UCUENCA

Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería en Telecomunicaciones

Mejora de la privacidad en Sistemas Inteligentes de Transporte (ITS) mediante aprendizaje distribuido: caso de estudio sistema de control de emisiones en la zona urbana de Cuenca

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

Autores:
Esteban Ricardo Arcos Salamea
David Sebastián González Saguay

Director:
Pablo Andrés Barbecho Bautista
ORCID: 0000-0002-5281-9208

Cuenca, Ecuador

2024-08-27

Resumen

Actualmente, el mundo está en proceso de transición hacia un entorno interconectado, lo que ha aumentado la disponibilidad de grandes volúmenes de datos y ha subrayado la importancia de proteger la privacidad. En el ámbito de la movilidad, los *Intelligent Transportation Systems (ITS)* son un ejemplo significativo de cómo se genera una gran cantidad de datos en redes interconectadas.

Este trabajo de titulación propone de arquitectura basada en técnicas de aprendizaje automático, específicamente en el Aprendizaje Federado (FL), para mejorar la privacidad de los usuarios de ITS, con un enfoque en la reducción de emisiones de CO_2 en Cuenca, Ecuador. La metodología empleada permite que los datos permanezcan en los dispositivos locales de los usuarios, compartiendo únicamente los parámetros del modelo, lo que garantiza una mayor protección de la privacidad. El uso de diversos algoritmos de aprendizaje automático facilita la segmentación y clasificación de datos, reduciendo así el tiempo de espera de los vehículos y contribuyendo a la disminución de las emisiones de CO_2.

Los resultados obtenidos muestran que el FL consume más recursos en comparación con el *Reinforcement Learning (RL)*. El uso promedio de CPU en FL es notablemente mayor, con valores cercanos al 41 %, en contraste con el 16.69 % del RL. Además, el FL utiliza un 8.52 % más de Memoria RAM que el RL. Esto indica que, aunque el FL ofrece una mayor protección de la privacidad de los datos, puede afectar la eficiencia del sistema. Asimismo, los resultados revelan que en el contexto de la privacidad diferencial, un mayor valor de ϵ está asociado con una menor privacidad. Por lo tanto, es fundamental encontrar un valor óptimo de ϵ que permita equilibrar la protección de la privacidad con la eficiencia del sistema en el Aprendizaje Federado.

Palabras clave del autor: aprendizaje federado, privacidad de datos, transportación urbana, reducción de emisiones

Abstract

Currently, the world is transitioning towards an interconnected environment, which has increased the availability of large volumes of data and highlighted the importance of protecting privacy. In the realm of mobility, ITS are a significant example of how large amounts of data are generated in interconnected networks.

This thesis proposes an architecture based on machine learning techniques, specifically FL, to enhance the privacy of ITS users, with a focus on reducing CO_2 emissions in Cuenca, Ecuador. The methodology employed allows data to remain on users' local devices, sharing only model parameters, thus ensuring greater privacy protection. The use of various machine learning algorithms facilitates data segmentation and classification, thereby reducing vehicle wait times and contributing to decreased CO_2 emissions.

The results obtained show that FL consumes more resources compared to RL. The average CPU usage in FL is significantly higher, with values close to 41 %, in contrast to 16.69 % in RL. Additionally, FL uses 8.52 % more RAM than RL. This indicates that, while FL offers better privacy protection, it can impact system efficiency. Furthermore, the results reveal that in the context of differential privacy, a higher value of ϵ is associated with lower privacy. Therefore, finding an optimal ϵ value is crucial to balance privacy protection with system efficiency in Federated Learning.

Author Keywords: federated learning, data privacy, urban transportation, emission reduction

Índice de contenido

Índice de figuras

Índice de tablas

Agradecimientos

En este significativo momento de culminación de nuestro trabajo de titulación, deseamos expresar nuestro más sincero agradecimiento a todas las personas que han contribuido de manera significativa al desarrollo y éxito de este trabajo.

En primer lugar, queremos agradecer a nuestro tutor de tesis, Ing. Pablo Barbecho, por su orientación constante, paciencia y apoyo a lo largo de todo el proceso. Su vasta experiencia y conocimientos en el campo resultaron fundamentales para enriquecer nuestro trabajo y orientarnos adecuadamente. Agradecemos profundamente su tiempo, su disposición para aclarar nuestras dudas y su valiosa retroalimentación, que fue importante para perfeccionar nuestro trabajo.

Agradecemos a Lucas Alegre, desarrollador de sumo-rl, librería a partir de la cual implementamos nuestro trabajo. A la comunidad de Flower cuyos comentarios y desarrollos fueron de ayuda para cumplir nuestros objetivos. También a nuestros profesores y compañeros de la Facultad de Ingeniería de la Universidad de Cuenca por sus comentarios constructivos y su colaboración, que enriquecieron significativamente nuestro trabajo.

Dedicatoria

A mis padres, Richard y Diana, que con su apoyo incondicional desde el primer día hasta el último han sido fundamentales en mi trayectoria académica. Sus consejos y enseñanzas han sido muy importante para culminar esta etapa académica. A mis abuelos, Raúl y Elsa, que siempre me mostraron su apoyo en cada paso que daba y por preocuparse siempre de mí. A mi hermano Sebas y mi amigo Manolo, por acompañarme y animarme en las noches de desvelo. A mi abuelo Abdón, que sé que me estará observando desde algún lugar, por enseñarme a defender mis ideales y creer siempre en mí. A mi compañera de toda esta aventura, Fernanda, por su motivación constante, por mostrarme que todo es posible y sobre todo por brindarme apoyo en los momentos más difíciles de esta etapa. Finalmente, pero no menos importante, a mis amigos David, Erick, Jean, Juan Diego, Lucas y los Henry's, por brindarme su apoyo en esta etapa, haciendo que sea más llevadera, especialmente en situaciones complicadas. Cada uno de ustedes es especial para mí, este logro no es solo mío, es de todos ustedes que me acompañaron en este camino, ya que sin ustedes esto no habría sido posible.

Esteban

A mi pa, a mi ma, a mis ñañas. Por todo. A mis amigos Erick, Esteban, Jean, Juan Diego, Lucas y los Henry's, por los recuerdos creados en las aulas y fuera de ellas. A Doménica, por ser semilla. A Nicolás, Andrés, Hugo, Freddy, Xavier, Dayanna, Karla, Jonnathan y David, por su compañía y motivación en estos años, por estar ahí desde el momento en el que se fundó este sueño, que por entonces parecía muy lejano. A Davicho, Gonza, Cholo, por ser yo. Por cumplir las metas de aquel niño curioso, por contener multitudes, por estar siempre bien. A todos quienes formaron parte de esta historia, a todas las pequeñas partes de este hombre inconcluso: familiares, mascotas, amigos, compañeros y profesores.

David G

Glosario

5G Quinta Generación. 13

API Application Programming Interface. 46

DL Deep Learning. 35, 36

DP Differential Privacy. 32, 78

DSRC Dedicated Short-Range Communications. 13

FL Aprendizaje Federado. 2, 3, 9, 15, 16, 27, 29, 32, 34, 35, 53, 74, 79–81

IA Inteligencia Artificial. 14, 36

ITS Intelligent Transportation Systems. 2, 3, 7, 9, 13–16, 18, 19, 33, 35–38, 82

ML Machine Learning. 14, 30, 35, 36

NR-CV2X New Radio - Cellular Vehicle-to-Everything. 13

PCA Análisis de Componentes Principales. 20, 21, 49, 50, 56, 65, 70, 71

QL Q Learning. 25, 43

RL Reinforcement Learning. 2, 3, 8, 24, 33, 43, 52–54, 64, 74, 79–81, 100

SUMO Simulation of Urban MObility. 14, 15, 32, 33, 44, 48

TraCI Traffic Control Interface. 16, 33, 44, 46

V2I Vehicle-to-Infrastructure. 32

V2V Vehicle-to-Vehicle. 32

V2X Vehicle-to-Everything. 13

1. Generalidades

En una sociedad cada vez más interconectado, la gestión eficiente de datos y la protección de la privacidad se han convertido en prioridades cruciales. Los Sistemas de Transporte Inteligente (ITS), que utilizan tecnologías avanzadas para optimizar el flujo de tráfico y la seguridad vial, son un claro ejemplo de cómo la tecnología puede transformar el transporte urbano. No obstante, esta transformación viene acompañada de desafíos significativos, particularmente en lo que respecta a la protección de la privacidad de los datos de los usuarios.

En este capítulo, se abordará cómo los ITS, al generar y manejar grandes volúmenes de datos, enfrentan el dilema de equilibrar la funcionalidad con la privacidad. A través de la introducción de técnicas de aprendizaje automático y en particular el Aprendizaje Federado (FL), se explora una solución innovadora que promete mitigar estos desafíos. La sección que sigue proporciona una visión detallada del papel de los ITS en la movilidad moderna y cómo el FL puede ser una herramienta clave para mejorar la privacidad sin comprometer la eficiencia del sistema. Además, se presentarán los *objetivos* de este trabajo de titulación, el *alcance* de la investigación, la *justificación* del estudio y los *antecedentes* relevantes.

1.1. Introducción

El mundo actual se encuentra inmerso en una profunda evolución hacia un entorno altamente interconectado, por lo que existe una disponibilidad de cantidades masivas de datos, esto hace que la protección de privacidad de las personas sea más necesaria que nunca. En el campo de la movilidad, los Sistemas de Transporte Inteligente (conocidos por sus siglas en inglés como ITS), que han ganado gran importancia en los últimos años, son un claro ejemplo de la producción de datos en redes interconectadas [1].

Los ITS están fundamentados en sistemas de comunicaciones inalámbricas como *Dedicated Short-Range Communications (DSRC)* y *New Radio - Cellular Vehicle-to-Everything (NR-CV2X)*. La tecnología DSRC está basada en el estándar IEEE 802.11p [2]. Tiene como objetivo la comunicación entre vehículos para la prevención de colisiones. Usa los datos de trayectoria de vehículos vecinos y considerando su propia trayectoria calcula la probabilidad de que exista una colisión [3]. NR-CV2X se define como un progreso avanzado en las comunicaciones *Vehicle-to-Everything (V2X)* en el marco de la tecnología de quinta generación (5G), particularmente en el ámbito del *3rd Generation Partnership Project (3GPP)*. Este enfoque, fundamentado en la tecnología de Quinta Generación (5G) NR *(New Radio)*, presenta mejoras

sustanciales en la capacidad de comunicación V2X. Estas mejoras incluyen técnicas robustas de conformación de haz como la conformación de haz de Capon que minimizan el impacto de incertidumbres y ruido, y el uso de Filtros de Kalman Extendidos (EKF) para un seguimiento más preciso de los vehículos en movimiento. [4]

Las herramientas usadas en el despliegue de ITS se dedican a la recopilación, procesamiento, integración y provisión de información para operadores, autoridades de tráfico, proveedores de transporte público y comercial, así como para usuarios individuales [5]. Estos datos incluyen información crítica, como registros de tiempo, coordenadas geográficas, altitud [6], áreas de alta densidad de tráfico, longitud de los trayectos y velocidad promedio [7]. La gestión adecuada de la privacidad en relación con estos datos es de vital importancia, ya que pueden revelar datos sensibles, como la ubicación en tiempo real de individuos, sus rutas habituales y velocidades de desplazamiento. Por tanto, se vuelve esencial garantizar que estos datos estén protegidos de posibles amenazas.

El uso de simuladores de tráfico es una herramienta pertinente en la optimización de procesos de transporte para diseñar nuevas infraestructuras viales y reconstruir las existentes. A medida que los sistemas de transporte se vuelven más complejos y congestionados, las simulaciones de tráfico se utilizan cada vez más. Los simuladores de tráfico permiten modelar una versión digital de una ciudad, creando un modelo funcional del tráfico que corresponde al movimiento real en las carreteras. Esto es fundamental, ya que probar nuevas estrategias y soluciones en un entorno real puede ser costoso, arriesgado y logísticamente complicado. Los simuladores ofrecen una forma segura y eficiente de evaluar y mejorar las infraestructuras y políticas de tráfico sin interferir con el tráfico real [8].

Los datos generados por los ITS requieren ser analizados por herramientas externas, como lo es *Simulation of Urban MObility (SUMO)*. SUMO es un paquete de simulación de tráfico de código abierto que permite modelar y simular la movilidad urbana con base en datos de diferentes fuentes de entrada. Se utiliza para analizar y comprender patrones de tráfico, diseñar soluciones para la gestión eficiente del tráfico y evaluar el impacto de diversas estrategias de movilidad en entornos urbanos. Además, dispone de un modelo de emisión de ruido, un modelo de emisión de contaminantes, consumo de combustible y consumo de energía [9].

La aplicación de Inteligencia Artificial (IA) y *Machine Learning (ML)* en ITS puede revolucionar estos sistemas al mejorar la gestión del tráfico, prevenir accidentes y optimizar el consumo de recursos [10]. Sin embargo, estos avances no están exentos de desafíos, especialmente en términos de privacidad y seguridad. En entornos de comunicaciones inalámbricas móviles, la

información de los vehículos puede estar expuesta a riesgos constantes de interceptación o manipulación. Este problema se agrava en aplicaciones que requieren compartir datos entre múltiples dispositivos y sistemas [11].

En este escenario de creciente interconexión y datos masivos, el FL, emerge como una solución prometedora para la preservación de la privacidad de los usuarios de los ITS. El FL, al operar con algoritmos de inteligencia artificial distribuida, ha demostrado su capacidad para preservar la privacidad de los datos sin comprometer su utilidad [1]. Esta metodología permite el análisis y la generación de conocimiento sin la necesidad de compartir información sensible de manera centralizada. Dentro del marco de los ITS, el enfoque del FL busca garantizar la protección de los datos sensibles [7], mientras se mantiene la capacidad de análisis para lograr una gestión segura y avanzada del tráfico vehicular.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar una arquitectura basada en técnicas de aprendizaje automático, con un enfoque en el Aprendizaje Federado (FL), para mejorar la privacidad de los usuarios en el contexto de los Sistemas de Transporte Inteligente (ITS).

1.2.2. Objetivos Específicos

- Recopilar información en la literatura científica acerca de los potenciales problemas a los que se enfrentan los ITS respecto a la privacidad de datos.
- Evaluar diferentes técnicas de aprendizaje de máquina aplicadas a los ITS, incluyendo aprendizaje supervisado, no supervisado y por refuerzo.
- Definir métricas para medir la privacidad y el rendimiento del sistema, estableciendo un equilibrio entre la recopilación de datos necesarios para el funcionamiento eficiente del ITS y la protección de la privacidad de los usuarios.
- Implementar el sistema de control de emisiones para la zona urbana de Cuenca en el entorno simulado.
- Diseñar y evaluar la arquitectura que mejora la privacidad de los ITS, utilizando simulación como enfoque principal haciendo uso de la herramienta SUMO y el Framework

Flower FL, implementando una API de alto nivel utilizando Python *Traffic Control Interface (TraCI)* para la comunicación entre estos.

1.3. Alcance

El presente trabajo de titulación se enfoca en la aplicación de técnicas de aprendizaje automático, con énfasis en el FL, como una solución para abordar los desafíos de privacidad en los ITS.

Se propone llevar a cabo una comparación entre un ITS que emplea técnicas de aprendizaje de máquina centralizadas y otro que utiliza técnicas de aprendizaje de máquina distribuidas. La metodología implica un enfoque experimental utilizando el software de simulación SUMO y Flower FL. Se evaluarán los algoritmos propuestos utilizando métricas convencionales, como lo son el consumo de recursos, privacidad y seguridad. Además, se establecerá una interfaz de programación de aplicaciones (API) entre SUMO y Flower para llevar a cabo la simulación correspondiente.

1.4. Justificación

El FL se elige debido a su capacidad para preservar la privacidad de los datos, al tiempo que permite el análisis y la generación de conocimiento sin la necesidad de compartir información sensible de manera centralizada. Se plantea la hipótesis de que la implementación de inteligencia artificial distribuida mejora la privacidad de los usuarios de los ITS, al mismo tiempo que preserva la utilidad de los datos para la realización de predicciones y la identificación de patrones.

La suposición subyacente en esta hipótesis es que el sistema que implementa el FL podría tener un rendimiento inferior en comparación con el sistema que transmite los datos sin restricciones. No obstante, se espera que esta pérdida de rendimiento se vea compensada por la mejora en la privacidad de los datos utilizados.

1.5. Antecedentes del Proyecto

En lo que respecta al ITS en el que se evaluarán los algoritmos propuestos, se basa en datos previamente recopilados del tráfico vehicular en la ciudad de Cuenca. El ITS se centra en dos

parámetros principales: la actualización de las rutas de los vehículos para evitar la exposición a la contaminación (política de elección de ruta) y la actualización de los límites de velocidad de los vehículos (política de control de velocidad) [12].

Asimismo, los algoritmos propuestos serán evaluados mediante el empleo de métricas convencionales como la varianza explicada y la pérdida de valor, ambas reconocidas para la evaluación de modelos predictivos. La varianza explicada determina si la política aprendida constituye un predictor eficaz del rendimiento o la recompensa total. Valores inferiores a cero indicarán un desempeño peor que la ausencia de predicción, mientras que valores cercanos o iguales a uno denotan una predicción acertada. En paralelo, la métrica de pérdida de valor, cuya disminución señala predicciones más precisas del valor asociado a la política actual, deberá ser elevada durante la fase de aprendizaje del agente, decreciendo posteriormente una vez que la recompensa se haya estabilizado [13].

2. Marco Teórico

En este capítulo se abordarán todos los conceptos teóricos considerados de importancia, los cuales permitieron desarrollar el presente trabajo de titulación.

2.1. Intelligent Transportation Systems

Los Sistemas de Transporte Inteligente, también conocidos por sus siglas en inglés como *Intelligent Transportation Systems (ITS)*, constituyen un conjunto innovador de tecnologías en el ámbito de las Telecomunicaciones e Informática, con el propósito de potenciar la eficiencia, seguridad y sostenibilidad del transporte. Los ITS se vuelven cada vez más indispensables ante la rápida expansión de métodos avanzados de aprendizaje automático y la aparición de nuevas fuentes de datos en constante evolución [14].

Una visión general sobre estos sistemas se puede observar en la Figura 2.1, donde se observan diferentes iconos sobrepuestos que representan componentes y funciones de los Sistemas de Transporte Inteligente (ITS). Entre ellos, se incluyen el *cloud computing* para el análisis de datos en tiempo real, coches conectados que interactúan con la infraestructura vial y otros vehículos, y sistemas de geolocalización para la navegación precisa. Además, se visualizan símbolos de transporte público, señalización inteligente, y dispositivos de monitoreo como sensores y cámaras. Estos elementos reflejan cómo los ITS integran tecnologías avanzadas para mejorar la eficiencia, seguridad y sostenibilidad del transporte urbano, optimizando el flujo de tráfico y reduciendo emisiones de CO_2, en un entorno donde se considera tanto a vehículos privados como al transporte público y a los peatones.

El objetivo principal de este sistema se centra en optimizar la administración de los recursos urbanos y mejorar la experiencia de las personas mediante la implementación de servicios de información y alerta. En este sentido, esta mejora no solo favorece la fluidez del tráfico en la ciudad, reduciendo el tiempo perdido en congestiones, sino que también conlleva una disminución significativa en el consumo de combustible, las emisiones de CO_2 y las pérdidas económicas [15].

Figura 2.1: ITS. Fuente: [16]

En Ecuador, la adopción de Sistemas de Transporte Inteligente (ITS) ha mostrado un aumento reciente, pero sigue siendo limitada. A pesar del crecimiento en la investigación sobre *Big Data* desde 2019, la implementación práctica de ITS enfrenta varios desafíos, como la falta de proyectos consolidados y la necesidad de una mayor inversión en tecnología [17]. En ciudades como Quito y Guayaquil, se han desarrollado proyectos como la implementación de sistemas de control de tráfico y monitoreo en tiempo real, además de la integración de datos de movilidad para mejorar la planificación urbana. Sin embargo, la adopción efectiva de ITS continúa enfrentando barreras, y es necesario seguir avanzando en la inversión y desarrollo para superar estas limitaciones.

