones.

Para el segundo método, después de que la similitud se calcula entre la muestra de prueba y la población, los resultados se ordenan y las muestras con la correlación más alta son tomadas. La recomendación es calculada utilizando un porcentaje de las muestras de la población. El porcentaje de las muestras va de 1% a 12%.

Para el tercer método, después de que la similitud se calcula entre la muestra de prueba y la población, los resultados se ordenan y las muestras con la correlación más alta son tomadas. La recomendación es calculada utilizando las muestras similares que se encuentren dentro de un rango definido. El rango va del 50% al 100% con incrementos de 5% dentro de ese rango.

4.3. PageRank

Para el *PageRank* se implementó el modelo que se presenta en [23]. El propósito del modelo es estimar el siguiente estado de las configuraciones del vehículo con base en el estado actual y el historial de preferencias de dicho conductor.

4.3.1. Preprocesamiento de los Datos

El algoritmo *PageRank* se basa en los estados que toman las variables en el tiempo, por lo que, si las variables que se analizan tienen un número grande de estados, esto aumenta la complejidad computacional en el cálculo de las estimaciones. El algoritmo de *PageRank* recibe en la entrada datos discretos. Ya que el *dataset* es de datos continuos, se requirió de una etapa de preprocesamiento en la que el *dataset* se convirtió a valores discretos. El proceso se muestra en la Figura 4.7.

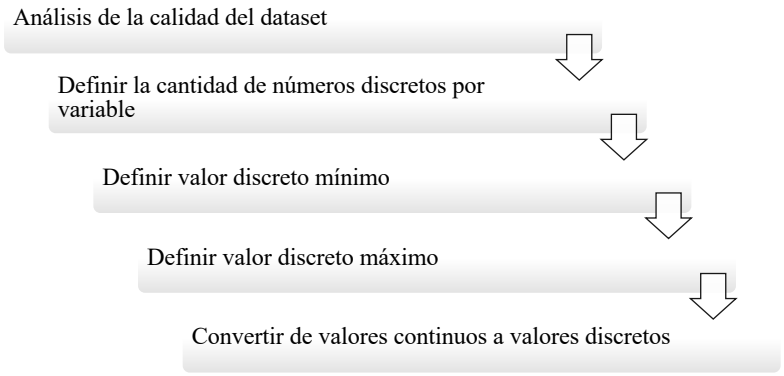

Figura 4.7 Proceso para convertir el dataset de valores continuos a discretos

La Tabla 4.1 contiene el análisis de la calidad de los datos. A partir de dicho análisis, se definió la cantidad de números discretos que se querían obtener por variable. No se utilizó ningún criterio en particular, solo se tomó en cuenta el origen de los datos, la resolución que se quería tener para representar significativamente cada variable y que conforme incrementa la cantidad de números discretos, se requerirían más recursos para ejecutar el algoritmo. Posteriormente, con base en los valores mínimos y máximos y la cantidad de números discretos por variable, se definió el valor discreto máximo y mínimo. Una vez que se establecieron todos los parámetros, i.e. cantidad de números discretos, valor mínimo y valor máximo en la Tabla 4.3, se convirtieron todos los valores de continuos a discretos con el siguiente código de Matlab.

```
for i = 1:20
    if isnan(tab_dqr.num_discrete_vals(i+1)) == false
        % Discretizar variables
        range = tab_dqr.max_to_discrete{i+1} -
tab_dqr.min_to_discrete{i+1};
        resolution = range / (tab_dqr.num_discrete_vals(i+1) - 1);
        m(:,i) = (m(:,i) - tab_dqr.min_to_discrete{i+1}) ./ resolution;
    end
end
```

Una vez calculados los valores discretos del *dataset*, se guardó en un archivo .mat para ser utilizado en la implementación del *PageRank*.

Tabla 4.3 Parámetros para convertir a valores discretos

Variables	# Valores discretos	Valor discreto mínimo	Valor discreto máximo
'Vehicle_sSpeed_inM_s_'	7	-50	35
'ShiftNumber_0_Intermediate Position_'	6	0	5
'TotalAcceleration_m_s_2_'	5	-80	10
'LateralAcceleration_m_s_2_'	7	-6	6
'PassengerCount_0_5_'	5	0	4
'AirConditioningStatus_0_4_'	5	0	4
'WindowOpening_0_10_'	3	0	6
'RadioVolume_0_10_'	3	0	8
'RainIntensity_0_10_'	4	0	3
'Driver_sWellbeing_0_10_'	4	0	8
'Driver_sRush_0_10_'	3	0	5
'tempOutsideVehicle'	5	15	35
'tempInsideVehicle'	3	10	35
'lightSensor'	4	0	100
'extLights'	2	0	1
'wiperState'	2	0	1
'brakeState'	4	0	80
'BTdeviceConnected'	2	0	1
'NavigationSysRunning'	2	0	1

4.3.2. Implementación del Algoritmo

Los pasos para la implementación del algoritmo de *PageRank*, como se muestran en la

Figura 4.8, fueron:

1. Generar la matriz de transición para múltiples variables como se muestra en la Figura 4.9.
2. Predecir las configuraciones del vehículo:
    a. Modelo observador de estados en el que se toman en cuenta todas las variables de entrada y a la salida solo cambian las variables de prueba como se muestra en la Figura 4.10.

b. Modelo estimador de ambiente en el que se toman en cuenta todas las variables de entrada y a la salida pueden cambiar todas las variables, es decir, tanto las variables de prueba como las variables que no se están probando como se muestra en la Figura 4.11.

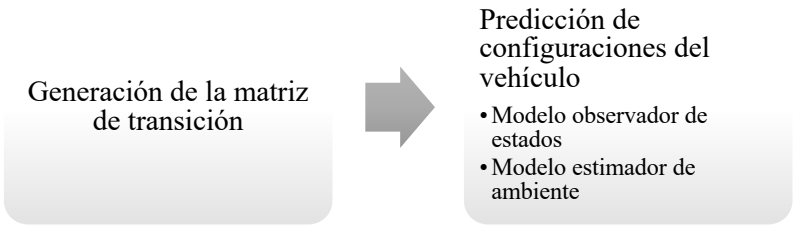

Figura 4.8 Proceso para implementación de *PageRank*

La generación de la matriz de transición requirió el mayor tiempo de implementación y procesamiento en la implementación del *PageRank*. Primero se eligieron las variables de prueba i.e. variables dependientes: estado del aire acondicionado (columna 7), apertura de la ventana (columna 8) y estado del sensor de luz (columna 15).

A continuación, se determina la cantidad de combinaciones posibles de todas las variables que generan la matriz, es decir, variables dependientes e independientes. Para esto se obtiene la cantidad de posibles estados para cada una de las variables. Utilizando la función combvec() de Matlab, se obtuvo iterativamente todas las combinaciones posibles de las variables. Se obtienen las combinaciones comenzando con dos variables, posteriormente la matriz resultante se combinó con los posibles valores de la siguiente variable y así sucesivamente hasta obtener todas las combinaciones posibles. Al final se transpuso la matriz, para obtener *n* filas dónde cada fila una combinación única y cada columna corresponde a una variable.

La matriz de transición resultante será de tamaño *n* x *n*, donde *n* es la multiplicación de la cantidad de valores discretos de cada variable. Esto quiere decir que conforme incrementa la cantidad de variables o de valores discretos de cada variable, el tamaño de la matriz incrementa por un factor; lo cual requiere mayor cantidad de memoria para almacenarse. Para las variables seleccionadas, la cantidad de combinaciones posibles n es bastante grande y muchas combinaciones nunca ocurren en el dataset, asignándoles el algoritmo una probabilidad de 0. Debido a lo anterior, en este trabajo se utilizaron matrices dispersas de Matlab, las cuales permiten ahorrar memoria al trabajar con matrices muy grandes con muchos elementos de la matriz iguales a 0.

Para obtener las probabilidades de la matriz de transición para pasar a un estado siguiente (fila) estando en el estado actual (columna), se iteró por cada uno de los valores únicos de los estados actuales sobre los valores únicos de los estados siguientes correspondientes a dicho estado actual. Se sumaron las ocurrencias del estado siguiente para dicho estado actual y al final se divide entre la cantidad total de muestras, es decir, se calculó la probabilidad empírica.

Al final se guardó la matriz de transición, P(Y|X), en el archivo Mt_MultiVariable.mat y las combinaciones de estados, para poder indexar la matriz de transición, en el archivo Mt_MultiVariableCombinations.mat.