2.2. Niveles Usuales de Emisión de CO2 en Entornos Urbanos

Las emisiones de CO2 de los automóviles están estrechamente relacionadas con la velocidad de circulación. Según [18], las emisiones de CO2 son directamente proporcionales al consumo de combustible, alcanzando los consumos mínimos a velocidades comprendidas entre 80 y 100 km/hora. Sin embargo, en entornos urbanos, donde la velocidad de circulación es generalmente más baja, el consumo de un vehículo de turismo medio puede ser típicamente un 60 % superior al alcanzado en el régimen óptimo, lo que implica una incidencia significativa del tráfico urbano sobre el total de emisiones. Se estima que el tráfico urbano puede representar más de la mitad de las emisiones totales, a pesar de que el porcentaje del kilometraje recorrido en las ciudades es mucho menor.

Por otro lado, en el Área Metropolitana del Valle de Aburrá (AMVA) se realizaron 5,6 millones de viajes diarios en el año 2012, responsables de la emisión de 3.545.633 kg de CO2 diarios [19]. Esta cantidad de emisiones no se distribuye uniformemente, ya que los viajes en automóvil, aunque representan solo el 15 % de los viajes, son responsables del 55,2 % de las emisiones. En términos grupales, los medios de transporte privados, responsables del 26 % de la movilidad, emiten el 60 % del CO2, mientras que los medios de transporte públicos, que cubren el 48 % de la movilidad, emiten el 40 % del CO2. Para poner en perspectiva los valores obtenidos en simulaciones de emisiones en entornos urbanos, se puede considerar que un viaje en automóvil en el AMVA puede emitir hasta ocho veces más CO2 que un viaje en bus y tres veces más que un desplazamiento en metro. Además, un desplazamiento promedio por el motivo de Comer o tomar algo puede emitir alrededor de 861 g de CO2, en comparación con 750 g de CO2 por un desplazamiento promedio por motivo de Trabajo y 300,9 g de CO2 por un desplazamiento promedio por motivo de Estudio.

2.3. Enfoques de Aprendizaje de Máquina

El aprendizaje supervisado, no supervisado y por refuerzo son enfoques fundamentales del aprendizaje automático, cada uno con características y aplicaciones distintas [20]. El aprendizaje supervisado se basa en datos etiquetados, donde el modelo aprende a predecir resultados a partir de ejemplos con entradas y salidas conocidas. Se utiliza en tareas como la clasificación *(Random Forest)* y la regresión (Regresión Lineal) [21]. El aprendizaje no supervisado, por otro lado, trabaja con datos sin etiquetar, buscando patrones y estructuras ocultas sin guía externa [22]. Se aplica en problemas como la agrupación de clientes en segmentos de mercado y la reducción de dimensionalidad como el Análisis de Componentes Principales (PCA) [23]. En contraste, el aprendizaje por refuerzo no se basa en un conjunto de datos estático; en su lugar, un agente aprende a tomar decisiones a través de la interacción con un entorno dinámico, recibiendo recompensas o castigos en función de sus acciones [24]. Este enfoque es utilizado en aplicaciones como el control de robots, la optimización de estrategias en juegos y la gestión de sistemas complejos, donde las decisiones secuenciales son críticas. Se pueden encontrar algoritmos como *Q-learning*. Mientras que los dos primeros enfoques se centran en el análisis de datos preexistentes, el aprendizaje por refuerzo se enfoca en la mejora continua de la toma de decisiones basada en la retroalimentación recibida del entorno. A continuación se explica con detalle cada uno de los enfoques de aprendizaje de máquina.

2.3.1. Aprendizaje No Supervisado

El aprendizaje no supervisado se distingue por la presencia de datos no etiquetados, es decir, aquellos que carecen de asignaciones explícitas. El propósito fundamental de este enfoque es extraer información significativa de la base de datos o conjunto de datos disponible [22].

En este aprendizaje, se busca que el algoritmo pueda encontrar patrones en el conjunto de datos, para posteriormente clasificar los datos. Algunos de los problemas de este tipo de aprendizaje son: asociación, detección de anomalías y problemas de autoencoder [25]. Ejemplos de estos algoritmos son el PCA, que se utiliza para la reducción de dimensionalidad al transformar los datos a un nuevo espacio con menos dimensiones, preservando la mayor varianza posible [23]. Otro ejemplo es el *K-means*, un algoritmo de agrupamiento que divide los datos en k grupos basados en características similares [26].

2.3.1.1. Análisis de Componentes Principales (PCA)

El PCA es un ejemplo de aprendizaje no supervisado que se utiliza en el análisis de posibles relaciones entre diversas variables cuantitativas. Con esta técnica se busca reducir el número de variables originales a un conjunto de combinaciones lineales más pequeño, mostrando así una explicación de la estructura de varianza del amplio volumen de datos medidos [23].

El PCA fue incorporado por Pearson [27], en el contexto de variables no aleatorias, para luego ser utilizado en variables aleatorias por Hotelling [28]. Utilizando este análisis se reduce la dimensionalidad de un conjunto de datos, preservando la varianza. Esto se logra mediante una transformación ortogonal, la cual convierte los datos en nuevos índices, los cuales pueden ser conocidos como componentes principales (CPs), que buscan cumplir dos criterios fundamentes: el primero, que cada CP es una combinación lineal de las variables originales, y el segundo que los CPs son mutuamente no correlacionados. Utilizando este método, el primer CP captura la mayor cantidad de varianza del conjunto de datos originales y cada CP subsiguiente contiene la variabilidad no capturada por sus predecesores [29].

2.3.2. Aprendizaje Supervisado

El aprendizaje supervisado es ampliamente utilizado en problemas de clasificación, ya que comúnmente se busca que la máquina aprenda un sistema de clasificación predefinido. El

objetivo principal suele ser construir un estimador capaz de predecir la etiqueta de un objeto basándose en un conjunto de características proporcionadas. Una vez logrado esto, el proceso de aprendizaje supervisado utiliza este conjunto de características junto con las salidas correctas correspondientes para aprender, comparando su salida predicha con las salidas reales para identificar posibles errores. Aunque el modelo creado no es imprescindible mientras se disponga de las entradas, la ausencia de algunos valores de entrada impide inferir conclusiones sobre las posibles salidas [21].

En este enfoque algorítmico, se identifica un atributo específico cuyo propósito es utilizar los datos disponibles para anticipar el valor de dicho atributo en instancias aún no observadas o de las cuales se carece de información. Estos datos, caracterizados por tener asignaciones explícitas a dicho atributo, se denominan con etiquetas [22].

En el proceso de Aprendizaje Supervisado, una etapa crítica es la preparación y preprocesamiento de los datos. Esto implica asegurarse de que los datos estén en un formato adecuado y limpio antes de aplicar algoritmos de aprendizaje. Los investigadores han desarrollado diversas técnicas para abordar desafíos comunes, como la presencia de datos faltantes o valores atípicos (ruido) en los conjuntos de datos [30].

Existen varios algoritmos populares en aprendizaje supervisado utilizados para la clasificación de datos. *Support Vector Machines (SVM)* destaca por su capacidad para encontrar el hiperplano óptimo que separa las clases en un espacio dimensional superior, maximizando el margen entre ellas [31]. Por otro lado, *Gradient Boosting Machines (GBM)* construye una serie de modelos predictivos secuenciales, cada uno corrigiendo los errores del anterior, con el objetivo de mejorar la precisión de las predicciones [32]. Otro algoritmo ampliamente utilizado es *Random Forest*, el cual emplea la técnica de combinación mediante la construcción de múltiples árboles de decisión. Cada árbol se entrena en un subconjunto aleatorio de datos y características, lo que contribuye a mitigar el sobreajuste y mejorar la capacidad de generalización del modelo [33]. Este último algoritmo será utilizado en este trabajo, por lo que a continuación se detallarán más aspectos relevantes sobre su funcionamiento y aplicaciones.

2.3.2.1. Random Forest

El *Random Forest*, también conocido como bosque aleatorio, es un ejemplo de aprendizaje supervisado que se emplea tanto en problemas de clasificación como de regresión. Este método de aprendizaje automático se centra en mejorar la precisión al combinar múltiples modelos

para abordar un mismo problema. Al integrar varios clasificadores, el *Random Forest* reduce la varianza, especialmente en situaciones donde los clasificadores individuales podrían ser inestables, lo que resulta en predicciones más confiables. Una técnica comúnmente utilizada es la votación por mayoría, donde cada muestra sin etiquetar recibe la etiqueta predicha por el clasificador que obtuvo la mayoría de votos entre los modelos del bosque [34].

Este método es ampliamente utilizado debido a su simplicidad y efectividad. El algoritmo *Random Forest* emplea bosques aleatorios con estructuras de árboles y variables de división significativamente diversas, lo que fomenta la aparición de distintas instancias de sobre ajuste y valores atípicos entre los diversos modelos de árboles en conjunto. En consecuencia, la votación final de predicción mitiga el problema del sobre ajuste en los casos de clasificación, mientras que el promedio se utiliza como solución para los problemas de regresión [33].

El uso de *Random Forest* en diferentes circunstancias han demostrado que este algoritmo tiene una mejor precisión en cuanto los otros métodos de aprendizaje supervisado, ofreciendo también medidas de importancia de las variables. Donde estás, son importantes cuando los estudios tienen múltiples fuentes de datos, siendo estos casos donde la dimensionalidad de los datos en muy alta [35].

En este estudio, se emplea el algoritmo *Random Forest* para clasificar los datos relacionados con el tiempo total de espera de los vehículos y, en consecuencia, las emisiones de CO_2. El objetivo fue determinar la duración óptima del ciclo de los semáforos para minimizar la contaminación en la zona urbana de Cuenca.

2.3.3. Aprendizaje por Refuerzo

En el Aprendizaje por Refuerzo un agente es el que establece una conexión hacia su escenario a través de un proceso de observación y operación. Esto se ilustra en la Figura 2.2. En cada interacción el agente obtiene la información del estado del escenario en ese momento, el cual se puede escribir como S_t y este realiza una acción A_t como respuesta. Al realizar este proceso, se realiza cambios en el estado del escenario S_t, por lo que se da una comunicación con el agente mediante una *señal de refuerzo* o recompensa R_t. El objetivo que tiene el agente, en este aprendizaje es seleccionar o tomar decisiones que maximicen una medida de recompensa, la cual se irá acumulando a largo plazo. El agente, para lograr esto, aprende a través de prueba y error [24].

Este modelo puede verse formalmente como:

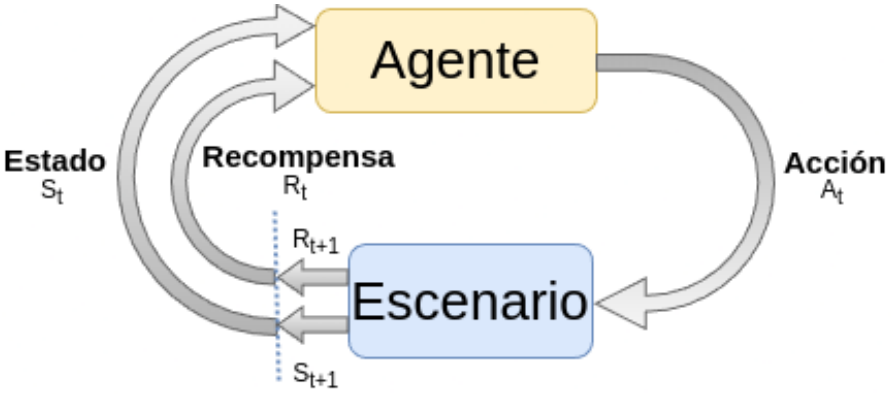


Figura 2.2: Modelo Estándar de Aprendizaje por Refuerzo.

- Un conjunto discreto de estados del escenario *S*.
- Un conjunto discreto de acciones del agente *A*.
- Un conjunto de señales de refuerzo escalares, comúnmente entre valores de [0, 1].

La principal diferencia entre el Aprendizaje por Refuerzo y el Aprendizaje Supervisado es que no existen los pares de entrada/salida. Ya que, después de elegir una acción, el agente es informado por la recompensa inmediata y el estado subsiguiente, pero este no sabe cuál acción habría sido mejor dependiendo del interés a largo plazo. Por lo que es importante que el agente obtenga experiencia útil sobre todas las acciones, procesos y recompensas activamente para actuar de manera óptima [24].

Existen varios tipos de RL, que se pueden clasificar principalmente en: Basado en Valores *(Value-Based RL)* y Basado en Políticas *(Policy-Based RL)*, a continuación analizaremos cada uno de estos.

- **Value-Based RL:** Los algoritmos basados en valores estiman el valor de cada estado o de cada par estado-acción y seleccionan la acción óptima para mejorar el rendimiento del agente. Ejemplos típicos son *Q-learning*: que aproxima el valor Q para cada par estado-acción cada vez y decide qué acción tomar en qué estado. Utiliza políticas ε-greedy para la exploración. Y *Deep Q-Network (DQN)*: el cual utiliza redes neuronales convolucionales para el reconocimiento de imágenes, para eliminar la correlación entre muestras [36].
- **Policy-Based RL:** Un algoritmo basado en políticas optimiza directamente la política que determina el comportamiento del agente. Los algoritmos de gradiente de política producen directamente una política que determina qué acción tomar en cada estado. Ejemplos mas usados pueden ser: *Proximal Policy Optimization (PPO)*, que optimiza la política

limitando la relación de actualizaciones para prevenir la inestabilidad, manteniendo el rendimiento y la eficiencia computacional. Y *Trust Region Policy Optimization (TRPO)*: que agrega una restricción de divergencia KL para evitar cambios rápidos en la política, mejorando la estabilidad del aprendizaje pero requiriendo un cálculo significativo [36].

2.3.3.1. Q Learning

En el aprendizaje por refuerzo *Q Learning (QL)*, los agentes ajustan iterativamente sus estrategias de acción a través de las recompensas obtenidas de la retroalimentación del entorno después de realizar una acción específica. El agente selecciona una acción en un estado particular basándose en el refuerzo recibido o en la experiencia previa, conocida como el valor Q. El refuerzo incluye una recompensa directa y una expectativa futura del valor Q. A través del refuerzo, los agentes pueden evaluar la eficacia de una acción en el estado actual y tomar una mejor acción en el siguiente paso. El objetivo del agente es maximizar las expectativas de las recompensas acumuladas a lo largo de las iteraciones secuenciales. La ventaja del QL radica en que la función de recompensa puede diseñarse utilizando múltiples objetivos ponderados para alcanzar diversas metas. Además, una estrategia adecuada para la exploración y explotación puede ayudar a lograr la solución óptima [37]. En este estudio, se utilizó el método *Epsilon-Greedy* para la exploración y explotación. Este método se explica en detalle en la siguiente subsección.

QL busca determinar una política de acción óptima mediante la estimación de la función óptima de estado-acción $Q'(s, a)$, donde *s* representa un estado del conjunto de posibles estados *S*, y *a* denota una acción del conjunto de posibles acciones *A*. La función *Q* cuantifica la recompensa máxima alcanzable al ejecutar una acción *a* en un estado *s* [38]. La ecuación de QL se expresa de la siguiente manera:

$$Q'(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma max Q'(s',a') - Q(s,a)] \tag{2.1}$$

En donde:

- α representa la tasa de aprendizaje,
- γ representa el factor de descuento, y
- R representa la recompensa obtenida al ejecutar la acción *a* en el estado *s*.

Supongamos que un agente debe aprender a navegar en un entorno de cuadrícula de 4x4

para llegar a una meta (G) desde un estado inicial (S). Cada celda de la cuadrícula representa un estado, y las acciones posibles son moverse hacia arriba, abajo, izquierda o derecha. Si el agente llega a la meta, recibe una recompensa de +1; de lo contrario, la recompensa es 0.

- **Inicialización:** Inicializamos la tabla $Q(s, a)$ arbitrariamente (por simplicidad, se puede inicializar en cero) podemos observar como se inicializa en la Tabla 2.1:

$Q(s_1, a)$	$Q(s_2, a)$	$Q(s_3, a)$	$Q(s_4, a)$
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Tabla 2.1: Tabla Inicializada

- **Iteración:**

  1. **Estado inicial:** $s = (1, 1)$.
  2. **Seleccionar acción** a: Usamos una política ϵ-greedy. Suponemos que seleccionamos mover a la derecha.
  3. **Ejecutar acción y observar recompensa** r: Nos movemos a la derecha, nuevo estado $s' = (1, 2)$, recompensa $r = 0$.
  4. **Actualizar** $Q(s, a)$:

$$Q((1,1), \text{derecha}) \leftarrow Q((1,1), \text{derecha}) + \alpha \left[0 + \gamma \max_{a'} Q((1,2), a') - Q((1,1), \text{derecha}) \right]$$

$$Q((1,1), \text{derecha}) \leftarrow 0 + \alpha \left[0 + \gamma \cdot 0 - 0 \right] = 0$$

  5. **Nuevo estado:** $s = (1, 2)$.

Repetimos este proceso hasta que la tabla $Q(s, a)$ converja a los valores óptimos.

2.3.3.2. Epsilon-Greedy Exploration

En el método *Epsilon-Greedy*, el agente realiza acciones aleatorias si se cumple una condición específica. El valor de Épsilon determina la probabilidad de que el agente ejecute una

acción aleatoria. Cuando el agente alcanza el estado objetivo por primera vez, recibe una recompensa, lo que ajusta los pesos y conduce al agente a seguir el mismo camino en futuras iteraciones. Esto implica que el agente se estabilizará en esta ruta sin explorar otras alternativas, lo que puede llevar a una solución subóptima y evitar que el modelo alcance el resultado globalmente óptimo. Sin embargo, al introducir un cierto nivel de aleatoriedad, el agente continuará buscando otras soluciones incluso después de haber encontrado una inicial [39].

La política *Epsilon-Greedy* es una de las estrategias de exploración más importantes en el aprendizaje por refuerzo [40]. La política *Epsilon-Greedy* define la probabilidad de selección p, centrando la explotación con una probabilidad $1 - \epsilon$ al utilizar el mejor candidato, mientras realiza la exploración con una probabilidad ϵ. La política *Epsilon-Greedy* equilibra bien la exploración y la explotación mediante el parámetro ϵ y se emplea ampliamente en muchos algoritmos de inteligencia artificial [41].

2.4. Aprendizaje Federado

El aprendizaje federado FL es un marco algorítmico de aprendizaje automático que permite a múltiples partes colaborar en la construcción y entrenamiento de un modelo de aprendizaje conjunto, abordando desafíos como la protección de la privacidad, seguridad de datos, aplicabilidad a diversas estructuras e instituciones de datos, rendimiento garantizado e igualdad de estado entre las partes colaboradoras. La construcción de un modelo de aprendizaje federado implica dos procesos: entrenamiento del modelo e inferencia del modelo.

Durante el entrenamiento, las partes pueden intercambiar parámetros que alimentan al modelo sin compartir los datos de identificación de los clientes, preservando la privacidad. Una vez que el entrenamiento colaborativo concluye, cada parte puede realizar inferencias locales en sus propios datos utilizando el modelo global compartido. Esto permite que cada participante realice predicciones o tome decisiones basadas en el conocimiento adquirido durante el entrenamiento conjunto. El mecanismo de distribución justa de valores asegura la sostenibilidad de la federación.

La metodología de FL implica el uso de algoritmos de agregación, como el algoritmo de promedio federado (FedAvg), y técnicas de preservación de la privacidad, asegurando la confidencialidad de los gradientes y pesos del modelo durante la comunicación entre las partes colaboradoras [42]. En la Figura 2.3, se presenta un ejemplo de arquitectura de aprendizaje federado en un modelo cliente-servidor. Basado en la arquitectura antes mencionada, el proceso

puede describirse en los siguientes pasos:

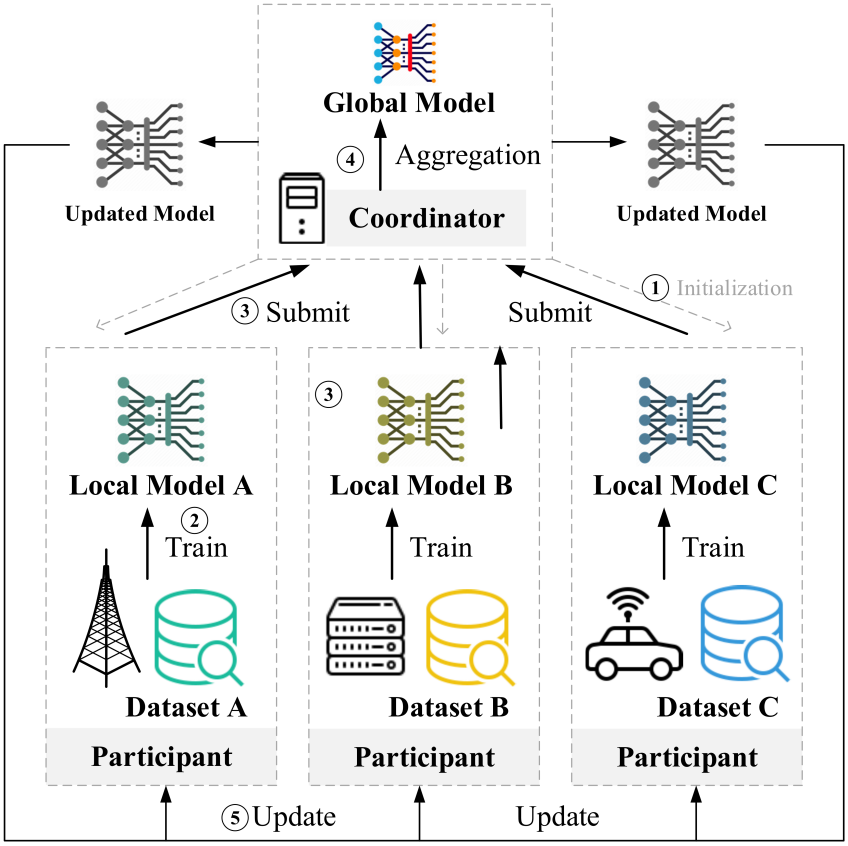


Figura 2.3: Arquitectura de Aprendizaje Federado: Modelo Cliente - Servidor. Fuente: [42]

1. **Inicialización**: El coordinador o servidor crea un modelo inicial, llamado *Global Model*. Este modelo inicial es enviado a todos los clientes involucrados en el proceso de aprendizaje federado.