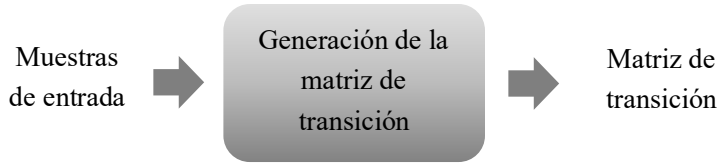

Figura 4.9 Paso 1: Generar la matriz de transición para múltiples variables

Los pasos que se siguieron para la implementación del modelo observador de estados fueron los siguientes:

1. Cargar la matriz de transición, P(Y|X), generada previamente del archivo Mt_MultiVariable.mat.
2. Cargar las combinaciones de estados para indexar la matriz de transición del archivo Mt_MultiVariableCombinations.mat.
3. Se encuentran todas las combinaciones en las que las variables independientes son iguales al estado de dichas variables en el vector de prueba.
4. Se asigna la probabilidad 1 / x, dónde x es la cantidad de combinaciones encontradas en el paso anterior, a todas las filas en las de las combinaciones encontradas en el paso anterior para general el vector P(X).
5. Calcular la probabilidad de ocurrencia de cada uno de los vectores con la ecuación (4.3).
6. Del vector resultante, P(Y), se obtienen la máxima probabilidad dónde las variables independientes son iguales en el estado actual y estado siguiente.
7. Los valores siguientes de las variables de prueba, es decir, los valores predichos serán los valores de dichas variables dependientes en el vector correspondiente a la máxima probabilidad encontrado en el paso anterior.
8. Se tomaron 10000 muestras aleatorias para predecir el estado siguiente con el modelo observador de estados y se comparan los resultados con los valores esperados de los vectores de prueba.

$$P(Y) = P(Y|X)P(X) \qquad (4.3)$$

Donde P(X) es el vector de probabilidad de que ocurra cada combinación de estados actuales, P(Y|X) es la matriz de transición y P(Y) es la probabilidad de que ocurra cada combinación en el estado siguiente.

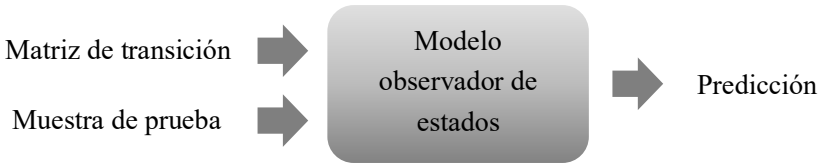

Figura 4.10 Paso 2.a: Predicción con modelo observador de estados

Los pasos que se siguieron para la implementación del modelo estimador de ambiente fueron los siguientes:

1. Cargar la matriz de transición, P(Y|X), generada previamente del archivo Mt_MultiVariable.mat.

2. Cargar las combinaciones de estados para indexar la matriz de transición del archivo Mt_MultiVariableCombinations.mat.
3. Se encuentran todas las combinaciones en las que las variables independientes son iguales al estado de dichas variables en el vector de prueba.
4. Se asigna la probabilidad 1 / x, dónde x es la cantidad de combinaciones encontradas en el paso anterior, a todas las filas en las de las combinaciones encontradas en el paso anterior para general el vector P(X).
5. Calcular la probabilidad de ocurrencia de cada uno de los vectores con la ecuación (4.3).
6. Del vector resultante, P(Y), se obtienen la muestra con la máxima probabilidad.
7. El estado siguiente de todas las variables, dependientes e independientes, será el vector correspondiente a la máxima probabilidad encontrado en el paso anterior. Esto significa que todo el ambiente va a ser predicho por el modelo.
8. Se tomaron 10000 muestras aleatorias para predecir el estado siguiente con el modelo observador de estados y se comparan los resultados con los valores esperados de los vectores de prueba.

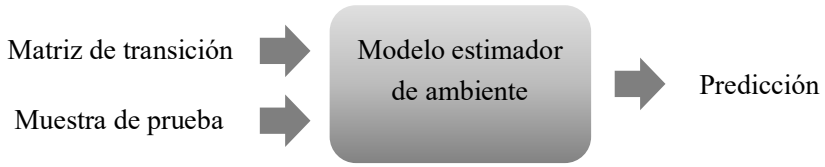

Figura 4.11 Paso 2.b: Predicción con modelo estimador de ambiente

4.4. Predicción de Series de Tiempo con Redes Neuronales

Para la predicción de series de tiempo con redes neuronales se implementó el modelo que se presenta en [30]. El propósito del modelo es calcular cuál será el siguiente estado de las configuraciones del vehículo con base en el estado actual y el historial de preferencias de dicho conductor.

4.4.1. Preprocesamiento de los Datos

La red neuronal se entrenó y se probó con el *dataset* generado. Previo a la implementación del modelo se realizó el siguiente procesamiento de los datos:

1. Se normalizaron los datos para que todos los valores estén en un rango entre 0 y 1, para variables solo con valores positivos, y un rango entre -1 y 1, para variables con valores positivos y negativos.
2. Se separaron las entradas y las salidas para el modelo. Las salidas, que son los valores esperados, corresponden al estado del aire acondicionado (columna 7), la apertura de la ventana (columna 8) y el estado del sensor de luz (columna 15). Las entradas, que son las variables independientes, son el resto de las columnas del *dataset*.
3. El *dataset* se dividió aleatoriamente en dos subconjuntos: entrenamiento y prueba. El subconjunto de entrenamiento es el 70% de los datos y el subconjunto de pruebas es el 30% de los datos como lo recomienda Site et al. en [30].

4.4.2. Implementación del Algoritmo

Los pasos para la implementación del modelo de predicción de series de tiempo con redes neuronales, como se muestran en la

Figura 4.12, fueron:

1. Definición de los parámetros del experimento: la función de activación, la cantidad de neuronas en la capa oculta, la constante alfa, la constante lambda para la regularización y el número de iteraciones para entrenar el modelo.
2. Entrenamiento del modelo con el subconjunto de entrenamiento.
3. Evaluación del modelo resultante con el subconjunto de prueba.
4. Se repiten los pasos anteriores hasta encontrar un modelo óptimo.

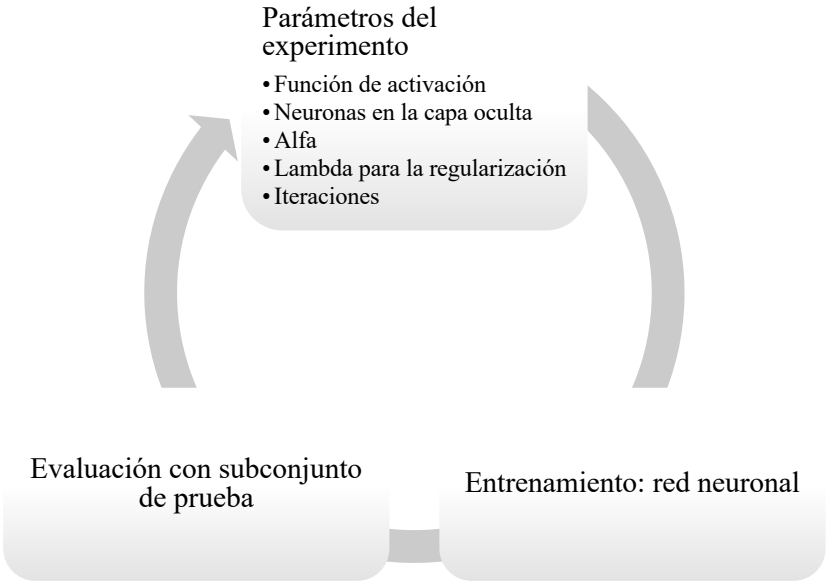

Figura 4.12 Proceso de implementación del modelo de predicción de series de tiempo con redes neuronales

Los pasos para la implementación de la red neuronal, como se muestran en la Figura 4.13, fueron:

1. Se implementó la función de inicialización Xavier.
2. Se implementaron las funciones de activación sigmoidea logística, sigmoidea tangente hiperbólica, ReLU y PReLU. Se elige una de estas funciones como parámetro del experimento.
3. Se implementó la función de regularización. Se manda el valor de lambda como parámetro para cada experimento.
4. Se implementó el algoritmo gradiente descendente al cuál se le pasan como parámetros para el experimento: la cantidad de neuronas en la capa oculta, el valor de alfa, el valor de lambda, el número de iteraciones en el entrenamiento y la función de activación que se va a usar.

5. Se regresa el modelo resultante una vez que se ha entrenado la red neuronal.

Figura 4.13 Proceso para implementación de algoritmo de predicción de series de tiempo con redes neuronales

Para la inicialización de los valores de los pesos de las neuronas de la red neuronal utilizada para generar el modelo de predicción, se implementó la función de Xavier mostrada en la ecuación (3.19) para cuando se utiliza cuando la función de activación es la logística, y la función de Xavier mostrada en la ecuación (3.20) se utiliza cuando la función de activación es la tangente hiperbólica.

Se implementaron cuatro funciones de activación distintas para elegir como parámetro durante el entrenamiento de la red neuronal: función sigmoidea logística, función sigmoidea tangente hiperbólica, *Rectified Linear Unit* (ReLU) y *Parametric Rectified Linear Unit* (PReLU). La función sigmoidea logística se implementó con la ecuación (3.9). La función sigmoidea tangente hiperbólica se implementó con la ecuación (3.10). La función ReLU se implementó con la ecuación (3.11). La función PReLU se implementó con la ecuación (3.12).

Para evitar que la red neuronal se sobreajuste a los datos de entrenamiento, se puede agregar un termino de regularización durante el entrenamiento. En este caso se implementó la función de regularización con la ecuación (3.25) en la que el parámetro λ se puede variar como parte del experimento. Para el entrenamiento de la red neuronal, se implementó el algoritmo del gradiente descendente descrito en el capítulo 3.1.3.

La evaluación del modelo resultante en cada experimento se realiza con la función de costo, como se muestra en la

Figura 4.14. Ya que la red neuronal se entrenó con el subconjunto de datos de entrenamiento, la evaluación se realiza con el subconjunto de prueba. Primero se predicen los valores de las salidas con el modelo resultante del entrenamiento. Después se calcula el error para cada uno de los valores de prueba con la ecuación (3.21). Por último, se evalúa la función costo del modelo, que es un promedio de los errores calculados previamente, con la ecuación (3.22). El funcional de costo cuantifica el rendimiento del modelo; este valor se usó para determinar cuál es el mejor modelo.

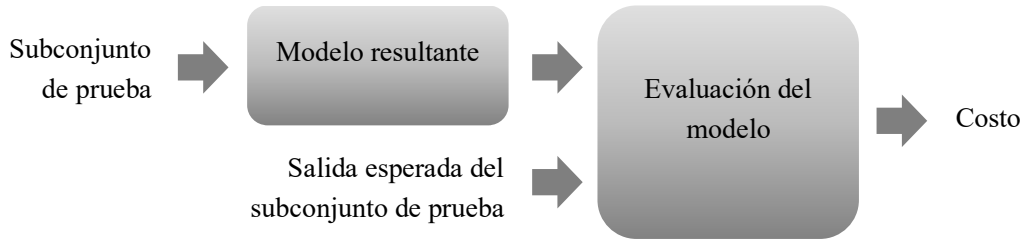

Figura 4.14 Evaluación del modelo resultante con el subconjunto de prueba

Ya que se implementaron todas las funciones y el algoritmo necesarios para generar modelos, se corrieron varios experimentos con diferentes parámetros para encontrar el mejor modelo. En la Tabla 4.4 se muestran los parámetros de los experimentos que se corrieron para este trabajo.

Tabla 4.4 Parámetros de los experimentos para el entrenamiento del modelo

#	Algoritmo	Alfa	Lambda	Epochs	Neuronas ocultas	Activación
1	SGD batch 100	0.05	0	1000	10	tanh
2	SGD batch 100	0.05	0	10000	10	tanh
3	SGD batch 100	0.01	1E-06	1000	15	tanh
4	SGD batch 100	0.01	1E-06	1000	10	logística
5	SGD batch 100	0.05	1E-06	1000	15	logística
6	SGD batch 100 - cross validation k=7	0.05	1E-06	500	10	tanh
7	SGD batch 100	0.05	0	1000	10	ReLU
8	SGD batch 100 - alpha prelu 0.2	0.05	0	1000	10	PReLU
9	SGD batch 100	0.05	0.01	2000	10	tanh
10	SGD batch 100	0.05	0.05	2000	10	tanh
11	SGD batch 100	0.1	0.05	2000	10	tanh
12	SGD batch 100	0.01	0.05	2000	10	tanh
13	SGD batch 100	0.05	0.05	9600	10	tanh

5. RESULTADOS Y DISCUSIÓN

En este capítulo se analizan los resultados de los modelos desarrollados para este TOG y una discusión sobre los resultados obtenidos al correr cada uno de los experimentos. Los resultados de este TOG marcarían una directriz para desarrollar tecnología para mejorar la experiencia de usuario dentro de un vehículo.

5.1. Resultados

En los siguientes subcapítulos se presentan los resultados de la implementación de los modelos: sistema de recomendación, PageRank y predicción de series de tiempo con redes neuronales. Se analizaron los resultados de los experimentos realizados con cada modelo propuesto en este trabajo para cuantificar su desempeño.

5.1.1. Sistema de Recomendación

Para probar el sistema de recomendación se diseñaron experimentos en los que se eligen distintos métodos para elegir la población y métodos para elegir las muestras similares. Los resultados de dichos experimentos se presentan a continuación. El conjunto de datos de prueba utilizado tiene 1000 muestras distintas que incluyen el valor esperado de las variables a prueba.

El porcentaje de error promedio absoluto (*Mean Absolute Percentage Error* MAPE) en la ecuación (5.1) es la medida del error utilizada para evaluar el rendimiento del modelo basado en similitud de usuarios para los experimentos.

$$E = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{r_{uj} - \hat{r}_{uj}}{r_{uj}} \right| \qquad (5.1)$$

Donde E es el porcentaje de error promedio absoluto, n es el número de muestras, r_{uj} es la preferencia del usuario u y $\hat{r}_{uj}$ es la predicción calculada para el usuario u.

Se corrieron tres experimentos con el sistema de recomendación, en los cuáles se utilizaron tres métodos distintos para elegir la población, como lo describe la sección 4.2.2.1. Los métodos utilizados fueron:

1. La población está compuesta por todas las muestras del *dataset*.

2. La población son los centroides de los clústeres obtenidos de correr *K-means* con el *dataset*.
3. La población son el 10% de muestras aleatorias del *dataset*.

A su vez, cada experimento probó tres métodos para elegir la cantidad de muestras similares para calcular la recomendación con la ecuación (3.4). Por lo que los nueve experimentos corridos fueron:

1. La población está compuesta por todas las muestras del *dataset*.
    a. Las muestras similares son una cantidad constante.
    b. Las muestras similares son un porcentaje de muestras de la población.
    c. Las muestras similares son todas aquellas que están dentro de un rango de similitud.
2. La población son los centroides de los clústeres obtenidos de correr *K-means* con el *dataset*.
    a. Las muestras similares son una cantidad constante.
    b. Las muestras similares son un porcentaje de muestras de la población.
    c. Las muestras similares son todas aquellas que están dentro de un rango de similitud.
3. La población son el 10% de muestras aleatorias del *dataset*.
    a. Las muestras similares son una cantidad constante.
    b. Las muestras similares son un porcentaje de muestras de la población.
    c. Las muestras similares son todas aquellas que están dentro de un rango de similitud.

Primero se corrieron los experimentos utilizando como población todas las muestras del *dataset*, ya que no se tenía un valor de error esperado, esto ayudo a obtener los valores de referencia. La Figura 5.1 muestra los resultados de estos experimentos, se calculó el MAPE de la predicción respecto al valor esperado para las 1000 variables de prueba.

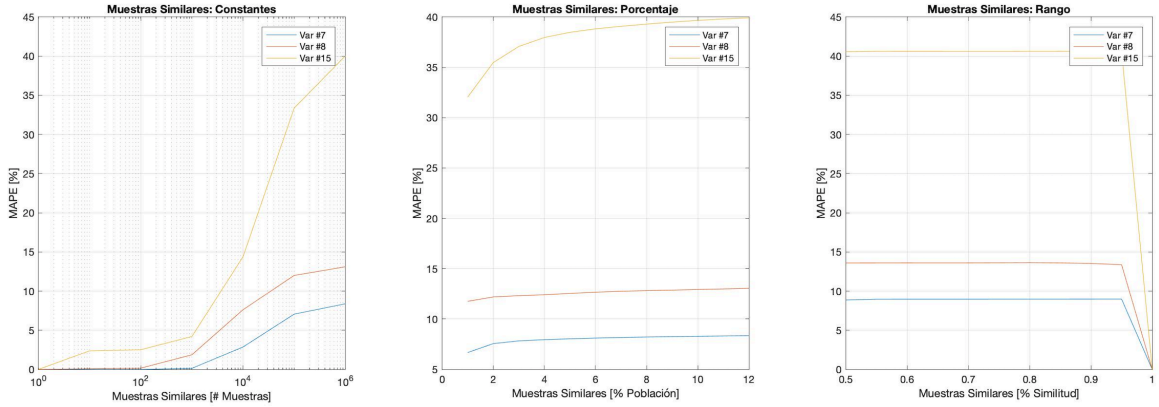

Figura 5.1 Error en experimentos que utilizaron la población completa

En la primera gráfica, “Muestras Similares: Constantes”, de la Figura 5.1 se puede observar que el error es muy bajo para las tres variables cuando se eligen menos de 1000 muestras similares, pero conforme la cantidad aumenta, el error también aumenta. Además, se puede observar que para las variables 7 y 8 los resultados son muy parecidos y el error se encuentra dentro de un rango aceptable, sin embargo, para la variable 15 el error llega al 40%.