2. **Entrenamiento Local**: Cada cliente recibe el modelo global y lo entrena utilizando su propio conjunto de datos local. En la Figura 2.3, estos modelos se denominan *Local Model A*, *Local Model B*, y *Local Model C*, correspondientes a los clientes con los conjuntos de datos A, B y C respectivamente. Durante esta fase de entrenamiento, cada participante ajusta el modelo localmente según sus datos específicos, sin compartir los datos locales con otros participantes o con el coordinador.

3. **Envío de Actualizaciones**: Una vez que los modelos locales han sido entrenados, los clientes envían las actualizaciones de los parámetros del modelo local al coordinador o servidor central. Este paso está indicado en la Figura 2.3 con las flechas y la etiqueta *Submit*.

4. **Agregación**: El coordinador central o servidor recibe todas las actualizaciones de los modelos locales y procede a combinarlas utilizando algoritmos específicos de agregación.

Este proceso es representado en la Figura 2.3 con la etiqueta *Aggregation*. La agregación puede realizarse mediante varios métodos, siendo uno de los más comunes el promedio ponderado de los parámetros del modelo.

5. **Actualización del Modelo Global**: Después de la agregación, el coordinador o servidor genera un modelo global actualizado. Este modelo actualizado se envía de nuevo a todos los clientes, cerrando así el ciclo de retroalimentación. Los clientes reciben el modelo global actualizado y pueden iniciar un nuevo ciclo de entrenamiento local con este modelo.

Este proceso se repite iterativamente, permitiendo que el modelo global mejore progresivamente a medida que se incorporan las actualizaciones basadas en los datos locales de los participantes.

Las posibles implementaciones del aprendizaje federado abarcan diversas áreas como la medicina, la banca, y las telecomunicaciones, donde la privacidad de los datos es crítica. Una plataforma popular para implementar sistemas de aprendizaje federado es *Flower (Federated Learning Framework)*, debido a su flexibilidad y facilidad de uso. Flower permite la implementación de algoritmos personalizados y es compatible con diversas bibliotecas de *machine learning* como TensorFlow y PyTorch. Su diseño modular facilita la escalabilidad y la experimentación con diferentes estrategias de agregación y comunicación [43].

2.5. Framework Flower

Flower es un *framework* de FL cuyo objetivo es facilitar el desarrollo de investigaciones experimentales que usan una arquitectura cliente - servidor [43]. La arquitectura de *Flower* se presenta en la Figura 2.4, en donde se observa que la lógica global que incluye la selección de clientes, la configuración, la agregación de actualizaciones de parámetros y la evaluación de modelos federados se realiza a través de una abstracción llamada estrategia.

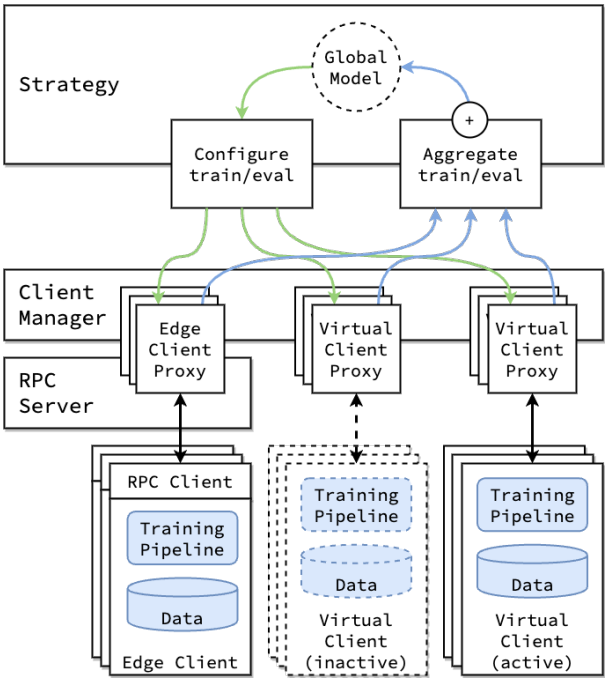


Figura 2.4: Arquitectura de *Flower* con *Edge Client Engine* y *Virtual Client Engine*. Fuente: [43]

La lógica local, por otro lado, se centra en el entrenamiento y evaluación del modelo de ML, el mismo que puede ser federado a nivel del protocolo de *Flower* o utilizando una abstracción de cliente de alto nivel. La federación se basa en un modelo cliente-servidor en donde los principales componentes del servidor son: *Client Manager*, que gestiona la conexión con el servidor. El bucle de federación y la estrategia, en donde se realiza la toma de decisiones. El lado del cliente recibe los mensajes proporcionados por el servidor y basándose en estos realiza el llamado a las funciones de entrenamiento y evaluación. A continuación analizaremos cada una de las secciones mostradas en la Figura 2.4.

- **Estrategia:**
  - ***Global Model***: Representa el modelo central que se actualiza y distribuye a los clientes.
  - ***Configure train/eval***: Este componente se encarga de configurar las tareas de entrenamiento y evaluación que se enviarán a los clientes.
  - ***Aggregate train/eval***: Este componente recoge y agrega los resultados del entrenamiento y evaluación de los clientes para actualizar el modelo global.
- ***Client Manager:***
  - ***Edge Client Proxy***: Actúa como un intermediario entre los clientes de borde y la

estrategia central. Se comunica con los clientes de borde a través del servidor RPC.

- ***Virtual Client Proxy***: Similar al *proxy* de clientes de borde, pero maneja clientes virtuales, que son simulaciones de clientes en lugar de dispositivos físicos. Estos clientes virtuales pueden ser activos o inactivos.

- **RPC *Server***

  - **RPC *Client***: El servidor RPC maneja las comunicaciones entre la estrategia central y los clientes. Cada cliente (tanto de borde como virtual) se comunica con el servidor RPC a través de un cliente RPC.

- **Clientes**

  - ***Edge Client***: Dispositivos físicos que tienen un proceso de entrenamiento y datos locales. Se comunican con el servidor RPC y realizan las tareas de entrenamiento/evaluación configuradas por la estrategia.

  - ***Virtual Client (active/inactive)***: Clientes simulados que tienen un proceso de entrenamiento similar al de los clientes de borde. Los clientes virtuales activos realizan tareas de entrenamiento/evaluación, mientras que los inactivos están listos para activarse cuando sea necesario.

- **Flujo de Datos**

  - **Desde la Estrategia hacia los Clientes**: La estrategia configura las tareas de entrenamiento/evaluación y distribuye estas configuraciones a través del *Client Manager*. Los proxies de clientes reciben estas configuraciones y las envían a los clientes específicos.

  - **Desde los Clientes hacia la Estrategia**: Los clientes ejecutan las tareas y envían los resultados de vuelta a través de los proxies de clientes. La estrategia central agrega estos resultados para actualizar el modelo global.

- **Conexiones y Comunicación**

  - Las líneas verdes representan la configuración de entrenamiento/evaluación que fluye desde la estrategia a los clientes.

  - Las líneas azules representan los resultados de entrenamiento/evaluación que regresan a la estrategia para la agregación.

2.6. Differential Privacy

Differential Privacy (DP) es una técnica de preservación de la privacidad que proporciona garantías matemáticas de que la inclusión o exclusión de un solo dato en un conjunto no afectará significativamente el resultado del análisis, protegiendo así la privacidad individual de los datos sensibles. DP se logra introduciendo ruido aleatorio en los resultados de consultas a bases de datos o en los parámetros de modelos de aprendizaje automático, dificultando que un intruso pueda inferir información específica sobre individuos. La implementación de DP en sistemas de aprendizaje automático distribuidos, como FL, es crucial para mantener la privacidad de los datos [44].

La aplicación de DP en el aprendizaje automático ofrece varios beneficios, como la protección de datos sensibles y la robustez frente a la variabilidad de los datos de los usuarios. A pesar de la adición de ruido, los enfoques de DP mantienen una buena precisión del modelo y propiedades de convergencia. Sin embargo, la implementación práctica de DP enfrenta desafíos, como el equilibrio entre la precisión del modelo y los niveles de privacidad, así como la gestión de la heterogeneidad de los datos de los usuarios [45].

El ruido Gausiano aplicado por DP se describe por medio de la Ecuación 2.2.

$$\frac{\Delta \times \sqrt{2 \times log\left(\frac{1.25}{\delta}\right)}}{\epsilon} \tag{2.2}$$

En donde Δ corresponde a la sensibilidad del modelo. Como la precisión se evaluará en base a los datos sin DP la sensibilidad se establece en 1. Esto debido a que es la máxima diferencia que puede existir entre los valores. δ corresponde a la probabilidad de fallo, es decir la probabilidad de que el mecanismo de privacidad no proporcione el nivel deseado de privacidad, se establece en 1×10^{-5}. ϵ se define como el nivel de protección de privacidad.

2.7. Simulation of Urban MObility

SUMO es una suite de simulación de tráfico de código abierto desarrollada por el Centro Aeroespacial Alemán (DLR). Creado en 2001, SUMO se ha convertido en una herramienta integral para modelar el tráfico en entornos urbanos y ha encontrado aplicaciones significativas en la investigación de comunicación vehicular, específicamente en *Vehicle-to-Vehicle (V2V)* y *Vehicle-to-Infrastructure (V2I)*.

La suite SUMO permite la representación y simulación detallada de redes viales, demanda de tráfico y comportamiento de vehículos en un entorno urbano. Ofrece utilidades para la generación de demanda, enrutamiento de vehículos y una interfaz de control remoto TraCI que permite la adaptación en tiempo real de la simulación. SUMO se destaca por su capacidad para modelar el flujo de tráfico convencional, integrándose con simuladores de comunicación externos, como ns2 y ns3.

La suite SUMO se ha vuelto importante en la investigación de sistemas de transporte inteligentes y comunicación vehicular, brindando a los investigadores la capacidad de simular y evaluar eficazmente el impacto de diversas estrategias de movilidad y tecnologías de comunicación en entornos urbanos [9].

2.8. Trabajos Relacionados

En lo que respecta al conocimiento sobre la privacidad en los ITS, el artículo [1] plantea una perspectiva de privacidad que considera los datos como información que debe mantenerse en secreto frente a otros usuarios, en particular, cuando se trata de datos de trayectoria y ubicación. Se aborda el riesgo de ataques de reidentificación mediante el uso de información del historial de trayectorias y proponen un modelo llamado modelo de privacidad de k-correlación para resolver el ataque de preferencia de movimiento. Además, se presenta un algoritmo llamado TRAMP (Anonimato de Trayectoria contra Preferencia de Movimiento) para abordar el problema. Los resultados obtenidos muestran que el modelo de privacidad de k-correlación y el algoritmo TRAMP reducen significativamente la correlación entre las ubicaciones de estacionamiento y los usuarios, disminuyendo así el riesgo de reidentificación. Esto se logra mediante la anonimización efectiva de los datos de ubicación, lo que proporciona una mayor protección de la privacidad de los usuarios en los sistemas de transporte inteligentes.

Por otro lado, en [13] se destaca la elección del RL debido a la capacidad inherente de adaptarse a entornos variables, lo que lo hace idóneo para situaciones cambiantes y dinámicas. Por otro lado, los enfoques supervisados y no supervisados son más adecuados para contextos estables y predecibles. Se destaca que el aprendizaje por refuerzo permite que los vehículos apliquen el modelo de manera local, posibilitando la anticipación y predicción de los patrones de comportamiento del sistema en tiempo real. Este enfoque descentralizado tiene como ventaja el adaptarse a las particularidades del entorno sin depender de un control centralizado, lo que facilita la toma de decisiones garantizando un funcionamiento óptimo en diferentes

escenarios y condiciones cambiantes.

Así también, en [46] se emplea él FL con el fin de proteger la información personal recopilada mientras se conduce un vehículo autónomo. Se plantea un método de participación utilizando FL que aprovecha dispositivos interconectados, permitiendo así preservar la información personal. Con esta estrategia se busca asegurar la eficiencia en el funcionamiento de vehículos autónomos a través de la protección de datos en un entorno de aprendizaje distribuido. Para evaluar el trabajo, se emplearon métricas de precisión del modelo y eficiencia en el procesamiento de datos. Los resultados demostraron que el uso del aprendizaje federado mejoró la precisión de los modelos locales de inteligencia artificial y la generalización del modelo global, al mismo tiempo que se minimizó la exposición de datos sensibles, lo que garantiza una mayor protección de la privacidad.

En [47], se presenta un sistema de FL basado en *Blockchain* (BFL), en donde se resalta la importancia de mantener privados los datos durante el intercambio y la compartición de conocimientos entre vehículos en una red. Para lograrlo, se proponen dos algoritmos denominados como algoritmos oVML y Miner. La comparación se lleva a cabo con el algoritmo de FL de Google, utilizando variables críticas como el retraso medio del sistema, las probabilidades de bifurcación y pérdida, así como el impacto de los errores en el sistema. Todo esto se evalúa considerando diferentes números de vehículos en la red y diversas condiciones de los canales. Los resultados obtenidos muestran que el retraso del sistema disminuye con el aumento de la relación señal-ruido (SNR), mientras que las probabilidades de descarte de bloques y de bifurcación aumentan exponencialmente con los fallos de transmisión. Los errores de canal afectan más a la probabilidad de bifurcación que a la de descarte, incrementando significativamente el retraso total del sistema BFL. Con baja probabilidad de fallos de transmisión, las retransmisiones compensan las pérdidas; sin embargo, con errores significativos, las retransmisiones no son suficientes, aumentando la pérdida de bloques y la probabilidad de bifurcación.

En [48], se busca implementar el FL de manera más eficiente mediante el desarrollo de un blockchain híbrido, denominado PermiDAG, que mejora un modelo de FL mediante una selección específica de nodos. PermiDAG se compone de un blockchain principal con permisos, mantenido por las RSUs, y un Grafo Acíclico Dirigido (DAG) local ejecutado por los vehículos para facilitar el intercambio eficiente de datos en el IoV. Se propone un esquema asíncrono de FL para aprender modelos a partir de datos en el borde, mejorando la eficiencia del FL al seleccionar los nodos participantes con el fin de minimizar el costo total. Esto permite abordar las capacidades heterogéneas de comunicación y cómputo de los vehículos. Se destaca que

la forma segura de compartir datos es un área de investigación abierta.

De igual manera, en [49] involucra la comunicación entre el FL basado en cifrado homomórfico y el FL basado en computación multipartida; sin embargo, estas comunicaciones conllevan una carga computacional significativa y requieren más tiempo de procesamiento. Así mismo, exploramos el paradigma de aprendizaje colaborativo, conocido como FL, con el propósito de salvaguardar la utilidad de los datos y, por ende, mantener la exactitud de los modelos de aprendizaje automático. Este enfoque se orienta a garantizar la privacidad y confidencialidad de los usuarios involucrados en un ITS.

En cuanto a la implementación de técnicas de ML para abordar problemas de seguridad en los ITS. En [50] se investiga cómo el uso de técnicas de *Deep Learning (DL)* puede mejorar la seguridad y protección de las personas dentro de los ITS. Este artículo usa la precisión y la tasa de detección como métricas para evaluar la eficacia de las soluciones propuestas, destacando la importancia de estas técnicas en la mitigación de riesgos de seguridad.

En [51], centrado en la predicción del tráfico mediante técnicas de ML, se aborda las vulnerabilidades que pueden surgir con la implementación de estas técnicas en ITS. Se usa la precisión de las predicciones, y la eficacia operativa del transporte para evaluar todo el sistema.

En el contexto de las ciudades inteligentes, en [52] se discute la aplicación de DL para la detección de ataques DDoS en sistemas basados en *blockchain*. Este estudio demuestra cómo estas técnicas pueden reducir los problemas de seguridad en los ITS, enfocándose en la tasa de detección de ataques como métrica principal. Los resultados experimentales, obtenidos a partir de la evaluación de tres conjuntos de datos diferentes, muestran que el enfoque propuesto es efectivo para detectar y clasificar distintos tipos de ataques DDoS, alcanzando tasas de F1-score superiores al 95 % en promedio. Esto indica que la combinación de *blockchain* y DL proporciona una protección robusta y confiable contra diversos ciberataques en el sistema de transporte inteligente.

En [53] se investiga específicamente cómo las técnicas de DL pueden resolver problemas de seguridad en los ITS. La precisión y la robustez de los sistemas implementados son usadas como métricas para evaluar las soluciones propuestas.

El papel de la ciberseguridad en los ITS es otro tema estudiado en la literatura. En [54] se discute cómo el ML presenta vulnerabilidades en la fase de entrenamiento que pueden ser explotadas. Este estudio enfatiza la necesidad de robustez y resiliencia en los ITS para mitigar estas vulnerabilidades, destaca la importancia de un enfoque integral de seguridad.

Un informe publicado por el Departamento de Transporte de los Estados Unidos [55] examina los desafíos y las lecciones aprendidas en la implementación de IA y ML en los ITS. El informe aborda cómo estas tecnologías pueden hacer que los sistemas de transporte sean más seguros, equitativos y confiables, destacando los problemas de seguridad y privacidad como principales preocupaciones.

La aplicación de IA en la conducción autónoma también ha sido objeto de investigación. En [56] se analiza diversas aplicaciones habilitadas por IA, utilizando técnicas de DL para mejorar la precisión en la conducción. Se destacan los estándares y amenazas de seguridad que deben ser considerados, proporcionando una evaluación integral de las métricas de control, por ejemplo, la tasa de detección de errores, la precisión en la predicción de rutas, el tiempo de respuesta del sistema, y la tasa de éxito en la identificación de obstáculos, necesarias para mantener la seguridad.

En [57] se discute los desafíos en la seguridad de datos y privacidad en los ITS. A medida que estos sistemas se vuelven esenciales para la comunicación vehicular, el estudio destaca los problemas de seguridad y privacidad como desafíos críticos que deben ser abordados para asegurar la eficiencia y seguridad.

Finalmente, en la Tabla 2.2 se presenta un resumen de los artículos analizados que emplean técnicas deML en los ITS. La tabla detalla el tipo de ML utilizado, las métricas de control aplicadas y los problemas de seguridad abordados en cada una de las técnicas analizadas. En el siguiente capítulo, se presenta la metodología empleada en este estudio para diseñar y evaluar una arquitectura basada en técnicas de aprendizaje automático que mejoren la privacidad en los ITS. Este capítulo incluye el planteamiento del escenario y detalla la implementación de las diversas técnicas de aprendizaje utilizadas.