En la segunda gráfica, “Muestras Similares: Porcentaje”, de la Figura 5.1 se puede observar que el error varía muy poco independientemente del porcentaje de muestras similares que se usan en el experimento para calcular la recomendación. Al igual que en el caso anterior, los resultados para las variables 7 y 8 son

muy parecidos y el error se encuentra dentro de un rango aceptable, sin embargo, la predicción de la variable 15 es más inexacta.

En la tercera gráfica, “Muestras Similares: Rango”, de la Figura 5.1 se puede observar que el error es prácticamente el mismo para las tres variables si se toman muestras con similitud desde 50% hasta el 95%. El error es 0 si se toman las muestras con similitud del 100%, sin embargo, no sería el método más conveniente, ya que el modelo no podría calcular una recomendación si el vehículo alcanza un estado nuevo, es decir, un estado en el que el usuario no haya provisto una preferencia anteriormente. Al igual que en los dos casos anteriores, los mejores resultados se obtuvieron para calcular la recomendación de las variables 7 y 8.

En general, los resultados de las tres variantes del experimento muestran que el modelo puede calcular mejores recomendaciones para las variables 7 y 8, es decir, al comparar el valor calculado contra el valor esperado, el error es menor y se encuentra dentro de un margen de error aceptable. De las tres variantes de este experimento, el mejor resultado se obtuvo cuando se toman menos de 1000 muestras similares para calcular la recomendación.

Como segundo paso, se corrieron los experimentos utilizando como población los centroides de los clústeres obtenidos de correr *K-means* con el *dataset*. Dada la cantidad de muestras en el *dataset* y que constantemente se siguen generando datos en el vehículo, al correr un método de aprendizaje no supervisado se obtienen los grupos representativos de la población y la cantidad de muestras disminuyó notablemente para el cálculo de la predicción. Esto es importante para que el costo computacional de dicho cálculo sea óptimo, sin embargo, implica una etapa de procesamiento fuera de línea que es muy pesada i.e. correr *K-means* con todos los datos del *dataset*. La Figura 5.2 muestra los resultados de estos experimentos, se calculó el MAPE de la predicción respecto al valor esperado para las 1000 variables de prueba.

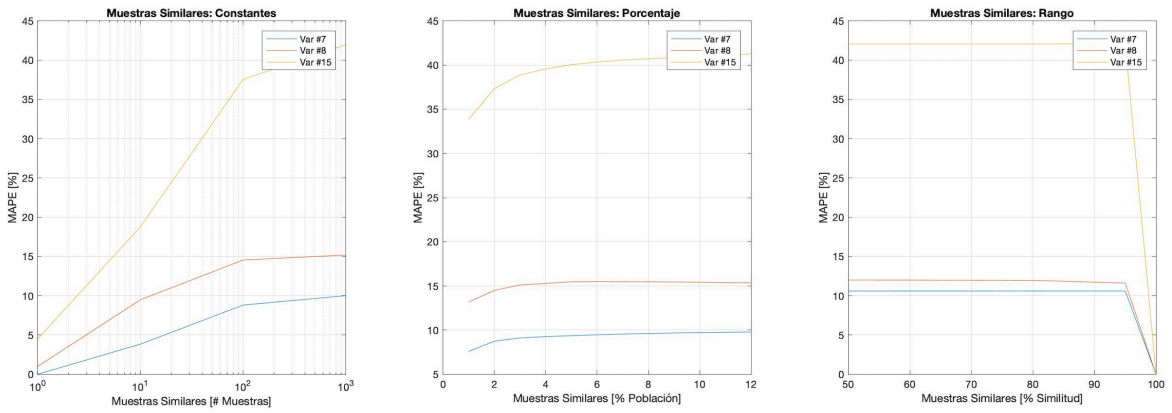

Figura 5.2 Error en experimentos que utilizaron los centroides de los clústeres como población

En la primera gráfica, “Muestras Similares: Constantes”, de la Figura 5.2 se puede observar que el error es bajo para las tres variables cuando se eligen menos de 10 muestras similares, pero conforme la cantidad aumenta, el error también aumenta. Además, se puede observar que para las variables 7 y 8 los resultados son muy parecidos y el error se encuentra dentro de un rango aceptable, sin embargo, para la variable 15 el error llega al 40%. La tendencia de comportamiento es similar a la gráfica análoga en la Figura 5.1, solo

la cantidad de muestras similares es distinta, lo cual tiene sentido ya que al aplicar el algoritmo *K-means* se puede decir que ya estaban agrupadas las muestras en los centroides resultantes.

En la segunda gráfica, “Muestras Similares: Porcentaje”, de la Figura 5.2 se puede observar que el error varía muy poco independientemente del porcentaje de muestras similares que se usan en el experimento para calcular la recomendación. Al igual que en el caso anterior, los resultados para las variables 7 y 8 son muy parecidos y el error se encuentra dentro de un rango aceptable, sin embargo, el error es mayor para la predicción de la variable 15.

En la tercera gráfica, “Muestras Similares: Rango”, de la Figura 5.2 se puede observar que el error es prácticamente el mismo para las tres variables si se toman muestras con similitud desde 50% hasta el 95%. El error es 0 si se toman las muestras con similitud del 100%, sin embargo, no sería el método más conveniente, ya que el modelo no podría calcular una recomendación si el vehículo alcanza un estado nuevo, es decir, un estado en el que el usuario no haya provisto una preferencia anteriormente. Al igual que en los dos casos anteriores, los mejores resultados se obtuvieron para calcular la recomendación de las variables 7 y 8.

En general, los resultados de las tres variantes del experimento muestran que el modelo puede calcular mejores recomendaciones para las variables 7 y 8, es decir, al comparar el valor calculado contra el valor esperado, el error es menor y se encuentra dentro de un margen de error aceptable. De las tres variantes de este experimento, el mejor resultado se obtuvo cuando se toma 2% de la población como muestras similares para calcular la recomendación. Una gran desventaja de este método es que correr la etapa fuera de línea i.e. el algoritmo *K-means* es muy pesado computacionalmente por lo que requiere de recursos y tiempo, ya que al generarse nuevos datos constantemente, este proceso tendría que ser periódico.

Por último, se corrieron los experimentos utilizando como población tan sólo el 10% de muestras aleatorias del *dataset*. Esto se hizo con la intención de optimizar el costo computacional en un 90% de entrada, sin la necesidad de una etapa de procesamiento fuera de línea y para comparar los resultados con los dos experimentos anteriores. La Figura 5.3 muestra los resultados de estos experimentos, se calculó el MAPE de la predicción respecto al valor esperado para las 1000 variables de prueba.

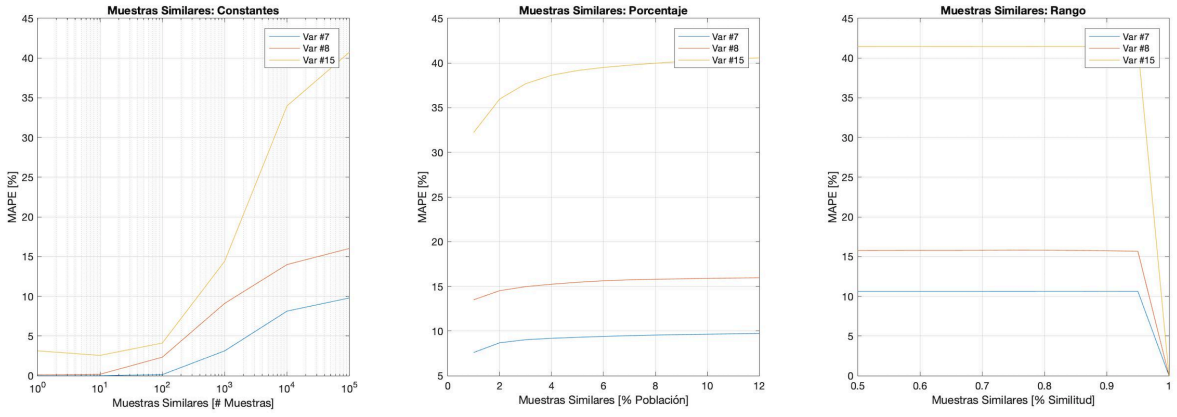

Figura 5.3 Error en experimentos que utilizaron 10% de muestras tomadas aleatoriamente de la población completa

En la primera gráfica, “Muestras Similares: Constantes”, de la Figura 5.3Figura 5.2 se puede observar que el error es bajo para las tres variables cuando se eligen menos de 1000 muestras similares, pero conforme

la cantidad aumenta, el error también aumenta. Además, se puede observar que para las variables 7 y 8 los resultados son muy parecidos y el error se encuentra dentro de un rango aceptable, sin embargo, para la variable 15 el error llega al 40%. La tendencia de comportamiento es similar a la gráfica análoga en la Figura 5.1, pero la ventaja es que se ahorraron 90% de los recursos computacionales.

En la segunda gráfica, "Muestras Similares: Porcentaje", de la Figura 5.3 se puede observar que el error varía muy poco independientemente del porcentaje de muestras similares que se usan en el experimento para calcular la recomendación. Al igual que en el caso anterior, los resultados para las variables 7 y 8 son muy parecidos y el error se encuentra dentro de un rango aceptable, sin embargo, el erro es mayor para la predicción de la variable 15.

En la tercera gráfica, "Muestras Similares: Rango", de la Figura 5.3 se puede observar que el error es prácticamente el mismo para las tres variables si se toman muestras con similitud desde 50% hasta el 95%. El error es 0 si se toman las muestras con similitud del 100%, sin embargo, no sería el método más conveniente, ya que el modelo no podría calcular una recomendación si el vehículo alcanza un estado nuevo, es decir, un estado en el que el usuario no haya provisto una preferencia anteriormente. Al igual que en los dos casos anteriores, los mejores resultados se obtuvieron para calcular la recomendación de las variables 7 y 8.

En general, los resultados de las tres variantes del experimento muestran que el modelo puede calcular mejores recomendaciones para las variables 7 y 8, es decir, al comparar el valor calculado contra el valor esperado, el error es menor y se encuentra dentro de un margen de error aceptable. De las tres variantes de este experimento, el mejor resultado se obtuvo cuando se toma 2% de la población como muestras similares para calcular la recomendación. Aunque los resultados que arrojó este experimento son muy similares a los dos anteriores, se puede decir que fueron los mejores ya que no requiere de un procesamiento fuera de línea y optimiza un 90% de los recursos computacionales requeridos.

La Tabla 5.1 muestra el MAPE resultante del método que utilizó la población de 10% de muestras aleatorias del *dataset* y para elegir las muestras similares utilizó un porcentaje de la población. Este experimento tiene los mejores resultados para calcular una recomendación, por lo que sería el modelo para utilizar como sistema de recomendación para este trabajo.

Tabla 5.1 MAPE del modelo que utilizó 10% de muestras aleatorias del *dataset* como población y un porcentaje de la población como muestras similares

Muestras	Variable 7	Variable 8	Variable 15
1%	7.59%	13.49%	32.22%
2%	8.68%	14.51%	35.96%
3%	9.02%	14.97%	37.68%
4%	9.19%	15.23%	38.63%
5%	9.30%	15.45%	39.15%
6%	9.40%	15.62%	39.50%
7%	9.48%	15.73%	39.76%
8%	9.54%	15.79%	39.98%
9%	9.60%	15.84%	40.16%
10%	9.64%	15.89%	40.32%
11%	9.68%	15.93%	40.46%
12%	9.73%	15.96%	40.57%

Los resultados de los experimentos indicaron que el mejor método para seleccionar la población es elegir 10% muestras del *dataset* aleatoriamente. La diferencia en los resultados fue mínima en comparación con utilizar toda la población, aunque el costo computacional es notablemente menor con solo el 10% de muestras. La utilización de los centroides de los clústeres se descartó debido a que el error en las recomendaciones es mayor y el alto consumo de tiempo de pre procesamiento de la población, este método no es factible para implementarlo en una solución en producción.

Para elegir las muestras similares, los experimentos mostraron que el mejor método es utilizar 1% o 2% de la población; la variación en el error de una recomendación fue mínima, sin embargo, se observó que al cambiar la variable a predecir se generaron mayores variaciones en el error. Por lo que puede ser un factor que necesite reajuste dependiendo de los datos generados por usuarios específicos para el modelo basado en similitud de usuarios.

5.1.2. PageRank

La implementación y la prueba del *PageRank* se hizo en dos partes: el modelo observador de estados y el modelo estimador de ambiente.

Para el modelo observador de estados, primero se implementó la matriz de transición correspondiente para el modelo como se muestra en la Figura 4.9. En Matlab, inicialmente la matriz cuadrada de tamaño *n* x *n* se definió para pocas variables y como una matriz de números reales, es decir reservando *n* x *n* espacio en memoria. Sin embargo, al momento de escalar el tamaño para cubrir todas las variables del *dataset*, las combinaciones totales son 51800000; por lo que la memoria que se tenía era insuficiente para alojar una matriz de 51800000 x 51800000 elementos. Así que para la implementación final se utilizó una matriz dispersa que se genera con la función sparse() en Matlab. Todas las operaciones matriciales están soportadas nativamente en Matlab para las matrices dispersas.

Ya que se tenía la matriz de transición del modelo observador de estados, se probó con 10000 muestras aleatorias como se muestra en la Figura 4.10. Al aplicar las variables de entrada se obtiene una predicción. Cada muestra de prueba tiene un valor esperado y para evaluar la exactitud del modelo, se comparó el valor de predicción del modelo contra el valor esperado. Se hicieron cuatro pruebas distintas para este modelo:

1. Predicción solo de la variable 7.
2. Predicción solo de la variable 8.
3. Predicción solo de la variable 15.
4. Predicción de las variables 7, 8 y 15.

La exactitud del modelo observador de estados se muestra en la Tabla 5.2. El costo computacional para generar la matriz de transición para el modelo es alto, debido a la cantidad de combinaciones de todas las variables. Sin embargo, se pudo optimizar al iterar solo en las combinaciones que tenían ocurrencias distintas a 0 en el *dataset*. Una vez que se tiene el modelo, el cómputo de una predicción es muy rápido, solo equivale a una multiplicación de matrices.

Para el modelo estimador de ambiente, también se implementó la matriz de transición correspondiente para el modelo, como se muestra en la Figura 4.9, con una matriz dispersa de 51800000 x 51800000 elementos. Con este método se optimizó el consumo de memoria considerablemente.

Ya que se tenía la matriz de transición del modelo estimador de ambiente, se probó con 10000 muestras aleatorias como se muestra en la Figura 4.10. Al aplicar las variables de entrada se obtiene una predicción. Cada muestra de prueba tiene un valor esperado y para evaluar la exactitud del modelo, se comparó el valor de predicción del modelo contra el valor esperado. Se hicieron cuatro pruebas distintas para este modelo:

1. Predicción solo de la variable 7.
2. Predicción solo de la variable 8.
3. Predicción solo de la variable 15.
4. Predicción de las variables 7, 8 y 15.

La exactitud del modelo estimador de ambiente se muestra en la Tabla 5.2. El costo computacional para generar la matriz de transición para el modelo es alto, debido a la cantidad de combinaciones de todas las variables. Sin embargo, se pudo optimizar al iterar solo en las combinaciones que tenían ocurrencias distintas a 0 en el *dataset*. Una vez que se tiene el modelo, el cómputo de una predicción es muy rápido, solo equivale a una multiplicación de matrices.

Tabla 5.2 Exactitud de los modelos observador de estados y estimador de ambiente

Modelo	Variable 7	Variable 8	Variable 15	Variables 7, 8 y 15
Estimador de ambiente	94.88%	79.88%	82.12%	66.03%
Observador de estados	94.87%	80.14%	83.16%	65.06%

En la Tabla 5.2 se puede observar que ambos modelos tienen resultados muy similares para las cuatro pruebas que se hicieron con ellos. La exactitud tiene una diferencia aproximada del 1% entre ambos modelos.

La predicción de la variable 7, estado del aire acondicionado, es la más exacta para ambos modelos. La predicción de la variable 8, apertura de la ventana, tiene una exactitud aproximada del 80% para ambos modelos. La predicción de la variable 15, intensidad del sensor de luz, tiene una exactitud aproximada del 82.5% para ambos modelos.

Con base en los resultados anteriores prácticamente se puede usar cualquiera de los dos modelos tanto para estimar el ambiente completo como predecir aisladamente el siguiente estado de las variables 7, 8 o 15.

La predicción de las variables 7, 8 y 15 simultáneamente tiene una exactitud aproximada del 65.5% para ambos modelos. Con base en los resultados prácticamente se puede usar cualquiera de los dos modelos tanto para estimar el ambiente completo como predecir aisladamente el siguiente estado de esta variable. Sin embargo, dada la exactitud se podría decidir hacer las tres predicciones independientemente, dependiendo de los resultados y restricciones computacionales que se tengan para ese caso de uso específicamente.