Referencia	Técnica de Machine Learning Usada	ITS	Métricas de Control	Problemas de Seguridad
[50]	Deep Learning	ITS General	Precisión, Tasa de Detección	Seguridad y Seguridad de las Personas
[51]	Predicción de Tráfico	ITS General	Precisión de Predicción	Vulnerabilidades en el Sistema
[52]	Deep Learning	ITS en Smart City	Tasa de Detección de DDoS	Problemas de Seguridad en Sistemas Futuros
[53]	Deep Learning	ITS General	Precisión, Robustez	Problemas de Seguridad en ITS
[54]	Machine Learning General	ITS General	Robustez, Resiliencia	Vulnerabilidad en Fase de Entrenamiento
[55]	Machine Learning General	ITS General	Seguridad, Equidad, Confiabilidad	Problemas de Seguridad y Privacidad
[56]	Deep Learning	Conducción Autónoma	Precisión en Conducción	Estándares y Amenazas de Seguridad
[57]	No especificado	ITS General	Seguridad de Datos, Privacidad	Desafíos en Seguridad de Datos y Privacidad

Tabla 2.2: Artículos Analizados con Técnicas de Machine Learning en ITS

3. Metodología

En este capítulo se describe la metodología utilizada para diseñar la arquitectura basada en técnicas de aprendizaje automático para mejorar la privacidad en los ITS. La metodología se divide en varias secciones que detallan desde el planteamiento del escenario hasta la implementación de diferentes técnicas de aprendizaje. Cada una de estas secciones se puede apreciar en la Figura 3.1. A continuación, se presenta el planteamiento del escenario, que proporciona el contexto y las condiciones iniciales necesarias para la aplicación de las técnicas de aprendizaje.

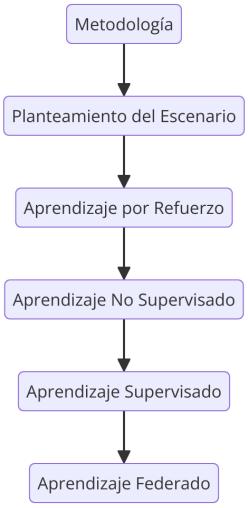


Figura 3.1: Diagrama con las Secciones de Metodología

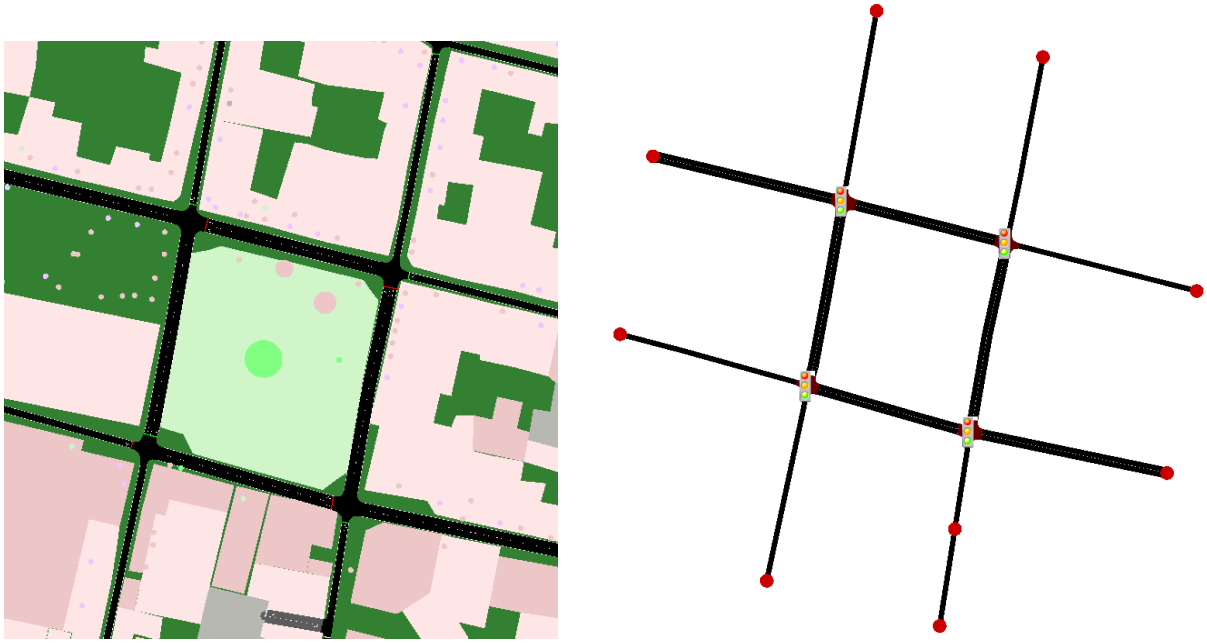
3.1. Planteamiento del Escenario

En este capítulo se describe el escenario en el que se aplicarán diversos métodos de aprendizaje automático. Estos métodos incluyen el Aprendizaje por Refuerzo, el Aprendizaje Supervisado, el Aprendizaje No Supervisado y el Aprendizaje Federado.

Inicialmente, se crea un mapa detallado del centro urbano de Cuenca utilizando la plataforma SUMO. Específicamente, se emplea el *script osmWebWizard.py* [58], que permite obtener cualquier parte del mundo para su simulación utilizando el software *Open Street Maps*; en este caso, se ha seleccionado el centro de la ciudad de Cuenca.

Este mapa generado incorpora el parque central de la ciudad, conocido por su disposición de cuatro esquinas, cuatro arterias principales, así como cuatro intersecciones y semáforos estratégicamente ubicados. Este entorno cartográfico se presenta de manera visual en las Figuras

3.2, donde la Figura 3.2(a) ofrece una visión detallada de las edificaciones y el parque central, mientras que la Figura 3.2(b) se enfoca en las calles y los semáforos en cada intersección, aspecto central de nuestro estudio.



(a) Mapa completo del Centro de Cuenca (b) Mapa vial del Centro de Cuenca

Figura 3.2: Mapa Utilizado como Escenario

Una vez generado el mapa, procederemos a modificar el archivo *.rou.xml*, que contiene las especificaciones de las rutas y flujos de cada vehículo. Este archivo permite ajustar diversos parámetros de la simulación, tales como las rutas, la velocidad de los vehículos, el carril a utilizar, la posición de partida y, lo más relevante en nuestro caso, la probabilidad de generación de nuevos vehículos en la simulación. En esta configuración específica, se ha optado por una probabilidad de generación del 15 %. Se ha seleccionado una probabilidad de generación de nuevos vehículos del 15 % para mantener un equilibrio adecuado entre el tráfico predefinido y el aleatorio. Este valor permite simular una cantidad realista de variabilidad en el tráfico, representando de manera efectiva las fluctuaciones que ocurren en entornos urbanos sin introducir un nivel de aleatoriedad que podría complicar la interpretación de los resultados. Así, se asegura que la simulación refleje tanto las rutas planificadas como las variaciones espontáneas en el flujo vehicular.

De igual manera, se procedió a modificar el archivo *.sumocfg* con el objetivo de obtener una nueva salida correspondiente a las emisiones generadas durante la simulación. Esta información es fundamental para nuestro análisis en los diferentes tipos de aprendizaje. Nos centramos principalmente en las emisiones de CO_2 y en el tiempo de espera de los vehículos en cada

una de las intersecciones de la simulación. Estos datos se utilizaron para evaluar y mejorar la eficiencia del tráfico urbano.

Cabe mencionar que también se consideró el archivo *.net.xml*, el cual define la duración de cada ciclo de los semáforos. Este valor es importante al momento de obtener los datos que se utilizaron para los aprendizajes supervisados y no supervisados. Para estos casos, se realizaron múltiples simulaciones, comenzando con ciclos de 10 segundos y aumentando progresivamente hasta los 30 segundos. De esta manera, se obtuvieron los datos necesarios para llevar a cabo los análisis de aprendizaje previamente mencionados.

Se propone que cada simulación tenga una duración de un día completo, lo que equivale a 86400 segundos. Todas estas configuraciones se pueden observar en la Tabla 3.1.

Parámetro	Valor
Tamaño Mapa	300m x 300m
Número de Calles	12
Número de Intersecciones	4
Número de Esquinas	4
Número Traffic Light System (TLS)	4
Densidad Vehicular	0.15 o 15 %
Duración de la Simulación	86400

Tabla 3.1: Valores Utilizados para las Simulaciones

3.1.1. Sistema de Control de Emisiones Propuesto

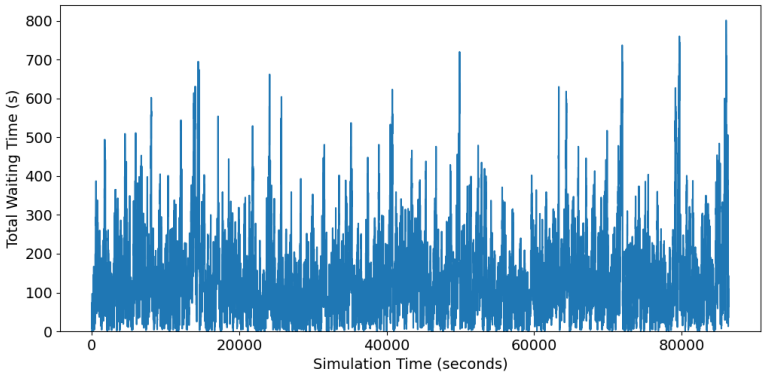
La obtención de la duración de cada semáforo se realizó a partir del archivo *.net.xml*, el cual detalla la duración del ciclo semafórico. Para el centro de la ciudad de Cuenca, se identificaron dos ciclos. El primero tiene una duración de 42 segundos para las luces roja y verde, y de 3 segundos para la luz amarilla. El segundo ciclo tiene una duración de 15 segundos para cada luz. Estos datos fueron extraídos durante la configuración del escenario para la simulación utilizando el software *OpenStreetMap*.

Como se mencionó anteriormente, el enfoque se centra en las emisiones de CO_2 y en el tiempo de espera de los vehículos frente a un semáforo. En la Figura 3.3, se muestra las características del sistema inicial propuesto. Mientras que, en las Figuras 3.3(a), 3.3(b) y 3.3(c), se presenta el tiempo total de espera con la duración de los semáforos por defecto en segundos, las emisiones de CO_2 y el número de vehículos con respecto a la duración de la simulación, que como se mencionó anteriormente, es de 86400 segundos.

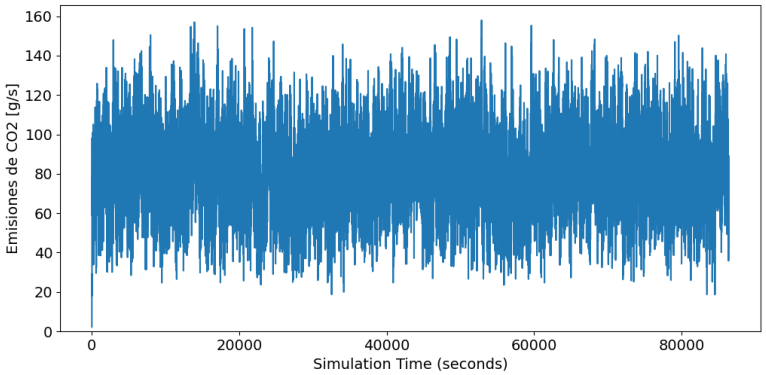
Este tiempo total de espera, o *Total waiting time*, representa la suma de todos los periodos de espera de los vehículos frente a los semáforos durante la simulación. En este análisis, se considera el tiempo total de espera de los cuatro semáforos. La Figura 3.3(a) muestra la variación del tiempo total de espera de los vehículos en las intersecciones con semáforos a lo largo de la simulación. Esta figura sugiere una variabilidad significativa en los tiempos de espera, lo cual puede atribuirse a los patrones de tráfico y a la configuración por defecto de los semáforos, afectando la densidad vehicular en las simulaciones. Los picos en esta simulación alcanzan hasta los 800 segundos.

De igual manera se puede observar la Figura 3.3(b), en donde se muestra el nivel de emisiones de CO_2 generado a lo largo de la simulación, con valores que varían considerablemente a lo largo del tiempo. Existen fluctuaciones significativas de emisiones de CO_2, con picos que alcanzan las 160 g/s.

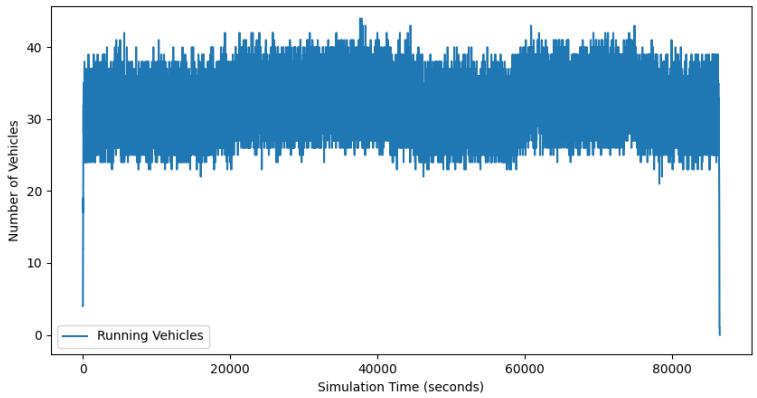
Así también en la Figura 3.3(c), se muestra el número de vehículos presentes en la red a lo largo del tiempo de simulación. La tendencia de vehículos es alrededor de los 30 y 40 vehículos. El número de vehículos es relativamente constante con ligeras fluctuaciones, excepto al inicio y al final de la simulación.



(a) Tiempo de Espera



(b) Emisiones de CO_2



(c) Número de Vehículos en la Simulación

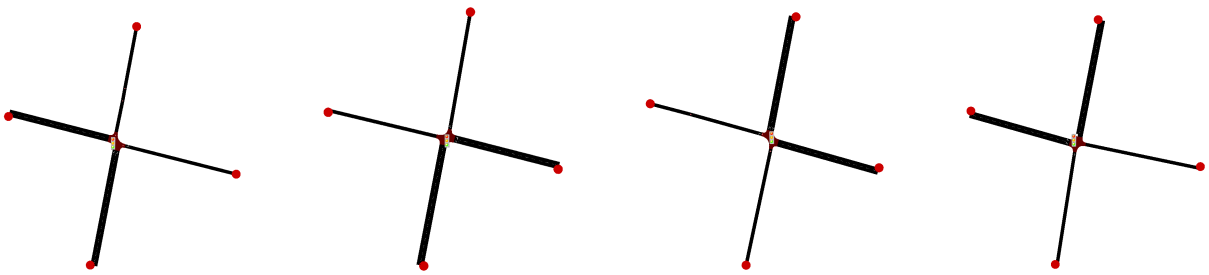
Figura 3.3: Características del Sistema Inicial

Se puede notar que la relación entre estas figuras radica en que los picos en el tiempo de espera (Figura 3.3(a)) tienden a coincidir con aumentos en las emisiones de CO_2 (Figura 3.3(b)), ya que los vehículos detenidos o en marcha lenta emiten más contaminantes. Asimismo, un mayor número de vehículos en la red (Figura 3.3(c)) puede contribuir a mayores tiempos de

espera y, en consecuencia, a mayores emisiones de CO_2. Estas interacciones muestran cómo la cantidad de vehículos, los tiempos de espera y las emisiones se afectan mutuamente, reflejando la complejidad del tráfico urbano y su impacto ambiental. Estos datos servirán como punto de partida para realizar las comparaciones con otros métodos de aprendizaje, como el aprendizaje por refuerzo y el aprendizaje federado.

3.1.2. Escenarios Propuestos para Aprendizaje Federado

Para llevar a cabo las simulaciones del Aprendizaje Federado, se diseñó un escenario que consta de cuatro sub escenarios, es decir, cuatro clientes representados por diferentes mapas, que a su vez estarán conectados a un único servidor. Los cuatro sub escenarios se muestran en la Figura 3.4. Cada uno de los mapas corresponde a una esquina del mapa utilizado en RL. Es decir, los semáforos ubicados en la esquina del Parque Calderón actuarán como clientes de un esquema de federación en donde cada cliente entrenará su modelo de RL en modo agente único.



(a) Cliente Semáforo 0 (b) Cliente Semáforo 1 (c) Cliente Semáforo 2 (d) Cliente Semáforo 3

Figura 3.4: Escenarios Utilizados para el Aprendizaje Federado

Todos los escenarios propuestos presentan el mismo flujo de vehículos, definido por un valor de probabilidad. Un vehículo será agregado al mapa aleatoriamente con la probabilidad configurada en cada segundo hasta que se alcance el tiempo de finalización. El número de vehículos insertados sigue una distribución binomial. Por lo tanto, se considerarán estas mismas condiciones para cada uno de los clientes en el estudio.

3.2. Aprendizaje por Refuerzo (Q-Learning)

Para la implementación QL se hizo uso de la librería SUMO-RL, la cual permite el desarrollo de algoritmos de RL aplicados al control de señales de tráfico. Cuenta con un *enviroment* de

Gymmnasium configurado para el despliegue de simulaciones en SUMO por medio de TraCI [59].

3.2.1. Estados

La definición del espacio de estados de la Ecuación 3.1 deriva de la consideración de un semáforo como agente que controla una intersección. Cada uno de los pasos de la simulación se relaciona con un tiempo t, por lo tanto, en cada paso se produce un vector s_t que representa el estado de la intersección. Para la definición del estado se considera además la fase actual ρ, el tiempo transcurrido de la fase actual δ, la densidad que se define como el número de vehículos sobre la capacidad de vehículos de cada fase, y el número de vehículos detenidos. Considerando como vehículos detenidos a los que tienen una velocidad inferior a 0.1 m/s.

$$s_t = [\rho, \delta, lane_1\_density, ..., lane_n\_density, lane_1\_queue, ..., lane_n\_queue] \tag{3.1}$$

3.2.2. Acciones

Se cuenta con dos acciones: un agente puede mantener el tiempo en verde del paso previo para la fase actual o actualizar la duración de la fase. Estas acciones son definidas como mantener y cambiar, respectivamente. La restricción de tiempos se realiza asignando un valor máximo y mínimo a la duración de la fase δ. Entre el cambio de fases se considera una fase intermedia amarilla con una duración constante de 2 s.

3.2.3. Función de Recompensa

La recompensa asignada se presenta en la Ecuación 3.2, esta recompensa se define con la variación entre el tiempo de espera acumulado en cada intersección entre las diferentes acciones.

$$r_t = W_t - W_{t+1} \tag{3.2}$$

Es decir, W_t y W_{t+1} representan el tiempo de espera acumulado en la intersección antes y después de la acción a_t. El tiempo de espera acumulado W_t se define como:

$$W_t = \sum_{v \in V_t} w_{v,t} \tag{3.3}$$

donde V_t es el conjunto de vehículos que llegan a una intersección en el paso de tiempo t, y $w_{v,t}$ es el tiempo total de espera del vehículo v desde que ingresó en una de las carreteras que desembocan en la intersección hasta el paso de tiempo t. Cuando el tiempo de espera de W_{t+1} es mayor a W_t se establece una recompensa negativa.

3.2.4. Agentes

Un agente en sumo-rl es un semáforo en una intersección. El agente se define en una clase que sigue el diagrama de flujo de la Figura 3.5. En este diagrama, el proceso comienza con la inicialización del agente. Posteriormente, explora la estrategia de aprendizaje con el método *epsilon-greedy*. Basado en la tabla Q el agente elige una acción, y después de su ejecución pasa al proceso de aprendizaje. Durante este proceso, el agente utiliza la recompensa y la nueva información del estado para actualizar la tabla Q, ajustando las estimaciones de los valores Q para mejorar la política de selección de acciones futuras. El estado del agente se actualiza basado en el comportamiento del entorno. Finalmente, se establece un valor de recompensa acumulada.

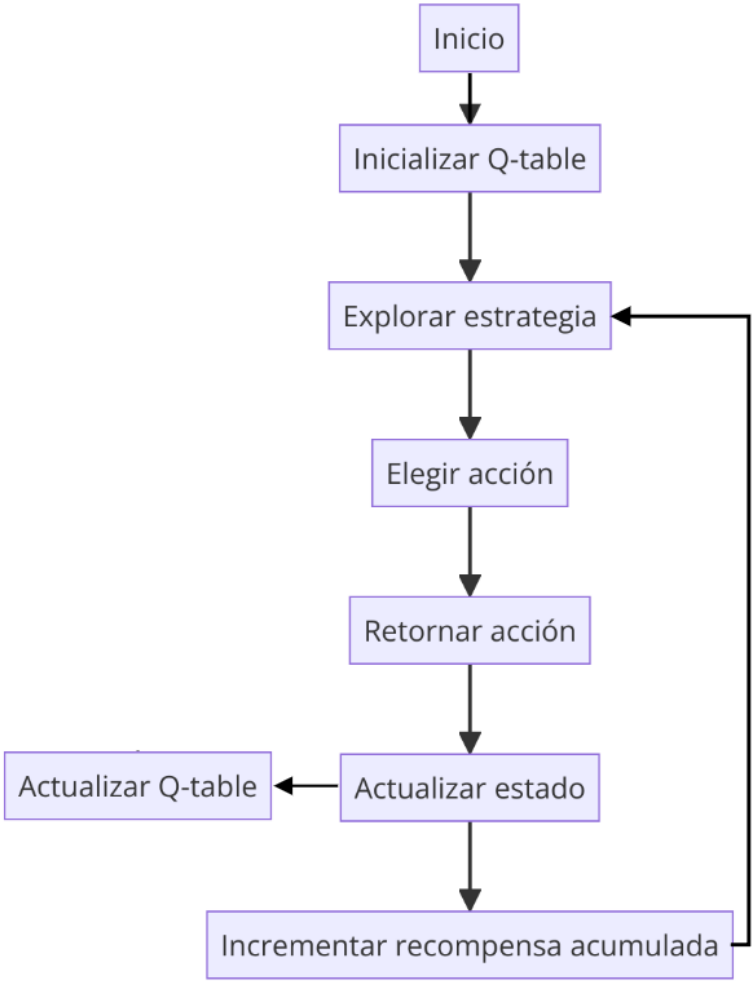


Figura 3.5: Agente Sumo-rl

Sumo-rl define también una clase llamada *TrafficSignal*, que se encarga de controlar a los semáforos utilizando la *Application Programming Interface (API)* de TraCI y cuyo diagrama se presenta en la Figura 3.6. Esta clase define la inicialización de los parámetros de los semáforos, las próximas fases de los semáforos basado en el cálculo de observaciones y recompensas y recupera la información detallada del tráfico, como el tiempo de espera acumulado, velocidad promedio y densidad de carriles.

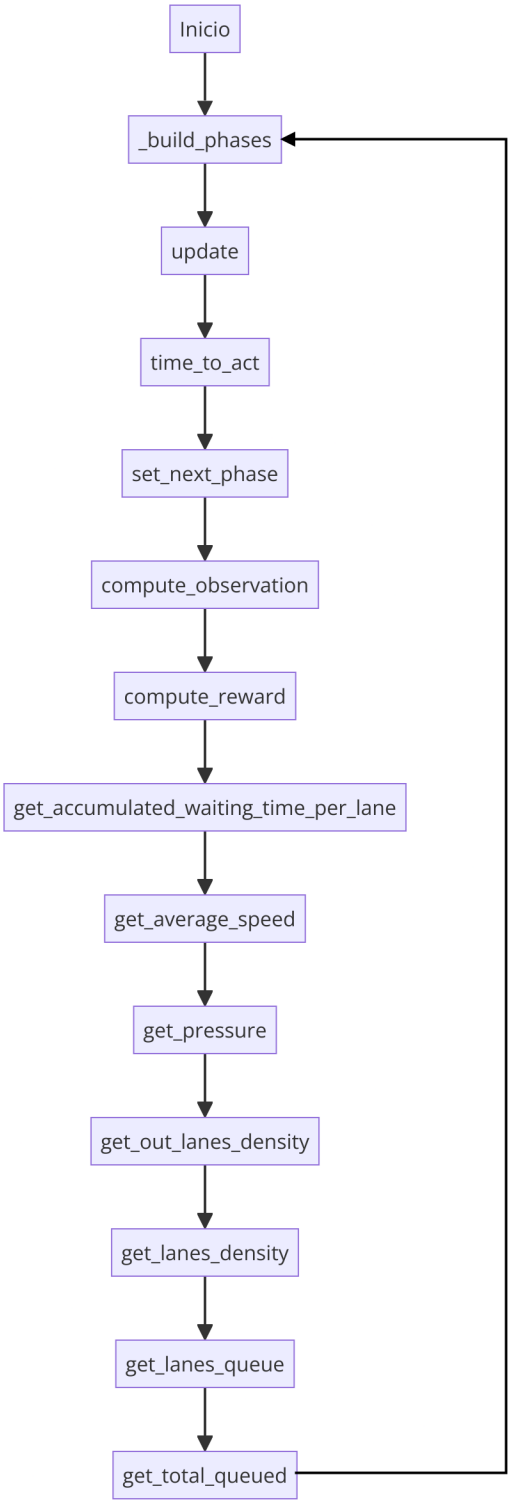


Figura 3.6: Traffic Signal Sumo-rl

3.2.5. Entorno

El entorno de SUMO se configura en *Gymmnasium*. La Figura 3.7 presenta el diagrama de flujo de la clase que define este entorno. El proceso comienza con el reinicio del entorno. Esta clase se encarga de iniciar la simulación y definir lo que sucede en cada paso. Durante cada paso, se aplican las acciones y se calculan las observaciones y recompensas. Además, se recopila información adicional y se configura el archivo de salida.

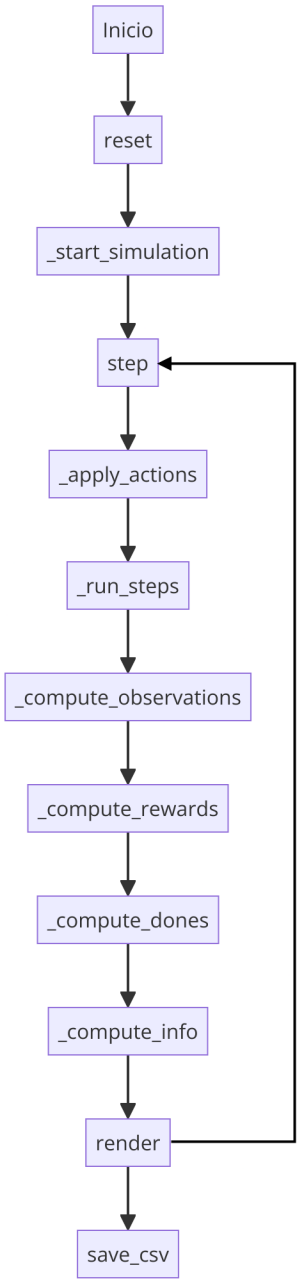


Figura 3.7: Entorno Sumo-rl

3.3. Aprendizaje No Supervisado

Dentro del contexto de Aprendizaje No Supervisado, incluimos el PCA. El objetivo de este enfoque es segmentar los datos de ingreso generados mediante el aprendizaje por refuerzo, los cuales han sido obtenidos simulando una densidad media vehicular con los parámetros mostrados en la Tabla 4.1.

Se ha procurado obtener la mayor cantidad de datos posibles, en este caso, se han recopilado un total de 14000 valores para cada una de las variables que se muestran en la Tabla 3.2. Podemos notar que estas variables corresponden a los valores obtenidos en el sistema (*system*) y los valores correspondientes a cada uno de los semáforos utilizados en la simulación, siendo estos *s1, s2, s3* y *s4*. Estos datos se encuentran almacenados en un archivo *.csv*, un formato idóneo para datos tabulares y ampliamente compatible con diversas herramientas de análisis.

step	system_out_lanes_density_avg	s3_stopped
system_vehicles	system_lanes_density_avg	s3_accumulated_waiting_time
system_total_stopped	system_queue_avg	s3_average_speed
system_total_waiting_time	system_mean_total_queued	s3_average_acceleration
system_VSP	s1_stopped	s4_stopped
system_mean_waiting_time	s1_accumulated_waiting_time	s4_accumulated_waiting_time
system_mean_speed	s1_average_speed	s4_average_speed
system_mean_acceleration	s1_average_acceleration	s4_average_acceleration
system_mean_VSP	s2_stopped	agents_total_stopped
system_total_C02_emissions	s2_accumulated_waiting_time	agents_total_accumulated_waiting_time
system_mean_fuel_consumption	s2_average_speed	total_accumulated_VSP_emissions
system_total_fuel_consumption	s2_average_acceleration	

Tabla 3.2: Variables Utilizadas para el Aprendizaje

Este proceso de reducción de dimensionalidad permitirá simplificar la complejidad de nuestros datos, preservando al mismo tiempo la mayor cantidad posible de su información inherente. La aplicación de PCA posibilita explorar la estructura de los datos de manera más eficiente y, potencialmente, identificar agrupaciones naturales o relaciones significativas entre las variables, lo que resulta fundamental para el análisis y la toma de decisiones informadas. A continuación, se detallarán las etapas seguidas para la implementación de este algoritmo y en la Figura 3.8 podemos observar el diagrama de flujo implementado.

- **Carga y Preparación de Datos:**
  - Los datos se cargan desde un archivo *.csv* utilizando la librería pandas, que permite una gestión eficiente y flexible de los datos.

- Se realiza un preprocesamiento de datos, incluyendo la limpieza de datos y la eliminación de columnas innecesarias, para asegurar que solo las características relevantes sean incluidas en el análisis.

- **Transformación y Normalización de Datos:**
  - Se aplica $\mathrm{StandardScaler}$ para normalizar las características. Este paso es para evitar que las características con mayor varianza dominen el análisis, asegurando que todas las características contribuyan equitativamente al PCA.
- **Análisis de Componentes Principales (PCA):**
  - Se realiza él PCA en los datos normalizados para identificar los componentes principales que explican la mayor parte de la variabilidad en los datos.
  - Se calcula la varianza explicada por cada componente principal y sé gráfica la varianza explicada acumulada para determinar el número óptimo de componentes a retener, facilitando así la reducción dimensional sin pérdida significativa de información.
- **Visualización de Resultados:**
  - Se visualizan los primeros componentes principales mediante gráficos de dispersión y otras técnicas de visualización, permitiendo una comprensión intuitiva de la estructura de los datos en un espacio de menor dimensión.
- **Interpretación de Componentes Principales:**
  - Se analizan los pesos (cargas) de las características en los componentes principales para interpretar la contribución de cada característica a los componentes. Esto ayuda a identificar las características más influyentes en la variabilidad de los datos.
- **Selección de Método de Normalización:**
  - Se compara el resultado del PCA utilizando $\mathrm{StandardScaler}$ y $\mathrm{MinMaxScaler}$ para evaluar cómo la elección del método de normalización afecta los resultados del PCA. Esta comparación proporciona una comprensión más profunda de cómo diferentes técnicas de normalización pueden influir en el análisis de componentes principales.

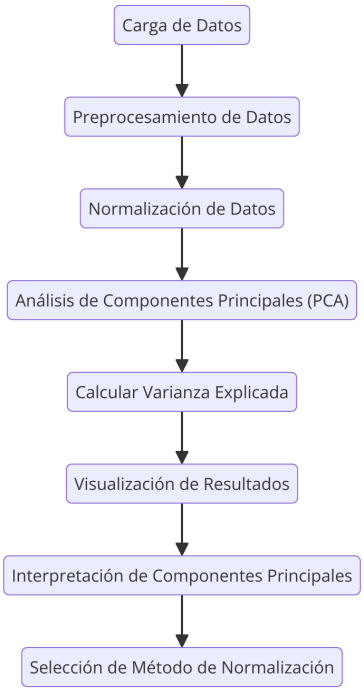


Figura 3.8: Diagrama de Flujo de Aprendizaje No Supervisado

3.4. Aprendizaje Supervisado

Para la implementación de un Aprendizaje Supervisado se usó el algoritmo *Random Forest* para clasificar los datos de tiempo de espera en distintos niveles de congestión de tráfico: *Alto, Moderado y Bajo*. Esto con el fin de encontrar el valor de duración de semáforo en el cual exista menos tiempo de espera de los vehículos y por ende menos emisiones de CO_2. A continuación, se describen los pasos seguidos para la implementación y análisis de los datos obtenidos a partir de diversas simulaciones en las que se ha variado el tiempo de los semáforos. En la Figura 3.9 se observa el diagrama de flujo implementado.

- **Carga y Preparación de Datos:** Los datos utilizados provienen de simulaciones de tráfico en las que se ha variado el tiempo de los semáforos. Los pasos seguidos son los siguientes:
  - **Carga de Datos:** Los datos se cargaron desde un archivo .csv que contiene los resultados de las simulaciones realizadas en SUMO, utilizando la librería pandas.
  - **Renombrado de Columnas:** Para reflejar los diferentes tiempos de semáforo, se renombraron las columnas. Esto facilita la identificación y manejo de los datos en el análisis posterior.

- **Transformación y Normalización de Datos:** Para preparar los datos para el análisis y la modelación, se realizaron los siguientes pasos:
  - **Transformación del Formato de Datos:** Utilizando la función *melt* de pandas, se transformaron los datos a un formato largo. Esto permite un análisis más eficiente al convertir múltiples columnas de tiempos de semáforo en una sola columna.
  - **Normalización de Tiempos de Espera:** Se aplicó la técnica de normalización *StandardScaler* para estandarizar los tiempos de espera. La normalización asegura que los tiempos de espera tengan una escala uniforme.
- **Clasificación:** Para clasificar los tiempos de espera en diferentes niveles de congestión, se siguieron estos pasos:
  - **Cálculo de Umbrales:** Se calcularon los percentiles 33 % y 66 % de los tiempos de espera normalizados para definir los umbrales de clasificación. El percentil 33 % divide los datos de manera que el 33 % de los tiempos de espera son menores o iguales a este valor. Mientras que el percentil 66 % divide los datos de manera que el 66 % de los tiempos de espera son menores o iguales a este valor.
  - **Definición de Categorías de Congestión:** Basado en estos umbrales, los tiempos de espera se clasificaron en tres categorías. Bajo (Flujo de tráfico bajo), tiempos de espera menores al percentil 33 %. Moderado (Flujo de tráfico moderado), tiempos de espera entre el percentil 33 % y el percentil 66 %. Alto (Congestión alta), tiempos de espera mayores al percentil 66 %.
- **Visualización de Resultados:** Para entender mejor la relación entre el tiempo de los semáforos y la congestión del tráfico, se realizaron algunas visualizaciones como lo son: gráficos de dispersión y gráficos de tiempo.
- **Modelado y Evaluación del Modelo:** Para evaluar la capacidad del modelo *Random Forest* en clasificar los datos de tráfico, se siguieron estos pasos:
  - **División de Datos:** Los datos se dividieron en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo.
  - **Entrenamiento del Modelo:** Se entrenó un modelo *Random Forest* utilizando los datos obtenidos de las simulaciones con RL.
  - **Evaluación del Modelo:** Se evaluó el rendimiento del modelo utilizando un reporte de clasificación *(classification_report)*, el cual proporciona métricas detalladas como

precisión, *recall* y *F1-score* para cada categoría de congestión.

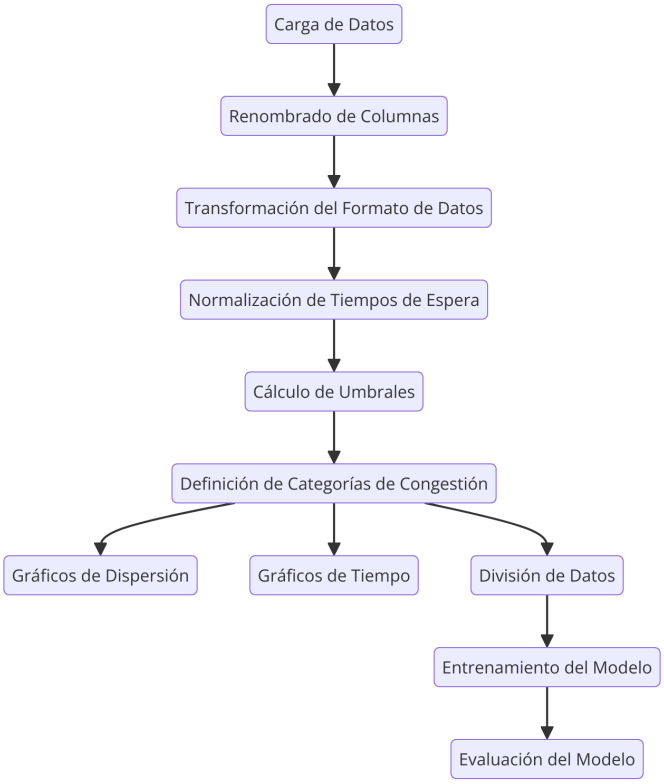


Figura 3.9: Diagrama de Flujo de Aprendizaje Supervisado

3.5. Aprendizaje Federado

Para la implementación en el servidor de *Flower* se selecciona el algoritmo (FedAvg) ampliamente utilizado en el FL, creado por McMahan [60] y presentado en el Algoritmo 1 en el Anexo A.A.

El código propuesto para realizar la federación con *Flower* se describe en el diagrama de flujo presentado en la Figura 3.10. El cual consta de dos partes: el cliente de *Flower* y el servidor *Flower*. En el contexto de la federación cada semáforo es considerado como un cliente, con su respectivo archivo de red y de rutas. A su vez cada cliente es considerado como un agente para el algoritmo de RL. Una ronda es considerada como el intercambio de parámetros entre los clientes y el servidor y consta de una etapa de entrenamiento y otra de evaluación. En cada etapa de entrenamiento se pueden establecer episodios de ejecución del algoritmo de RL, a su vez en cada episodio se establece un determinado número de pasos, que se interpreta además como el tiempo de simulación.

En el cliente de *Flower* se importan las librerías necesarias para el manejo y transformación

de datos, la generación de archivos de salida y la librería sumo-rl usada para el entrenamiento con RL.

Posteriormente, se inicializa el cliente mediante la creación de una instancia de *RLClient*, que incluye la configuración del cliente, y los parámetros necesarios para la ejecución de RL. Se configura el entorno de SUMO especificando el archivo de red y de ruta, los parámetros de simulación y la función de recompensa. Se crean y configuran agentes de *Q-Learning*, incluyendo la estrategia de exploración *EpsilonGreedy*.

Durante la fase de entrenamiento, el entorno de SUMO se reinicia para comenzar el entrenamiento. Los agentes ejecutan acciones en el entorno durante el número de episodios establecidos. En cada episodio se generan recompensas y se actualizando las tablas Q (ejemplo en la subsección 2.3.3.1). Las tablas Q de los agentes se actualizan basadas en las experiencias de entrenamiento, generando un diccionario relacionado con un valor de estado y un valor de acción. Los resultados se guardan en archivos CSV para su posterior uso y análisis.

En la fase de evaluación, se configura el entorno SUMO seleccionando la red que contiene a todos los agentes. Se cargan las tablas Q provenientes del servidor. Tras la ejecución se registran las métricas de recompensa y tiempo de espera.

En el servidor *Flower*, se lleva a cabo la comunicación cliente-servidor. Los clientes envían los parámetros del modelo entrenado al servidor central. El servidor recibe estos parámetros y los combina para crear un modelo global actualizado. El modelo global contiene los diccionarios de los clientes con los valores de estado-acción. Los resultados de las fases de entrenamiento son devueltos a los clientes para su evaluación. Por otro lado las métricas recibidas de la fase de evaluación se guardan para su análisis posterior.

El proceso de federación concluye una vez que se cumplen todas las rondas establecidas por el servidor.