5.1.3. Predicción de Series de Tiempo con Redes Neuronales

La implementación de la red neuronal para predicción de series de tiempo se hizo como muestra la Figura 4.13. Mientras que los resultados se evaluaron con el proceso que aparece en la Figura 4.14.

Para este trabajo, se realizaron trece experimentos distintos con los parámetros de entrada que se muestran en la Tabla 4.4. Cada experimento se dividió en dos etapas: entrenamiento y prueba. En la etapa de entrenamiento, se entrenaba la red neuronal con el subconjunto de entrenamiento. Posteriormente, el modelo resultante del entrenamiento se probaba con el subconjunto de prueba.

La Tabla 5.3 muestra los resultados de todos los experimentos corridos. Se puede observar que el mejor modelo se obtuvo en el experimento 3, en el cual se utilizaron los siguientes parámetros: alfa = 0.01, lambda = 1e-6, épocas = 1000, 15 neuronas ocultas y función de activación tangente hiperbólica. El mejor modelo se define como aquel que tiene menor costo tanto en el entrenamiento como en la etapa de prueba para las tres variables que se están probando respecto a los resultados obtenidos para los otros modelos.

Tabla 5.3 Costo de entrenamiento y de prueba de los experimentos

#	Costo Entrenamiento			Costo Prueba			Tiempo
	Variable 7	Variable 8	Variable 15	Variable 7	Variable 8	Variable 15	(min)
1	1.6053E-02	1.6395E-02	3.1533E-05	1.5966E-02	1.6305E-02	3.1400E-05	280.166
2	1.5136E-02	1.6765E-02	2.8808E-06	1.5234E-02	1.6728E-02	2.8815E-06	1731.575
3	1.3672E-02	1.5844E-02	1.0301E-03	1.3598E-02	1.5889E-02	1.0500E-03	277.178
4	1.7382E-02	2.0247E-02	7.0012E-04	1.7382E-02	2.0198E-02	6.9757E-04	126.571
5	1.2993E-02	1.5335E-02	1.5283E-04	1.2952E-02	1.5340E-02	1.5358E-04	151.798
6	1.6958E-02	1.7523E-02	7.7779E-06	1.7087E-02	1.7729E-02	7.7517E-06	1013.481
7	2.1013E-02	2.0947E-02	1.8875E-02	9.8275E-01	9.8052E-01	7.0084E-02	116.570
8	1.8942E-02	2.1982E-02	8.8703E-03	9.6242E+00	4.1287E-01	9.3376E-01	139.994
9	1.4558E-02	1.5289E-02	3.0697E-06	1.4548E-02	1.5270E-02	2.9010E-06	385.291
10	1.4589E-02	1.5540E-02	1.8227E-06	1.4572E-02	1.5495E-02	1.6201E-06	385.422
11	1.4535E-02	1.5381E-02	2.4764E-06	1.4501E-02	1.5310E-02	1.8629E-06	383.986
12	1.4983E-02	1.5928E-02	2.5711E-06	1.4972E-02	1.5910E-02	2.3110E-06	642.129
13	1.5706E-02	1.7658E-02	3.0122E-04	-	-	-	2149.278

El experimento 1 se corrió con parámetros tomados arbitrariamente de un modelo ajeno a este trabajo. Ya que, al ser el primero, no se tenía ninguna referencia de cuales parámetros podían funcionar mejor, ni cuáles eran los costos esperados para el modelo de este trabajo. Este experimento se corrió solamente con 10 neuronas ocultas y 1000 épocas para que el tiempo de entrenamiento no fuera tan largo, aun así, el entrenamiento tomó 280 minutos. Los costos de entrenamiento y prueba de este experimento se usaron como referencia para los demás experimentos.

El experimento 2 se corrió con los mismos parámetros que el experimento 1, a excepción de la cantidad de épocas, que se incrementó a 10 veces. Los costos de entrenamiento y prueba mejoraron para las variables 7 y 15, pero aumentaron para la variable 8. En magnitud, la mejora no fue tan significativa comparada con el tiempo de entrenamiento que tomó este experimento; 6.18 veces respecto al experimento 1.

Al graficar la función de costo contra la cantidad de épocas durante el entrenamiento de la red neuronal para las tres variables de prueba en la Figura 5.4, se puede observar que la mejora del costo se observa más significativamente en las primeras épocas del entrenamiento. Por esta causa, en los experimentos posteriores se eligieron valores de épocas que van entre 500 y 2000 para reducir el tiempo de entrenamiento.

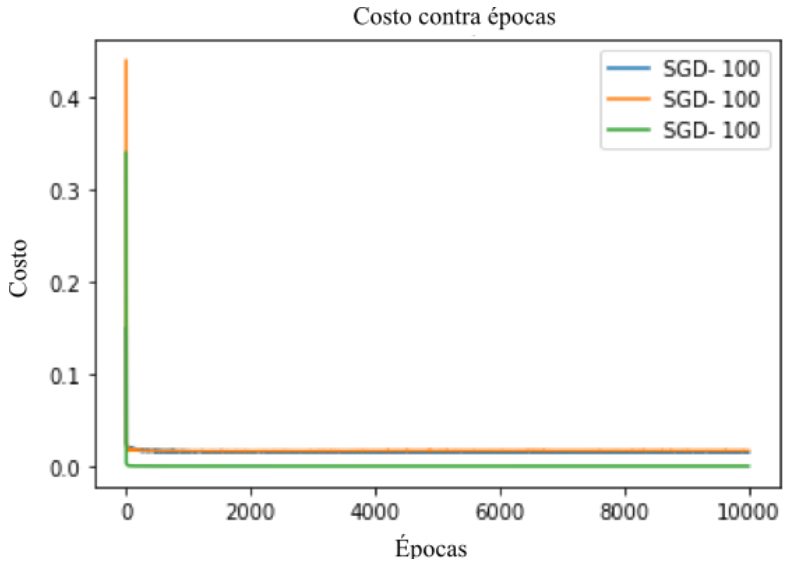

Figura 5.4 Gráfica de costo de entrenamiento contra épocas de entrenamiento para el experimento 2.

El experimento 3 se corrió con cambios en varios parámetros: alfa, lambda y cantidad de neuronas ocultas, respecto a los parámetros del experimento 1. Las funciones de costos del entrenamiento y prueba mejoraron para las tres variables y el tiempo de entrenamiento fue prácticamente el mismo respecto al experimento 1.

Para el experimento 4, los parámetros se eligieron con base en el experimento 3 que mostró los mejores resultados hasta ese momento. Los cambios en los parámetros fueron: la cantidad de neuronas ocultas y la función de activación. Como resultado, el tiempo de entrenamiento se recortó a la mitad respecto al experimento 3; sin embargo, tanto el costo de entrenamiento como el de prueba aumentaron inclusive respecto a los costos del experimento 1.

El experimento 5 cambió los parámetros de alfa y cantidad de neuronas ocultas respecto al experimento 4 con el propósito de mejor el costo de entrenamiento y prueba, pero manteniendo la ventaja de que la función de activación sigmoidea logística disminuye significativamente el tiempo de entrenamiento de la red neuronal. Los resultados de este experimento mostraron mejoras para las variables 7 y 8 en los costos de entrenamiento y prueba respecto al experimento 3, que tenía los mejores costos hasta el momento. El tiempo de entrenamiento también mejoró respecto al experimento 3, aunque no fue tan bueno como en el experimento 4.

El experimento 6 se corrió con los parámetros similares al experimento anterior, sólo disminuyeron la cantidad de épocas y la cantidad de neuronas ocultas para optimizar el tiempo de entrenamiento, ya que además se utilizó la técnica de validación cruzada con k=7. El subconjunto de entrenamiento se dividió en 7 subconjuntos, los cuales fueron intercalados como subconjuntos de entrenamiento y prueba para obtener el modelo final. El tiempo de entrenamiento de este modelo fue mucho más largo que cualquiera de los modelos corridos anteriormente y aunque se esperaba una mejora en la evaluación de la función de costo en el entrenamiento y prueba, en realidad no se obtuvo.

El experimento 7 utilizó los mismos parámetros que el experimento 1, a excepción de la función de activación. La razón para esto es que se quería probar si la función de activación ReLU mejoraba los costos

del modelo. El resultado del experimento es que se mejoró considerablemente el tiempo de entrenamiento, sin embargo, los costos de entrenamiento y prueba fueron malos comparados con el experimento 1. Así que ya no se corrieron más experimentos con la función de activación ReLU.

El experimento 8 utilizó los mismos parámetros que el experimento 1, a excepción de la función de activación. La razón para esto es que se quería probar si la función de activación PReLU mejoraba los costos del modelo. El resultado del experimento es que se mejoró considerablemente el tiempo de entrenamiento, sin embargo, los costos de entrenamiento y prueba fueron malos comparados con el experimento 1. Así que ya no se corrieron más experimentos con la función de activación PReLU.