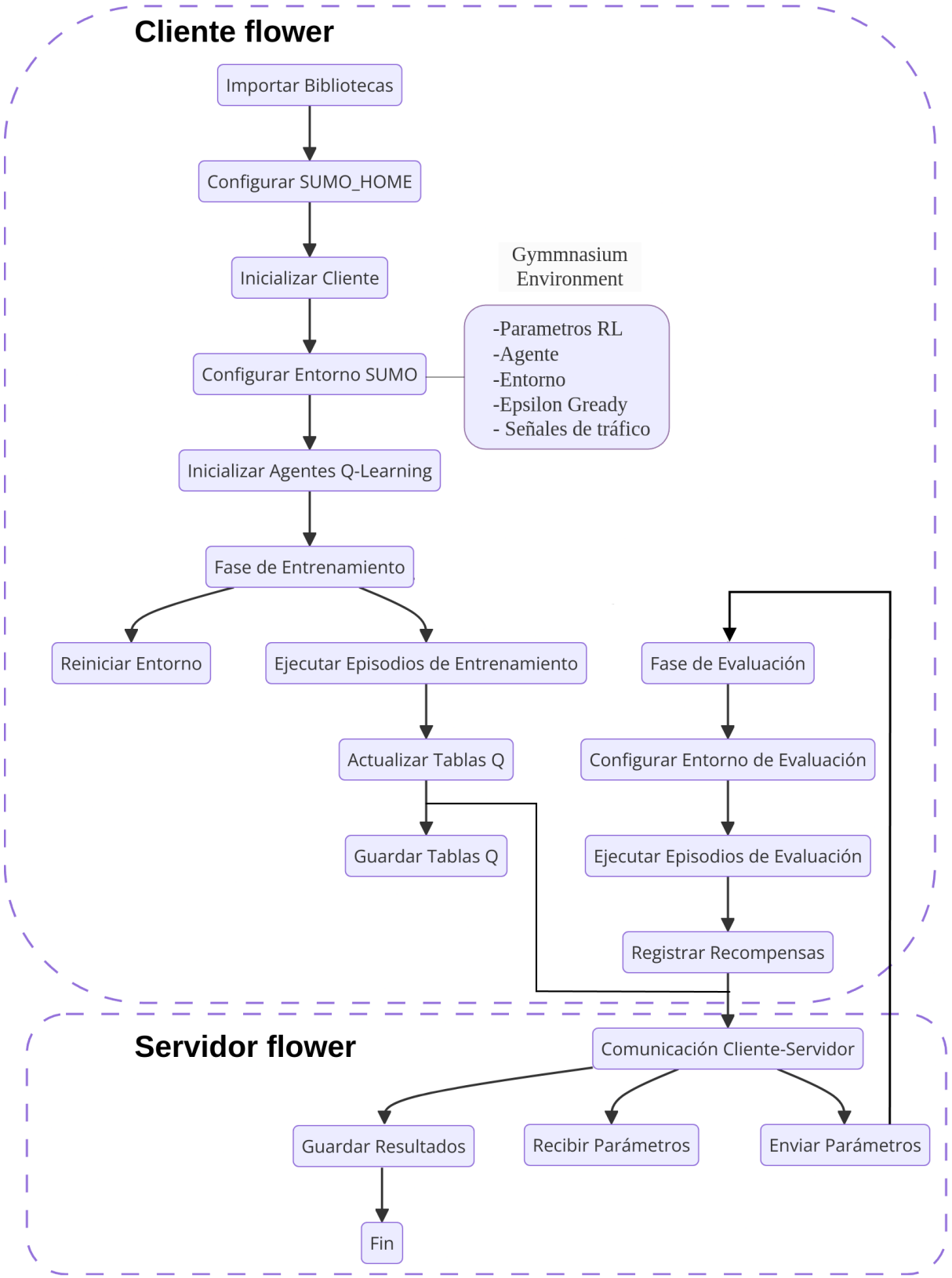


Figura 3.10: Esquema de Implementación Cliente - Servidor con Flower

4. Análisis de Resultados

En este capítulo se detallarán los resultados obtenidos al aplicar cada uno de los algoritmos de aprendizaje automático propuestos anteriormente, así como una comparación entre dos de estos enfoques. En la sección 4.1, se analizan los resultados del Aprendizaje por Refuerzo, específicamente aplicando el algoritmo de *Q-Learning*. Este análisis sirve como punto de partida para los dos siguientes tipos de aprendizaje: Aprendizaje No Supervisado y Aprendizaje Supervisado.

En la sección 4.2, se presentan los resultados del Aprendizaje No Supervisado, utilizando el algoritmo PCA. Este algoritmo permite reducir la dimensionalidad de los datos y descubrir patrones subyacentes sin la necesidad de etiquetas predefinidas.

En la sección 4.3, se describen los resultados del Aprendizaje Supervisado, empleando el algoritmo de *Random Forest*. Este método permite construir un modelo clasificador robusto basado en datos etiquetados, mejorando la precisión de las clasificaciones de datos mediante la combinación de múltiples árboles de decisión.

En la sección 4.4, se aborda el Aprendizaje Federado, aplicando nuevamente el algoritmo de *Q-Learning*, pero utilizando el marco del *framework Flower*. Este enfoque distribuye el proceso de aprendizaje a través de múltiples dispositivos o nodos, manteniendo los datos localmente y mejorando la privacidad.

Finalmente, en la sección 4.5, se realiza una comparación detallada entre el Aprendizaje por Refuerzo y el Aprendizaje Federado. Esta comparación evaluará el desempeño, la eficiencia y la escalabilidad de ambos enfoques, destacando sus ventajas y limitaciones en diferentes escenarios.

4.1. Aprendizaje por Refuerzo

Para el Aprendizaje por Refuerzo, como se mencionó anteriormente, se utiliza el algoritmo *Q-Learning*. Este algoritmo permite que los agentes aprendan y se adapten a los cambios en su entorno, así como a las diferentes variaciones que puedan presentarse. El objetivo es mejorar el comportamiento de los agentes mediante el aprendizaje continuo a partir de sus interacciones con el entorno.

Se analiza el comportamiento del aprendizaje bajo tres niveles distintos de densidad vehicular: *alta*, *media* y *baja*. Para lograr esto, se modifica el archivo *.rou.xml*, donde se encuentra el

parámetro *probability*. Este parámetro define la probabilidad de generación de un vehículo en cada segundo de la simulación, determinando así la frecuencia con la que los vehículos ingresan a la red vial desde el punto de origen hacia el destino especificado en cada flujo. Para cada uno de los niveles de densidad vehicular se utilizaron los parámetros indicados en la Tabla 4.1.

Parámetro	Valor
Tasa de aprendizaje α	0.1
Factor de descuento γ	0.99
Factor de decaimiento d	1
Número de ejecuciones completas del experimento	1
Episodios	10
Duración de la Simulación	86400

Tabla 4.1: Valores Utilizados para las Simulaciones

Los valores indicados anteriormente permiten tener una comprensión clara del funcionamiento de nuestro algoritmo y evaluar su desempeño. En primera instancia, consideramos una tasa de aprendizaje α de 0.1. La tasa de aprendizaje determina la cantidad de nueva información que sobrescribe la información antigua. En este caso, un valor de 0.1 permite un aprendizaje gradual, evitando grandes cambios que podrían desestabilizar el proceso de aprendizaje. Un valor pequeño asegura convergencia y estabilidad.

De igual manera, se utiliza el factor de descuento γ con un valor de 0.99 para estas simulaciones. Este factor determina la importancia de las recompensas futuras. Un valor cercano a uno indica que se otorga la misma importancia a las recompensas futuras como a las inmediatas. El valor propuesto permite que el agente tenga una visión a largo plazo.

Además, se ha utilizado un factor de decaimiento d de 1 en estas simulaciones. Este factor se refiere al decaimiento de la tasa de exploración-explotación. En este caso, un valor de 1 implica que no existe decaimiento, por lo que el agente mantiene una tasa constante de exploración durante todo el proceso de aprendizaje.

Por otra parte, se ha determinado el número de episodios, que es la cantidad de veces que el agente se entrena en el entorno. Se ha utilizado el valor propuesto porque, a partir de este valor y valores más altos, no se observa un incremento significativo en el aprendizaje.

En la Figura 4.1 se presentan los resultados de emisiones de CO_2 clasificado en tres categorías: Alta, Media y Baja. Cada barra representa el promedio de emisiones de CO_2 en gramos por segundo (g/s) para cada episodio. Por otro lado, en la Figura 4.2 se presentan los re-

sultados del tiempo de espera. Cada barra representa el promedio del tiempo de espera en segundos (s) para cada episodio. Las barras azules corresponden a las emisiones altas, las naranjas a las medias, y las verdes a las bajas. Las líneas negras en cada barra muestran los intervalos de confianza, indicando la variabilidad de los datos en cada categoría y episodio. De forma general se observa que los intervalos de confianza decrecen a medida que aumentan los episodios, es decir los valores varían menos conforme avanza la simulación. Para cada uno de los episodios se propone una duración de la simulación de 86,400 segundos, equivalente a un día completo. Este valor permite observar el comportamiento del sistema a lo largo de un periodo significativo de tiempo, facilitando la obtención de tendencias a largo plazo.

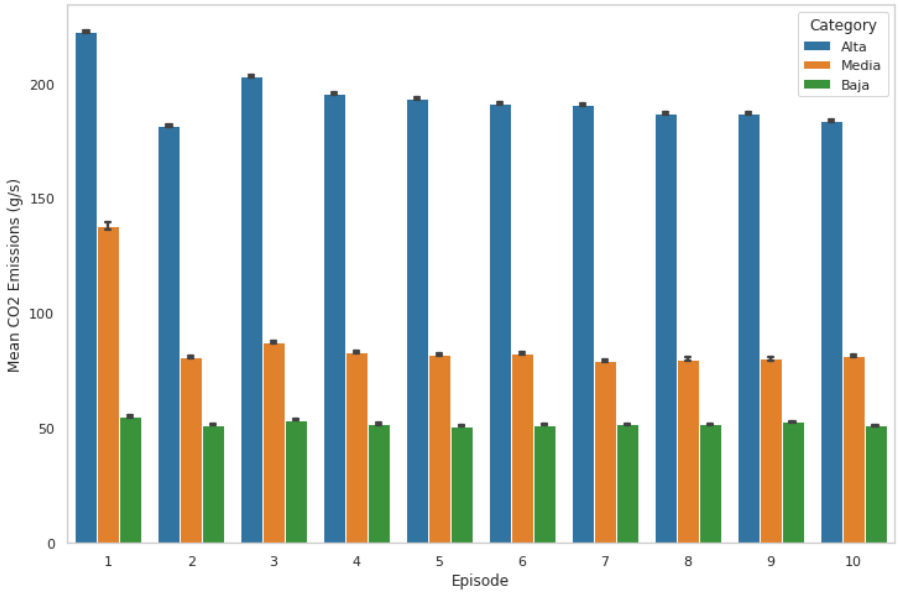


Figura 4.1: Promedio de Emisiones de CO_2 para Distintas Densidades

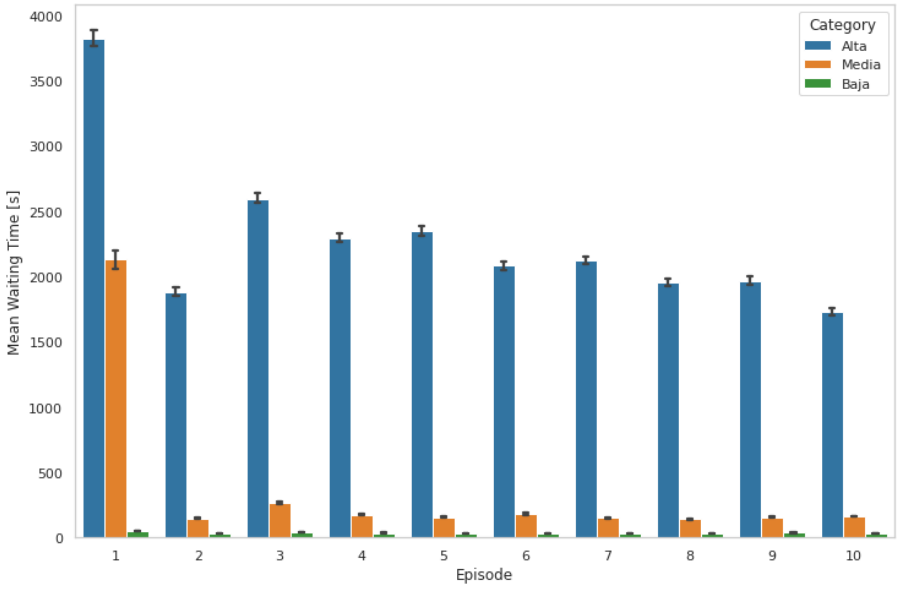
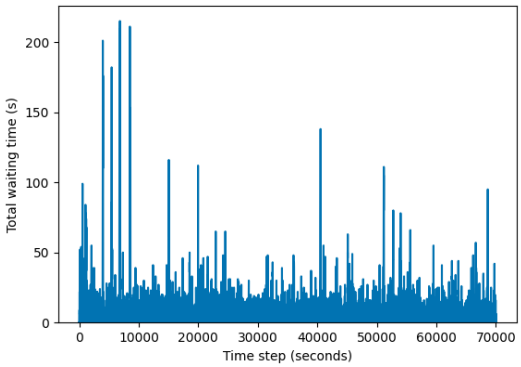


Figura 4.2: Promedio de Tiempo de Espera para Distintas Densidades

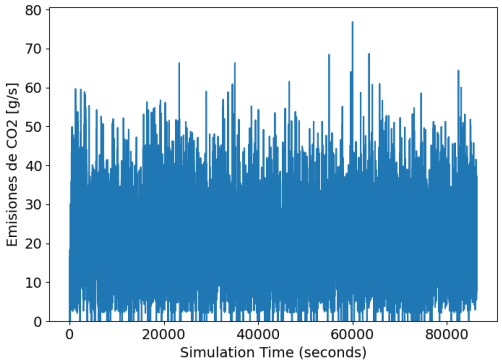
Para el caso de *Densidad Vehicular Baja*, se utilizó un valor de *probability* de 0.05, lo que indica una probabilidad del 5 % de que se genere un vehículo cada segundo. Las emisiones de CO_2 así como el tiempo de espera en el primer episodio, que es el más significativo, se presenta en la Figura 4.3.

Se observa en la Figura 4.3(a) que al inicio del episodio se tiene valores altos de tiempo de espera, que llegan a los 200 segundos, para después llegar a valores por debajo de los 100 segundos.

Por otra parte, se observa en la Figura 4.3(b) las emisiones de CO_2 obtenidas con la densidad vehicular baja. Las emisiones llegan a valores menores a 70 g/s, pero en promedio llegan a valores menores de 50 g/s. Hay picos altos en las emisiones, lo que indica momentos de mayor congestión vehicular o ineficiencias en el tráfico. Los altos picos en el tiempo de espera podrían indicar congestión vehicular, lo cual generalmente lleva a un aumento en las emisiones de CO_2. Esto es coherente con los datos de las emisiones de CO_2, donde se observan altos picos que pueden coincidir con los tiempos de espera prolongados.



(a) Tiempo de Espera



(b) Emisiones de CO_2

Figura 4.3: Resultados con Densidad Vehicular Baja

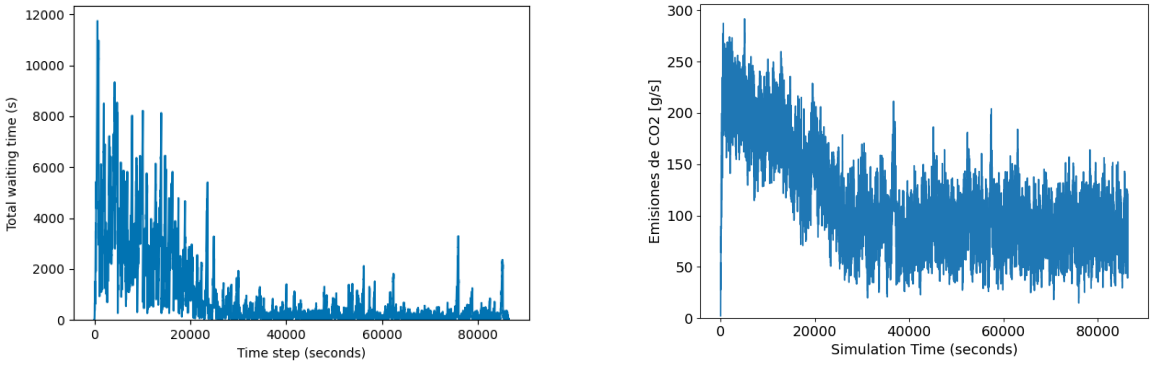
Para verificar que el número de episodios realizados en cada simulación converge, se realiza el monitoreo de convergencia. Se analizan las gráficas obtenidas en cada episodio para identificar el punto en el que tienden a un valor único. Todas las gráficas para los análisis de convergencia se presentan en el Anexo A.B. En la Figura A.1 se observa como con tres episodios realizados, ya existe una convergencia de un valor cercano a 40 segundos de tiempo de espera. Esto permite afirmar que, en la densidad vehicular baja, con tres episodios de simulación se puede obtener este valor constante. Sin embargo, al momento de analizar los valores promedio de emisiones de CO_2 y tiempo de espera de la Figura 4.1 y Figura 4.2 se observa que existe una tendencia constante a lo largo de los episodios, lo que dificulta comprobar el funcionamiento del algoritmo de *Q-learning* con esta densidad.

Para el caso de *Densidad Vehicular Media*, se utiliza un valor de *probability* de 0.15, lo que indica una probabilidad del 15 % de que se genere un vehículo cada segundo.

En la Figura 4.4 se observa el comportamiento del *Tiempo de espera* y *Emisiones* CO_2 en el primer episodio. En primera instancia, en la Figura 4.4(a) se puede observar como varía el tiempo de espera de los vehículos. Se observa que al inicio de la simulación se tienen picos que alcanzan los 12000 segundos, pero a medida que avanza la simulación este valor va reduciendo hasta llegar a valores menores a los 1000 segundos de tiempo de espera, lo que indica que el algoritmo está funcionando de manera adecuada.

Así también, en la Figura 4.4(b) se puede notar como se tiene al inicio de la simulación valores muy altos de emisiones de CO_2, en este caso alcanza los niveles de más de 250 g/s para luego ir decreciendo estos valores hasta llegar a valores alrededor de los 100 g/s. Esto es un buen indicativo de que el algoritmo funciona de manera adecuada, ya que ayuda a reducir el

nivel de contaminación, en este caso de CO_2. Se puede apreciar de mejor manera como el reducir el tiempo total de espera de los vehículos ayuda a reducir las emisiones de CO_2.



(a) Tiempo de Espera (b) Emisiones de CO_2

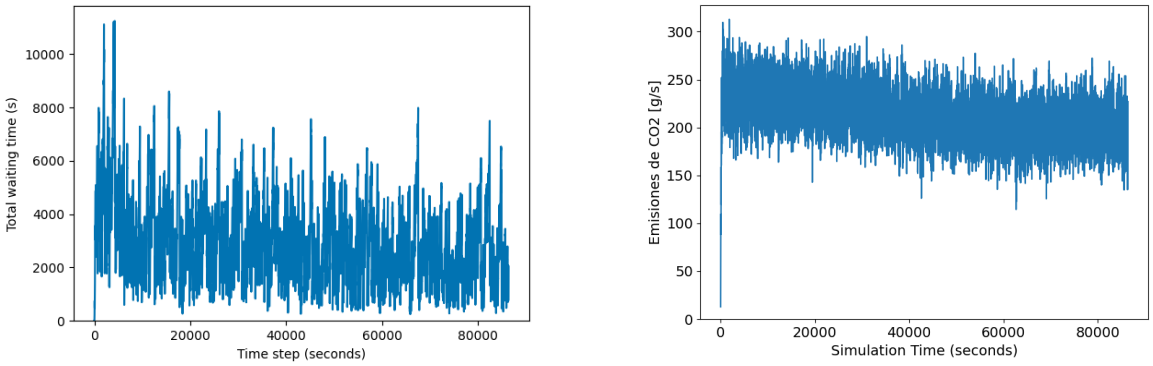
Figura 4.4: Resultados con Densidad Vehicular Media

En la Figura A.2 se puede observar como con cinco episodios realizados, existe ya una convergencia de un valor cercano a 800 segundos de tiempo de espera. Con lo que se puede decir que en la densidad vehicular media, con cinco episodios de simulación se puede obtener este valor constante. Esta tendencia se observa también en los valores promedio de emisiones de CO_2 y tiempo de espera de la Figura 4.1 y Figura 4.2, en donde ya se puede observar como el tiempo de espera y las emisiones de CO_2 disminuyen a lo largo de los episodios.

Para el caso de *Densidad Vehicular Alta*, se utilizó un valor de *probability* de 0.30, lo que indica una probabilidad del 30 % de que se genere un vehículo cada segundo.

En la Figura 4.5 se puede observar el comportamiento del *Tiempo de espera* y *Emisiones* CO_2. En primer lugar, se puede observar en la Figura 4.5(a) como varía el tiempo de espera de los vehículos respecto al tiempo de simulación. A medida que avanza la simulación, se aprecia una disminución en el tiempo de espera, lo que indica que el algoritmo está funcionando de manera adecuada. Esta variación puede ser notada, ya que al inicio de la simulación se tiene un tiempo total de espera de 10000 segundos y al finalizar la misma se tiene un valor menor a los 6000 segundos.

De igual forma, en la Figura 4.5(b) se puede analizar como empieza la simulación con valores cercanos a los 300 g/s de emisiones de CO_2, para a medida que avanza la simulación esta vaya disminuyendo, que en este caso llega a valores por debajo de las 200 g/s emisiones de CO_2. Por lo que se aprecia que el algoritmo sigue funcionando, no tan rápido como en la densidad media, pero sigue funcionando.



(a) Tiempo de Espera (b) Emisiones de CO_2

Figura 4.5: Resultados con Densidad Vehicular Alta

En la Figura A.3 se puede observar como con ocho episodios realizados, existe ya una convergencia de un valor cercano a 6000 segundos de tiempo de espera. Con lo que se puede decir que en la densidad vehicular alta, con cinco episodios de simulación se puede obtener este valor constante. En los valores promedio de emisiones de CO_2 y tiempo de espera de la Figura 4.1 y Figura 4.2 se observar como bajo estas condiciones el tiempo de espera y las emisiones de CO_2 disminuyen a lo largo de los episodios.