El experimento 9 regresó a la función de activación de la tangente hiperbólica, pero cambiando el valor de lambda y las épocas respecto al experimento 3, que había tenido los costos de entrenamiento y prueba más bajos para esa función de activación. En los resultados se observa una mejora en los costos de entrenamiento y prueba de la variable 8, pero los costos para las variables 7 y 15 aumentaron respecto al experimento 3. Además, el tiempo de entrenamiento también aumentó ya que la cantidad de épocas se duplicó.

El experimento 10 cambió solamente la lambda respecto al experimento 9. Los costos de entrenamiento y prueba mejoraron para las variables 8 y 15 respecto al experimento 9. En cuanto al tiempo de entrenamiento, fue prácticamente el mismo ya que la función de activación y épocas fueron las mismas.

El experimento 11 cambió solamente la lambda respecto al experimento 10. Los costos de entrenamiento y prueba fueron prácticamente los mismos para las variables 7 y 8, sin embargo, para la variable 15 aumentaron. El tiempo de entrenamiento fue muy similar al de los experimentos 9 y 10 debido a que usan la misma función de activación y épocas.

El experimento 12 cambió el alfa respecto al experimento 11, con la intención de mejorar el costo para la variable 15. Pero los resultados fueron desfavorables para las tres variables; el costo de entrenamiento y prueba aumentó para todas. El tiempo de ejecución también se incrementó 1.67 veces respecto al experimento anterior.

El experimento 13 utilizó los mismos parámetros que el experimento 10, solamente se aumentaron las épocas a 9600 ya que se había observado una mejora en el costo de las variables 8 y 15, con la intención de seguir disminuyendo los costos de estas dos variables. De antemano se sabía que, al incrementar las épocas, aumentaría el tiempo de entrenamiento y así sucedió al correr el experimento. De este experimento se tienen costos de entrenamiento, los cuáles son altos para las tres variables en comparación con el experimento 12. No se tienen los costos de prueba, ya que el modelo de entrenamiento convergió a valores NaN para los pesos de las neuronas. Esto no permite evaluar el modelo con el subconjunto de prueba.

Después de correr el experimento 13, se decidió terminar de hacer más experimentos por que los cambios en los parámetros no generaban modelos con mejores costos y desde el experimento 3 ya se tenía un modelo con costos bajos para las tres variables.

En general, los experimentos muestran que el tiempo de entrenamiento de un modelo está directamente relacionado con la función de activación que es elija. La función de activación que minimiza el tiempo de entrenamiento es PReLU, después le sigue ReLU, seguida por la función sigmoidea logística y por último la tangente hiperbólica.

La cantidad de neuronas ocultas también impacta el tiempo de entrenamiento. A mayor cantidad de neuronas en la capa oculta, mayor será el tiempo de entrenamiento. Lo mismo aplica para las épocas de entrenamiento: a mayor cantidad de épocas, mayor es el tiempo de entrenamiento.

En cuanto a los parámetros de alfa y lambda, no se encontró ninguna relación directa en cuanto a los valores del parámetro y los costos de entrenamiento y prueba del modelo resultante.

5.2. Discusión

Con base en los resultados obtenidos se puede observar que cada uno de los modelos tiene ventajas y desventajas. Además, presentan mejores resultados para algunas predicciones respecto a los otros modelos.

El sistema de recomendación basado en la similitud de usuarios mostró que si se utiliza solo una parte de la población aleatoriamente los resultados son buenos en comparación con utilizar todas las muestras de la población. Esta es una ventaja del modelo desde el punto de vista computacional, ya que se optimiza el cálculo en un 90%, además de optimizar al mismo tiempo el uso de memoria para generar el modelo.

El sistema de recomendación basado en la similitud de usuarios predice la variable 7 con más exactitud, después la variable 8 y por último la variable 15.

El *PageRank* generó dos sub-modelos: observador de estados y estimador de ambiente. Ambos sub-modelos utilizan la misma matriz de transición y solo son las predicciones las que se computan distinto. Esto representa una ventaja, ya que con una sola matriz de transición se obtienen dos modelos; después del alto costo computacional que representa calcular la matriz de competencia, desde el punto de vista de tiempo de ejecución y consumo de memoria. En contraste, el costo computacional del cálculo de las predicciones es muy bajo, pues solo es una multiplicación de matrices.

Si se implementa el *PageRank*, se debe de tomar en cuenta que conforme aumenten la cantidad de variables, el tamaño de la matriz de transición aumentará; por lo que se deberían de elegir las variables a usar cuidadosamente para que la implementación de este modelo siga siendo factible.

Ambos sub-modelos del *PageRank* tuvieron resultados similares. Los sub-modelos predicen la variable 7 con más exactitud, después la variable 15, le sigue la variable 8 y por último la predicción de las 3 variables simultáneamente.

La modelo de red neuronal predice con mayor exactitud la variable 15, después la variable 7 y por último la variable 8. Una ventaja de este modelo es que simultáneamente puede predecir las tres variables, sin embargo, el tiempo de entrenamiento de la red neuronal puede ser muy largo. Esto significa que la etapa de entrenamiento tiene un alto costo computacional desde el punto de vista de ejecución, aunque no tanto en memoria. En contraste, una vez que se tiene el modelo, el costo computacional de una predicción es muy bajo.

6. CONCLUSIONES

En este capítulo se presentan las conclusiones de este trabajo y se proponen algunos puntos que podrían desarrollarse en trabajos futuros.

6.1. Conclusiones

En este trabajo se implementaron tres modelos distintos con el propósito de predecir las preferencias del usuario de algunas configuraciones de confort. Cada uno de los tres modelos presentados tiene ventajas y desventajas, pero dependiendo de cuál es el objetivo que se quiera lograr al implementarse en producción podría elegirse uno o más modelos que se adapten al caso de uso.

La implementación de modelos de aprendizaje automático en la industria automotriz es un tema abierto a muchas áreas y aún hay mucho trabajo por hacerse. Los modelos implementados en este trabajo utilizan técnicas de aprendizaje automático con el propósito de ampliar las áreas de investigación y aplicación en los vehículos. Se espera que este trabajo contribuya y en un futuro haya más trabajos similares.

Primero, de los *datasets* disponibles se identificaron aquellas variables que indican preferencias del usuario y el estado del vehículo. A partir de esta información se diseñaron e implementaron los modelos en este trabajo. Para implementar estos modelos en producción, se deberían obtener los valores de entrada vendrían de los sensores del vehículo directamente; dependiendo del modelo se necesitaría hacer algún preprocesamiento de la información que provea el sensor y después pasaría por el modelo para hacer las predicciones de las configuraciones del vehículo.

La implementación de los modelos para predicción de configuraciones de confort del vehículo sería una aplicación más dentro de la arquitectura IoV para los vehículos. Lo ideal sería implementar los tres modelos en conjunto para así mejorar la exactitud de predicción para todas las variables. Sin embargo, en producción la implementación dependerá de los recursos que se tengan en el vehículo. Los sistemas electrónicos de los vehículos son sistemas embebidos que tienen recursos computacionales y memoria limitada.

Aunque el sistema de recomendación basado en la similitud de usuarios tiene la ventaja que no necesita procesamiento fuera de línea, para implementarse en producción necesita poder computacional y memoria disponible en el sistema electrónico del vehículo.

Para su implementación en producción, ambos sub-modelos del *PageRank* podrían entrenarse fuera de línea en computadoras o servidores con gran poder computacional. El sistema electrónico del vehículo solo necesita tener suficiente memoria para almacenar la matriz de transición y poder computacional para realizar una multiplicación de matrices dispersas.

En el caso de la red neuronal, para su implementación en producción, el modelo se puede entrenar fuera de línea en computadoras o servidores con gran poder computacional. El sistema electrónico del vehículo solo necesita tener el poder computacional para aplicar el modelo y la memoria para guardar los pesos de dicho modelo, que se podría decir que es muy poca.

Si estos modelos se implementan como una aplicación de los vehículos en producción, se enfrentará con el problema del estado inicial del vehículo, cuando no se ha obtenido información acerca del comportamiento de ningún usuario. Para los modelos de *PageRank* y la red neuronal se puede utilizar una base de conocimiento promedio de otros usuarios mientras que se obtiene suficiente información del dueño para generar su modelo personalizado. En el caso del sistema de recomendación basado en la similitud usuarios podría optarse por tener muestras similares del promedio de usuarios hasta reunir las suficientes muestras del dueño.

Como alternativa, la aplicación de predicción podría estar deshabilitada cuando se manufactura el vehículo y estaría disponible únicamente después de que se tiene información suficiente del dueño para hacer predicciones exactas.