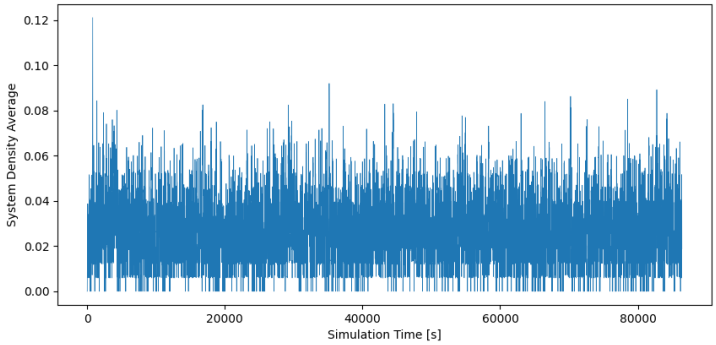
4.1.1. Comparación de Densidades Vehiculares

En primer lugar, se compararon las tres densidades propuestas: *Alta*, *Media* y *Baja*. En la Figura 4.6 se aprecia cómo varía la densidad vehicular en cada caso.

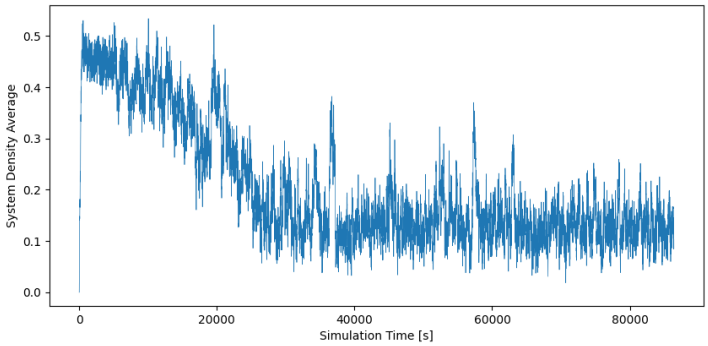
En la Figura 4.6(a), se observa la densidad baja, que tiene un pico máximo de 0.12. Esta densidad muestra una gran variabilidad con valores que oscilan principalmente entre 0.02 y 0.06 durante la mayor parte del tiempo de simulación.

En la Figura A.2(b), se aprecia la densidad media, que tiene el pico máximo alcanza un valor de 0.5. Sin embargo, después de aproximadamente 30,000 segundos de simulación, la densidad comienza a disminuir, estabilizándose en un valor estimado de 0.2. Esto indica que el sistema experimenta una reducción significativa de la densidad con el tiempo.

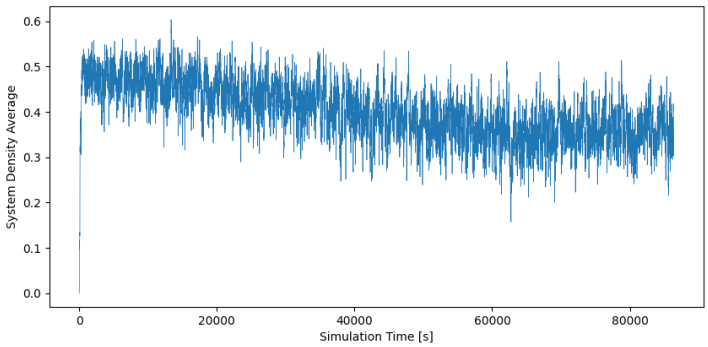
En la Figura A.3(b), se tiene la densidad alta, en la cual se observa el pico máximo es de 0.6, pero se observa una tendencia decreciente que lleva la densidad a estabilizarse en un valor alrededor de 0.4. Este comportamiento sugiere que, aunque inicialmente la densidad es alta, hay mecanismos en el sistema que ayudan a reducirla parcialmente con el tiempo.



(a) Densidad Vehicular Baja



(b) Densidad Vehicular Media



(c) Densidad Vehicular Alta

Figura 4.6: Escenarios Utilizados para el Aprendizaje Federado

También se consideró la duración total de la simulación para cada caso. Los resultados obtenidos se presentan en la Tabla 4.2. La simulación con mayor duración corresponde al caso de Densidad Vehicular Alta, alcanzando un tiempo de 2920.64 segundos. Este resultado era previsible, dado que un mayor número de vehículos en la red incrementa el tiempo necesario para completar la simulación.

Duración	Tiempo [s]
Densidad Vehicular Baja	174.08
Densidad Vehicular Media	468.00
Densidad Vehicular Alta	4292.20

Tabla 4.2: Tiempo de Duración de cada Simulación

4.1.2. Evaluación del Modelo

El modelo de RL se evalúa utilizando métricas estándar para la evaluación de modelos dé RL. Para realizar estas evaluaciones, se parte de los resultados obtenidos anteriormente, para trabajar sobre la densidad vehicular en la que se obtuvieron resultados más evidentes en cuanto al funcionamiento del algoritmo, que en este caso fue la densidad vehicular Media.

Para la evaluación del modelo, se debe definir en primer lugar los parámetros con los cuales se trabaja. En este caso, se usa casi los mismos valores y parámetros listados en la Tabla 4.1, con el cambio en el número de episodios realizados, que en este caso, se usa 10 episodios.

La Figura 4.7 presenta la métrica de la varianza explicada EVa. Esta métrica evalúa si la política aprendida es un buen predictor del rendimiento, es decir, de la recompensa total. Un valor de $EVa \leq 0$ índica que la predicción es peor que no hacer ninguna predicción, mientras que un valor de $EV_a = 1$ índica una excelente predicción [61]. En la Figura 4.7, se observa que EV_a se aproxima a 1, por lo que se ha encontrado una estrategia eficaz a nuestro problema.

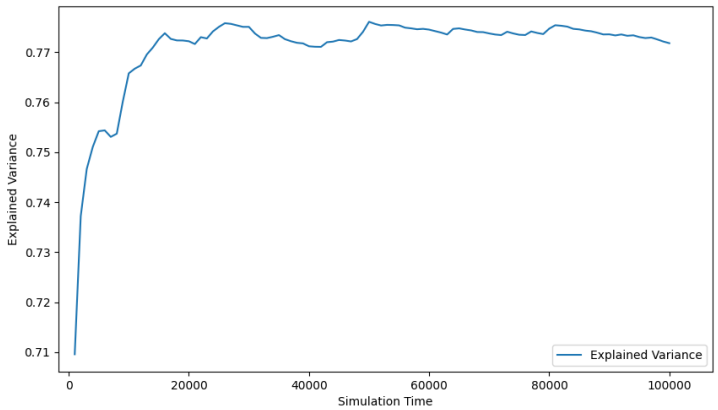


Figura 4.7: Varianza Explicada del Modelo

Otra de las métricas utilizadas para evaluar el modelo de RL es la pérdida de valor. Para calcular la pérdida de valor se calcula las recompensas acumuladas de los agentes presentes

en la simulación. La Figura 4.8 presenta las recompensas acumuladas a lo largo del tiempo de simulación. Como se indica en la Ecuación 3.2, esta recompensa es negativa.

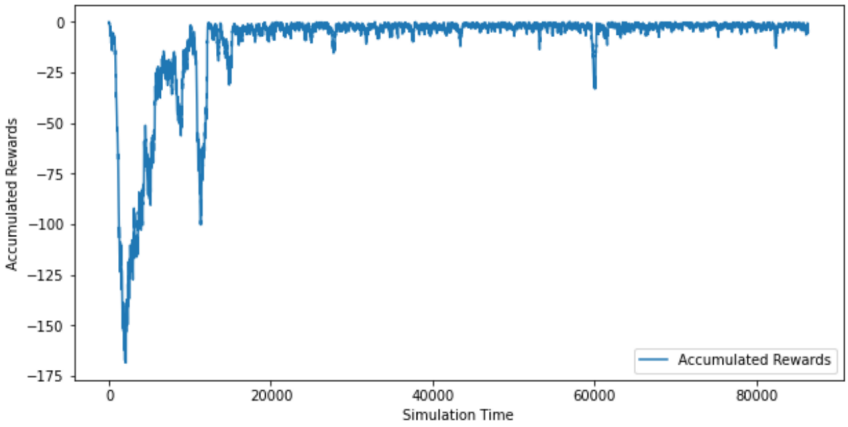


Figura 4.8: Recompensa Acumulada

La pérdida de valor se presenta en la Figura 4.9. Esta métrica se calcula como la diferencia de valor entre pasos para la recompensa. Se observa que al inicio de la simulación hay un gran cambio en el valor de la recompensa, pero a partir de los 20,000 pasos, esta tiende a estabilizarse, indicando un aprendizaje en los agentes.

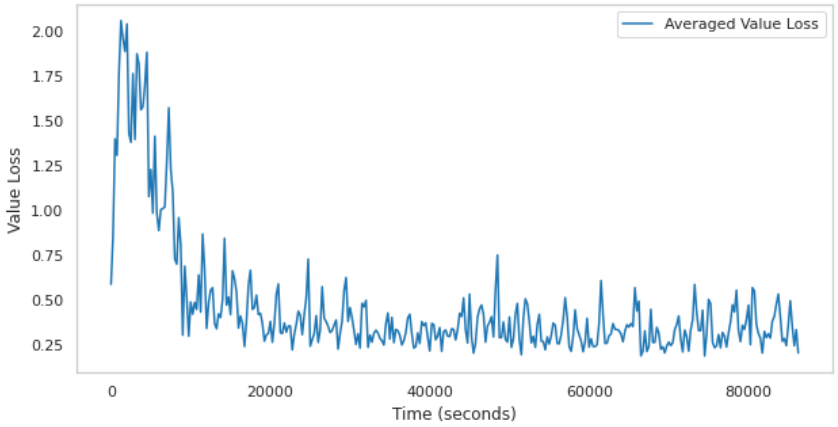


Figura 4.9: Pérdida de Valor

4.2. Aprendizaje No Supervisado

Como se menciona en la sección anterior, se trabaja con 35 variables, las cuales se pueden observar en la tabla 3.2 y con 14000 datos de cada una de ellas, por lo que al aplicar él PCA, se busca obtener la cantidad de variables mínimas relevantes, las componentes principales y las variables más importantes que se han encontrado. Así también, se compararán dos métodos de normalización, como lo son el StandardScaler y MinMaxScaler.

4.2.1. StandardScaler

Al utilizar el método $\mathrm{StandardScaler}$ para la normalización, se observa que se pueden utilizar 3 componentes principales con los cuales se puede abarcar aproximadamente el 83.1 % de la varianza de los datos, como se muestra en la Figura. 4.10.

Dado que se tiene un alto porcentaje de varianza acumulada utilizando tres componentes principales, es posible considerar estas tres componentes como suficientes para una clasificación efectiva de los datos. Estas tres componentes mencionadas están compuestas por las variables más relevantes, por lo que en las Tablas 4.4, 4.5 y 4.6 están las componentes principales uno, dos y tres respectivamente.

En la Figura 4.11, se presenta la proyección de los datos en los tres primeros componentes principales, utilizando $\mathrm{StandardScaler}$ para la normalización de los datos. Esta figura proporciona una visión detallada de la distribución de los datos en diferentes combinaciones de componentes principales, las cuales están en las Tablas 4.4, 4.5 y 4.6.

De color rojo se observa como los datos se distribuyen en el espacio definido por las dos primeras componentes principales, que capturan la mayor parte de la varianza en los datos originales. La distribución en esta proyección revela la estructura principal de los datos. De color verde se muestra, en cambio, como los datos se distribuyen en la primera y tercera componentes principales. Aunque la tercera componente captura menos varianza que la primera y segunda, sigue siendo significativa y ofrece una perspectiva adicional sobre la estructura de los datos. Mientras que de color azul, se visualiza la relación entre la segunda y tercera componentes principales. Esta proyección es útil para entender las variaciones secundarias en los datos y cómo se distribuyen en las componentes de varianza menor.

La utilización de las tres primeras componentes principales, las cuales están en las Tablas 4.4, 4.5 y 4.6, que explican un alto porcentaje de la varianza total, es suficiente para comenzar a clasificar los datos. La proyección de los datos en estas componentes muestra una estructura clara y bien definida, lo que facilita la identificación de patrones y agrupamientos.

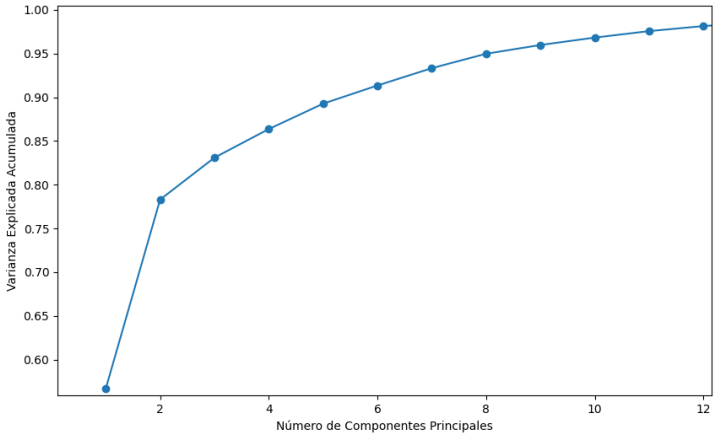


Figura 4.10: Varianza Acumulada por Componentes Principales StandardScaler

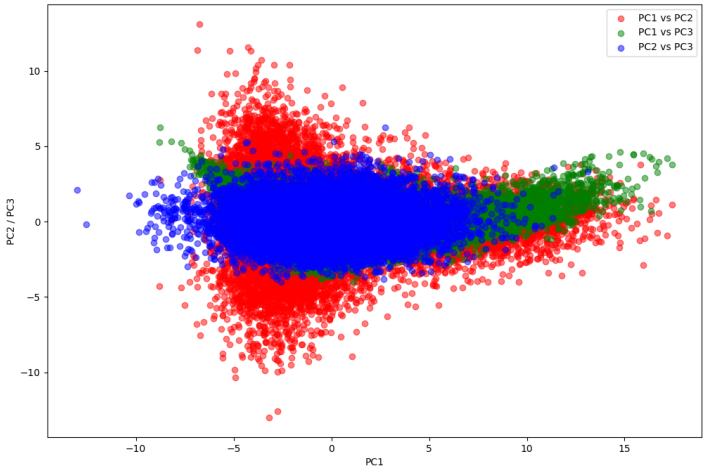


Figura 4.11: Proyección de Datos en las Primeras Tres Componentes Principales StandardScaler

4.2.2. MinMaxScaler

Por otra parte, al utilizar el método MinMaxScaler para la normalización se muestra igualmente que utilizando 3 componentes principales abarcamos aproximadamente el 86.5 %, que es más que al utilizar el método StandardScaler. Este resultado podemos apreciarlo en la Figura 4.12.

De igual manera que en el anterior método, se ha obtenido un porcentaje de varianza acumulada aceptable para utilizar únicamente tres componentes principales para realizar una clasificación efectiva de datos. En la figura 4.13 se presenta la proyección de los datos de las tres componentes principales utilizando en este caso MinMaxScaler para la normalización. Se

utilizan los mismos colores y las mismas componentes principales en comparación, con la diferencia más significada que al comparar las componentes dos y tres están más centradas.

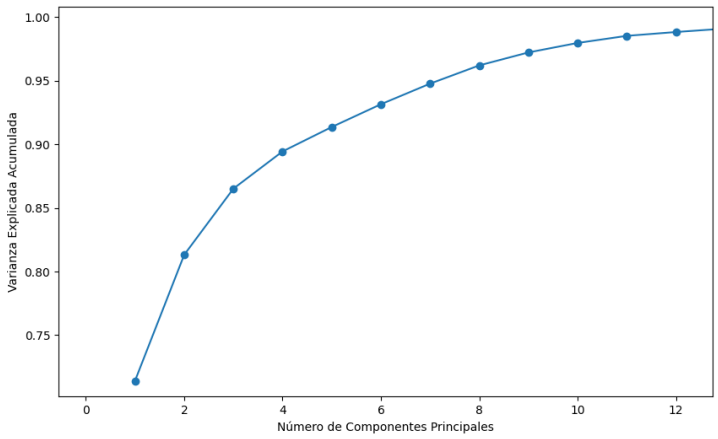


Figura 4.12: Varianza Acumulada por Componentes Principales MinMaxScaler

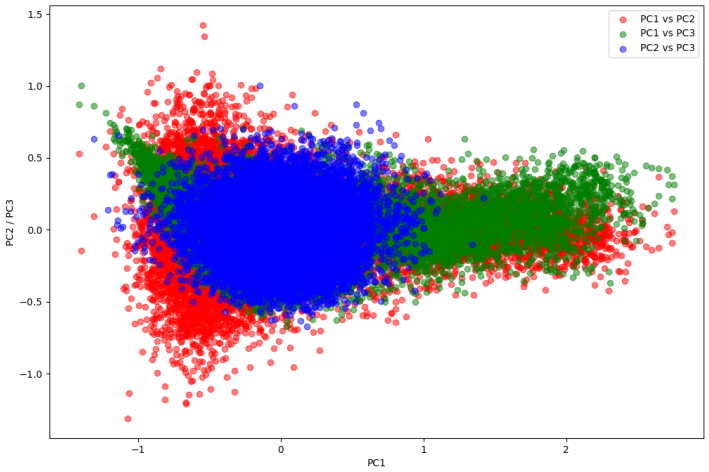


Figura 4.13: Proyección de Datos en las Primeras Tres Componentes Principales MinMaxScaler

4.2.3. Varianza Obtenida en las Primeras Seis Componentes Principales

A continuación, en la tabla 4.3 se detalla el valor de cada componente principal en cuanto a la varianza, en donde se observa las seis componentes principales para poder confirmar que con el uso de las primeras tres se puede empezar a usar un clasificador de datos. Cabe mencionar, que desde la componente principal 22 hasta la última componente principal el valor es de 0 %.

Varianza por Cada Componente Principal	Porcentaje (%)
Componente Principal 1 (PC1)	56.67
Componente Principal 2 (PC2)	21.63
Componente Principal 3 (PC3)	4.80
Componente Principal 4 (PC4)	3.27
Componente Principal 5 (PC5)	2.90
Componente Principal 6 (PC6)	2.08

Tabla 4.3: Varianza de las Primeras Seis Componentes Principales

4.2.4. Variables más Importantes en las Tres Primeras Componentes Principales

- **Componente Principal 1 (PC1):** Las variables que más contribuyen a PC1 son aquellas relacionadas con el tráfico y el consumo de combustible. Los valores obtenidos por cada una de las variables se observa en la Tabla 4.4.

Variable	**Valor**	**Variable**	**Valor**
system_total_stopped	0.223369	system_out_lanes_density_avg	0.216528
agents_total_stopped	0.223369	system_total_fuel_consumption	0.213701
system_queue_avg	0.223156	system_total_C02_emissions	0.213699
system_vehicles	0.220119	agents_total_accumulated_waiting_time	0.213402
system_lanes_density_avg	0.219356	system_mean_total_queued	0.213402

Tabla 4.4: Valores de las Variables más Importantes en la Componente Principal 1

De igual manera se define la tabla anterior mediante el uso de la siguiente ecuación:

$$\text{PC1} = W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + \cdots + W_{1p}x_p \tag{4.1}$$

Donde:

- PC1 es la primera Componente Principal.
- W_{1j} es el coeficiente o peso de la variable x_j en la primera Componente Principal.
- x_j es la j-ésima variable original del conjunto de datos.
- p es el número total de variables originales [62].

Esta ecuación puede ser utilizada para representar cualquier Componente Principal, se debe tener en consideración el número de componente con la cual estamos trabajando.

- **Componente Principal 2 (PC2):** PC2 está dominada por variables relacionadas con el promedio y la densidad de la cola del sistema. Los valores obtenidos por cada una de las variables se observa en la Tabla 4.5.

Variable	Valor	Variable	Valor
s2_stopped	0.017396	system_total_stopped	0.012875
s1_stopped	0.013502	system_mean_waiting_time	0.009303
system_queue_avg	0.013332	system_total_waiting_time	0.008509
system_lanes_density_avg	0.013204	s4_accumulated_waiting_time	0.007573
agents_total_stopped	0.012875	s2_accumulated_waiting_time	0.00673

Tabla 4.5: Valores de las Variables más Importantes en la Componente Principal 2

- **Componente Principal 3 (PC3):** PC3 destaca las velocidades promedio de las secciones del sistema, además del tiempo total de espera y otros tiempos acumulados. Los valores obtenidos por cada una de las variables se observa en la Tabla 4.6.