6.2. Trabajo Futuro

Aunque para este trabajo se desarrollaron tres modelos distintos que ya pueden ser implementados en producción, se prevé trabajo que puede desarrollarse a futuro sobre la misma línea de investigación.

Este trabajo propone tres modelos distintos que podrían ser utilizados individualmente, sin embargo, también se podrían utilizar combinados para mejorar la exactitud de la predicción. En el futuro, se puede investigar más acerca de la calidad de las predicciones utilizando varios modelos al mismo tiempo con algoritmos como *random forest*.

Otra línea de investigación que se abrió con este trabajo es encontrar el mejor método para reevaluar los modelos que requieren entrenamiento fuera de línea: el *PageRank* y la red neuronal. El método debe tomar en cuenta variables como el tiempo de entrenamiento, la cantidad de datos de entrada, la calidad de los nuevos datos generados, el costo computacional del entrenamiento, entre otros para que sea factible su implementación en producción.

Otro tema que quedó abierto a futura investigación es tomar en cuenta otros factores del contexto en la implementación de los modelos, como serían las estaciones del año que tienen comportamientos cíclicos o la ubicación geográfica de los usuarios.

Además, la implementación de los modelos podría hacerse en otros lenguajes de programación como Python, R o Spark que son *open source* y proveen más escalabilidad para implementar los modelos en producción.

BIBLIOGRAFÍA

[1] Tesla, «Tesla,» Tesla © 2017, 2017. [En línea]. Available: https://www.tesla.com/modelx. [Último acceso: 5 Nov 2017].

[2] National Traffic Law Center , «Investigation and Prosecution of Distracted Driving Cases (DOT HS 812 407),» National Highway Traffic Safety Administration, Washington DC, 2017.

[3] Mercedes-Benz, «Mercedes Benz USA,» ©2017 Mercedes-Benz USA, LLC, 2017. [En línea]. Available: https://www.mbusa.com/mercedes/vehicles/class/class-S/bodystyle-MAY. [Último acceso: 5 Nov 2017].

[4] Audi, «Audi México,» © 2017 AUDI AG, 2017. [En línea]. Available: http://www.audi.com.mx/mx/web/es/models/a8/a8-l-w12/interior.html#. [Último acceso: 5 Nov 2017].

[5] J. Yuan, «The Driver-Focused Human Machine Interface,» Janus Yuan MFA Thesis Project, 2014. [En línea]. Available: www.driverfocusedhmi.com. [Último acceso: 27 Diciembre 2017].

[6] Round Table Summary - Day 1, «Final workshop and exhibition presentations,» 15 Abril 2008. [En línea]. Available: http://www.aide-eu.org/pdf/final_workshop/day1/round_table/aide_day1_round_table_all.pdf. [Último acceso: 27 Diciembre 2017].

[7] M. Elbanhawi, M. Simic y R. Jazar, «In the Passenger Seat: Investigating Ride Comfort Measures in Autonomus Cars,» *IEEE Intelignet Transportation Systems Magazine,* vol. 7, nº 3, pp. 4-17, Fall 2015.

[8] The Royal Society for the Prevention of Accidents, «Cars In The Future : Human Machine Interface,» 14 Noviembre 2013. [En línea]. Available: https://www.rospa.com/road-safety/advice/vehicles/cars-in-the-future/human-machine-interface/. [Último acceso: 21 Enero 2018].

[9] AIDE: Recommendations for HMI Guidelines and Standards, «IST-1-507674-IP,» 1 Marzo 2004. [En línea]. Available: http://www.aide-eu.org/pdf/sp4_deliv_new/aide_d4_3_2.pdf. [Último acceso: 2018 Enero 23].

[10] McKinsey & Company, «Car data: paving the way to value-creating mobility,» Marzo 2016. [En línea]. Available: https://www.mckinsey.com/~/media/McKinsey/Industries/Automotive%20and%20Assembly/Our%20Insights/Creating%20value%20from%20car%20data/Creating%20value%20from%20car%20data.ashx. [Último acceso: 20 Enero 2018].

[11] M. N. O. Sadiku, M. Tembely y S. M. Musa, «Internet of Vehicles,» *International Journals of Advanced Research in Computer Science and Software Engineering,* vol. 8, nº 1, pp. 11-13, 2018.

[12] N. Chowdhary y P. Deep Kaur, «Addressing the Characteristics of Mobility Models in IoV for Smart City,» de *International Conference on Computing, Communication and Automation (ICCCA)*, Noida, India, 2016.

[13] G. Adomavicius y A. Tuzhilin, «Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions,» *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, nº 6, pp. 734-749, 6 Jun 2005.

[14] W. Khan, A. Daud, J. A. Nasir y T. Amjad, «A survey on the state-of-the-art machine learning models in the context of NLP,» *Kuwait Journal of Science,* vol. 43, nº 4, pp. 95-113, 2016.

[15] P. Zhen, S. Yuan, X. Wu, J. Li y A. Lu, «One-Class Adversarial Nets for Fraud Detection,» CoRR, 5 Junio 2018. [En línea]. Available: http://arxiv.org/abs/1803.01798. [Último acceso: 16 Febrero 2019].

[16] Y. Mirsky, T. Doitshman, Y. Elovici y A. Shabtai, «Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection,» CoRR, 27 Mayo 2018. [En línea]. Available: http://arxiv.org/abs/1802.09089. [Último acceso: 16 Febrero 2019].

[17] McKinsey & Company, «McKinsey,» © 1996-2017 McKinsey & Company, Oct 2013. [En línea]. Available: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers. [Último acceso: 5 Nov 2017].

[18] F. Ricci, L. Rokach, S. Bracha y P. B. Kantor, Recommender Systems Handbook, Boston, MA: Springer, 2011.

[19] C. A. Gomez-Uribe y N. Hunt, «The Netflix Recommender System: Algorithms, Business Value, and Innovation,» *ACM Transactions on Management Information Systems,* vol. 6, nº 4, pp. 13-13.19, 2016.

[20] C. C. Aggarwal, Recommender Systems, The Textbook, Springer International Publishing , 2016.

[21] B. Smith y G. Linden, «Two Decades of Recommender Systems at Amazon.com,» *IEEE Internet Computing,* vol. 21, nº 3, pp. 12-18, 2017.

[22] J. Davidson, B. Liebald, J. Liu, P. Nandy y T. V. Vleet, «The YouTube video recommendation system,» de *Proceedings of the fourth ACM conference on Recommender systems*, Barcelona, 2010.

[23] J. Leskovec, A. Rajaraman y J. D. Ullman, Mining of Massive Datasets, Cambridge University Press, 2014.

[24] D. M. Almeida, P. S. G. De Mattos Neto y D. C. Cunha, «Hybrid Time Series Forecasting Models Applied to Automotive On-Board Diagnostics Systems,» de *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, 2018.

[25] J. Bao, W. Chen, Y.-s. Shui y Z.-t. Xiang, «Complexity analysis of traffic time series based on multifractality and complex network,» de *2017 4th International Conference on Transportation Information and Safety (ICTIS)*, Banff, AB, Canada, 2017.

[26] O. Simeone, «A Very Brief Introduction to Machine Learning With Applications to Communication Systems,» CoRR, 2 Sep 2018. [En línea]. Available: http://arxiv.org/abs/1808.02342. [Último acceso: 3 Mar 2019].

[27] M. Montaner, B. López y J. L. De La Rosa, «A Taxonomy of Recommender Agents on the Internet,» *Artificial Intelligence Review,* vol. 19, nº 4, pp. 285-330, 2003.

[28] W. R. Gilks, D. Spiegelhalter y S. Richardson, Markov Chain Monte Carlo in Practice, Taylor & Francis, 1995.

[29] D. Gamerman y H. F. Lopes, Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference, Second Edition, Taylor & Francis, 2006.

[30] R. Site y J. Site, «Neural Network Systems Technology in the Analysis if Financial Time Series",» de *Intelligent Knowledge-Based Systems: Business and Technology in the New Millennium*, Boston, MA, Springer US, 2005, pp. 1564-1615.

[31] K.-K. Seo, «Neural Network Systems Technology and Applications in Product Life-Cycle Cost Estimates,» de *Intelligent Knowledge-Based Systems: Business and Technology in the New Millennium*, Boston, MA, Springer US, 2005, pp. 1543-1563.

[32] T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[33] X. Glorot y Y. Bengio, «Understanding the difficulty of training deep feedforward neural networks,» de *n Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[34] R. Vitor, «Car trips data log,» Kaggle, 2018. [En línea]. Available: https://www.kaggle.com/vitorrf/cartripsdatamining. [Último acceso: Abril 2018].