Variable	Valor	Variable	Valor
s4_average_speed	0.398883	system_mean_waiting_time	0.195839
s3_average_speed	0.398883	agents_total_accumulated_waiting_time	0.188464
s2_average_speed	0.398883	system_mean_total_queved	0.188464
s1_average_speed	0.398883	s4_accumulated_waiting_time	0.179076
system_total_waiting_time	0.210222	s2_accumulated_waiting_time	0.178604

Tabla 4.6: Valores de las Variables más Importantes en la Componente Principal 3

En este estudio, se realiza un PCA sobre un conjunto de datos que consta de 35 variables y 14,000 observaciones por cada una de ellas. El objetivo para llevar a cabo este PCA fue la necesidad de reducir la dimensionalidad del conjunto de datos. Al tratar con un alto número de variables, es fundamental abordar la posible redundancia y alta correlación entre ellas, lo cual puede afectar negativamente el rendimiento de los algoritmos de aprendizaje supervisado.

La reducción dimensional mediante PCA permite transformar el conjunto de datos original en un espacio de menor dimensión, preservando la mayor parte de la variabilidad intrínseca a los datos. Este proceso no solo facilita una mejor visualización y comprensión de los patrones subyacentes en los datos, sino que también mejora la eficiencia computacional y el rendimiento de los modelos de aprendizaje supervisado que se implementarán posteriormente.

4.3. Aprendizaje Supervisado

Las razones para aplicar aprendizaje supervisado en este análisis surge de la necesidad de clasificar el tiempo de espera y, consecuentemente, los niveles de emisiones de CO_2. Inicialmente, se realizó un PCA como una técnica de aprendizaje no supervisado para reducir la dimensionalidad del conjunto de datos y extraer las características más relevantes. Con los datos transformados y simplificados obtenidos del PCA, se procedió a implementar el aprendizaje supervisado con el fin de clasificar el tiempo de espera y los niveles de CO_2 en categorías de alto, medio y bajo.

Esta clasificación es para identificar patrones y tendencias en el comportamiento del tráfico vehicular y sus emisiones. El objetivo final de esta clasificación es optimizar la duración de las luces verdes en los semáforos. Al ajustar adecuadamente los tiempos de las señales de tráfico, se busca reducir las emisiones de CO_2, disminuyendo el tiempo de espera de los vehículos y mejorando la fluidez del tráfico.

Como se mencionó anteriormente, en esta sección se presentan los resultados obtenidos al aplicar el algoritmo *Random Forest*. Este algoritmo se implementa con el propósito de clasificar los tiempos de espera y emisiones de CO_2 en diferentes niveles, esto se obtiene utilizando la duración de la luz verde de cada uno de los semáforos. Cabe destacar que la variable *tiempo de espera total*, contenida en la componente principal uno mostrada en la Tabla 4.4, ha demostrado ser una de las más significativas, como se analizó en la sección 4.2. Por esta razón, se utiliza esta variable para clasificar los niveles de congestión en alto, moderado y bajo y por ende, poder clasificar estos datos en alto, moderado y bajo respecto a las emisiones de CO_2 generadas.

En la Figura 4.14 se presenta la clasificación en niveles de emisiones de CO_2 en función del tiempo de duración de encendido de la luz verde de cada semáforo, el cual varía entre 5 y 25 segundos. En la figura, se puede observar que, a medida que se dispone de una mayor cantidad de datos en cada una de las clases, el tamaño de las burbujas aumenta. El punto central de cada burbuja representa el promedio de cada clase en relación con los diferentes tiempos de duración del semáforo.

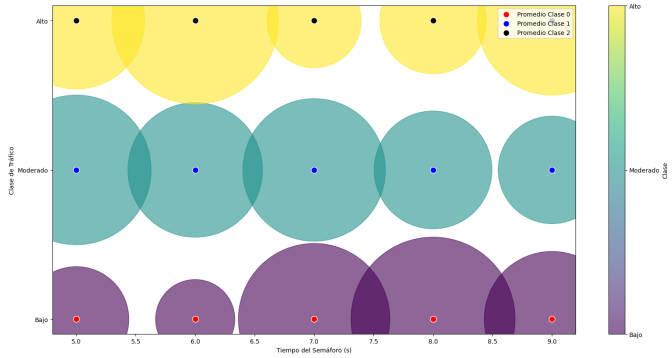


Figura 4.14: Clasificación de Niveles de CO_2 según el Tiempo del Semáforo

También, se ha podido obtener la cantidad de datos por cada una de las clases a lo largo de la duración del semáforo. Esto se puede apreciar en la Figura 4.15, en donde se puede ver la variabilidad de la cantidad de datos de cada una de las clases, ya sea alta, moderada o baja, dependiendo del tiempo de duración del semáforo que se configure. En esta figura, se visualiza que el tiempo del semáforo que se configure influye significativamente en las emisiones de CO_2 y en la congestión del tráfico, ya que se observan intervalos de tiempo donde se tiene congestión más alta y por ende, mayor nivel de emisiones de CO_2. El tiempo de semáforo de 15 segundos es el que tiene un flujo de tráfico más bajo y menor nivel de emisiones de $CO2$, lo cual es la duración de semáforo óptima.

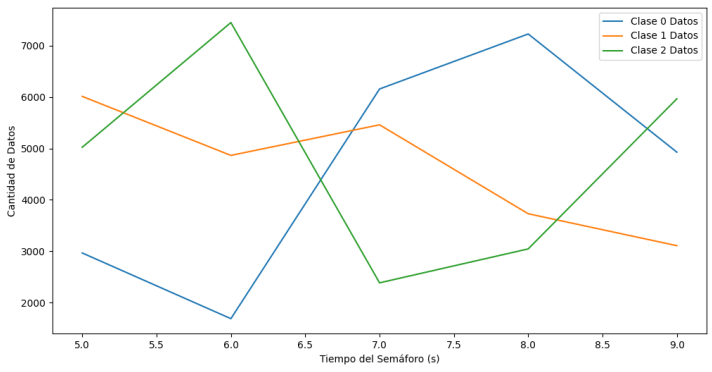


Figura 4.15: Cantidad de Datos por Clase a lo Largo del Tiempo de Semáforo

En la Tabla 4.7 tenemos el informe de clasificación al aplicar *Random Forest*, en ella se aprecia lo siguiente:

- *Precision:* Mide la exactitud de predicciones positivas.
- *Recall:* Mide la capacidad del modelo para identificar las instancias positivas.

- *F1-Score:* Es la media armónica de Precision y Recall, proporcionando un equilibrio entre ambas métricas.
- *Support:* Indica el número de instancias reales de cada clase en el conjunto de prueba.
- *Accuracy:* Indica la proporción de instancias correctamente clasificadas sobre el total de instancias.
- *Macro Avg:* Es la media de las métricas (Precision, Recall, F1-Score) calculadas independientemente para cada clase.
- *Weighted Avg:* Es la media ponderada de las métricas, teniendo en cuenta el Support de cada clase. [63].

	Precision	Recall	F1-Score	Support
Bajo	1.00	0.95	0.97	6984
Modera	0.89	1.00	0.94	6841
Alto	1.00	0.93	0.96	7175
Accuracy			0.96	21000
Macro Avg	0.96	0.96	0.96	21000
Weighted Avg	0.96	0.96	0.96	21000

Tabla 4.7: Informe de Clasificación

En la Tabla 4.7 se puede observar que: en los niveles *Bajo* y *Alto*, todas las predicciones fueron correctas, mientras que en la clase *Moderada* se alcanzó un 89 % de predicciones correctas. De igual manera, solo en la clase *Moderada* se identificaron correctamente todas las instancias reales. En la clase *Baja* se logró un 95 %, y en la clase *Alta* se obtuvo un 93 %.

El valor más alto se registró en la clase *Bajo* con un 0.96, indicando un buen equilibrio entre precisión y exhaustividad. Para las clases *Moderada* y *Alta*, se obtuvieron valores de 0.94 y 0.96 respectivamente, también mostrando un buen equilibrio entre estas dos métricas. La cantidad de instancias por clase es la siguiente: *Baja* con 6984, *Moderada* con 6841, y *Alta* con 7175 instancias.

La precisión general del modelo es del 96 %, lo que indica que este porcentaje de instancias fueron clasificadas correctamente. Los promedios macro y ponderado (Macro Avg y Weighted Avg) del modelo son ambos de 0.96. Observadas todas estas características, se puede decir que el modelo muestra un rendimiento sólido y equilibrado en todas las clases, con una alta precisión general y promedios consistentemente altos.

4.4. Aprendizaje Federado

El objetivo de aplicar FL es mejorar la privacidad en el manejo de los datos de los clientes. A través del aprendizaje federado, los datos permanecen en los dispositivos locales y solo se comparten los parámetros del modelo, evitando así la transferencia de datos sensibles. Aunque se espera que el rendimiento del modelo sea inferior en comparación con el aprendizaje por refuerzo, esta metodología proporciona una mayor protección de la privacidad de los usuarios. Este enfoque se usa en contextos donde la confidencialidad de los datos es una prioridad, permitiendo desarrollar modelos robustos mientras se salvaguardan los datos personales de los clientes.

Para analizar los resultados obtenidos con el FL, se parte del escenario propuesto en la Sección 3.1.1, en el cual contamos con cuatro clientes y un servidor central. Cada uno de los clientes tendrá su modelo funcional de RL y enviará los resultados al nodo central o servidor para obtener un modelo general. Para realizar este análisis, se utilizan los parámetros iniciales de simulación indicados en la Tabla 4.8.

Parámetros	Valor
Duración de la Simulación	3500
Densidad Vehicular	0.15
Número Episodios	1
Número de Rondas (Federado)	25
Runs	1
ϵ	100

Tabla 4.8: Valores Utilizados para Realizar FL

Inicialmente se realizaron simulaciones considerando 80 rondas en la ejecución del algoritmo de FL. Realizando el promedio entre los datos ofrecidos por los clientes, se obtiene el tiempo promedio de espera presentado en la Figura 4.16, en donde se muestran los datos reales y una regresión polinómica que permite observar la tendencia de los datos. De manera similar, en la Figura 4.17 se presentan las emisiones de CO_2 promedio. Estas figuras permiten observar que a partir de la ronda 25 el modelo tiende a estabilizarse, razón por la cual se establecen 25 rondas como el rango de análisis del algoritmo de FL. Como se desea obtener una distribución del algoritmo de *Q-learning*, se establece una duración de la simulación de 3500 segundos, cubriendo así los 86400 segundos que son el objeto de análisis.

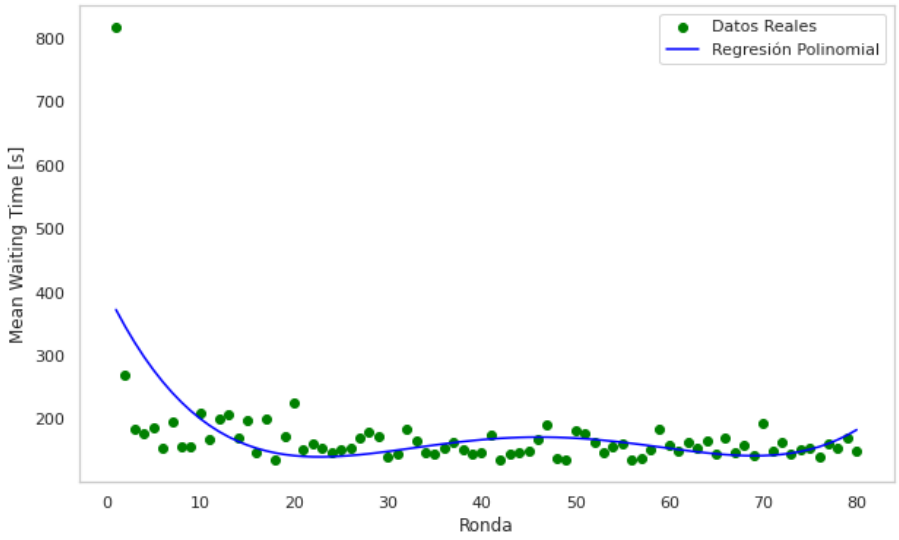


Figura 4.16: Tiempo Promedio de Espera por Ronda

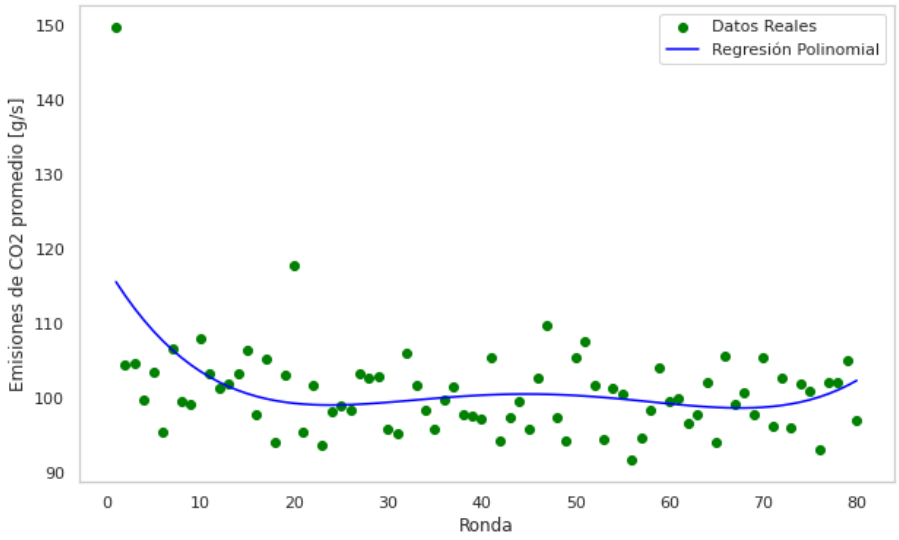


Figura 4.17: Emisiones de CO_2 Promedio por Ronda

4.4.1. Fase de Entrenamiento

En la Figura 4.18 se presenta el proceso de entrenamiento del cliente 0. Se observa una notable disminución en el tiempo de espera promedio del cliente en cada una de las rondas de federación, comenzando en aproximadamente 600 segundos y terminando por debajo de los 200 segundos. Los intervalos de confianza, en donde se consideró un valor de confianza del 95 %, demuestran que en las primeras rondas de entrenamiento existe una mayor variabilidad en el tiempo de espera promedio, la cual se reduce a medida que el propio modelo de federación se estabiliza.

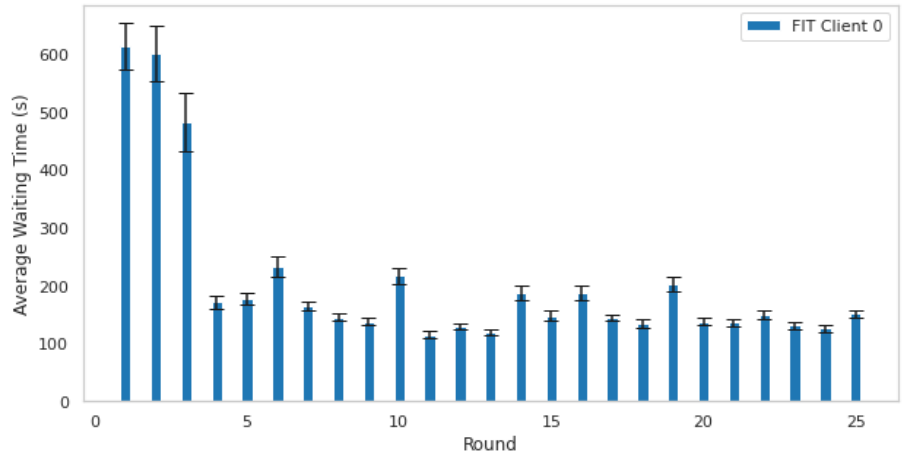


Figura 4.18: Resultados Obtenidos de Tiempo de Espera en cada Entrenamiento Cliente 0

De igual manera se analiza el comportamiento de las emisiones de CO_2 en cada uno de los episodios de entrenamiento del cliente 0, esto en la Figura 4.19. Se observa que se tiene un alto valor de emisiones, en este caso alcanza los valores de 120 g/s para luego disminuir significativamente a valores inferiores a los 100 g/s. El comportamiento del resto de los clientes es similar al observado en el cliente 0, y se detallan en el Anexo A.C.A.

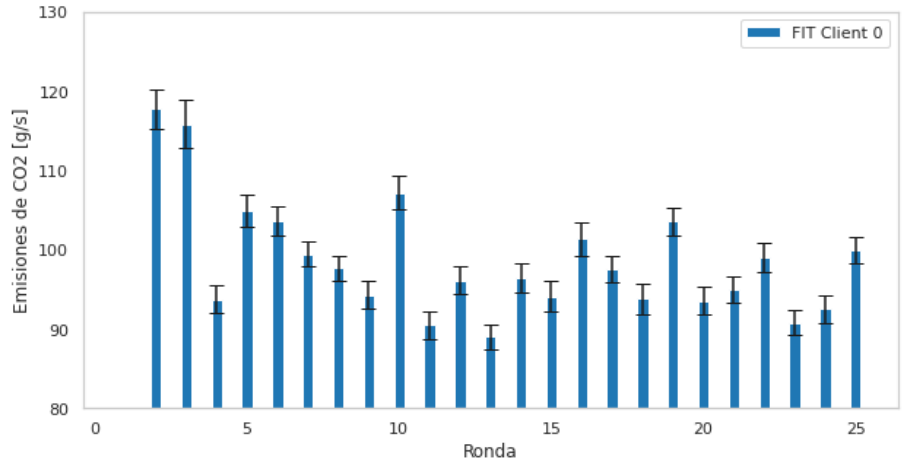


Figura 4.19: Resultados Obtenidos de Emisiones de CO_2 en cada Entrenamiento Cliente 0

4.4.2. Fase de Evaluación Utilizando los Parámetros Recibidos del Servidor.

Una vez completado el entrenamiento en cada uno de los clientes, estos envían sus parámetros al nodo central. Con toda esta información recopilada, el nodo central genera nuevos parámetros, que son posteriormente enviados de vuelta a cada cliente para evaluar el modelo actualizado. A continuación, se presenta el comportamiento de estos parámetros para el

cliente 0.

En la Figura 4.20, se presentan los resultados del cliente 0 en cuanto al promedio de tiempo de espera en cada ronda. Se observa que el valor máximo sobrepasa los 400 segundos y corresponde a la primera ronda de evaluación. Se visualiza una notable reducción en el tiempo de espera a medida que avanzan las rondas llegando a valores cercanos a los 400 segundos. En cuanto a los intervalos de confianza se observa una menor variabilidad a a medida que aumentan las rondas.

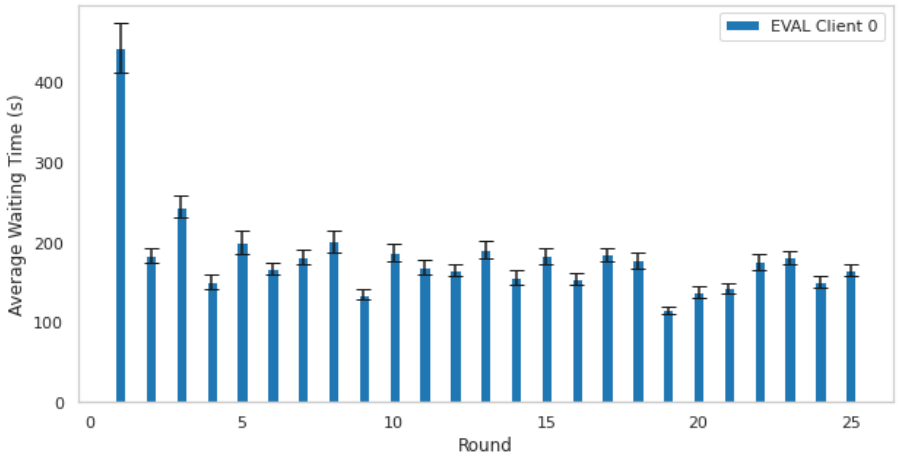


Figura 4.20: Evaluación de Tiempo de Espera del Modelo Recibido en el Cliente 0

De igual manera, se analizan las emisiones de CO_2 generadas durante la fase de evaluación. Estas emisiones se pueden observar en la Figura 4.21, donde se aprecia una reducción a medida que avanzan las rondas, con valores alrededor de 100 g/s.

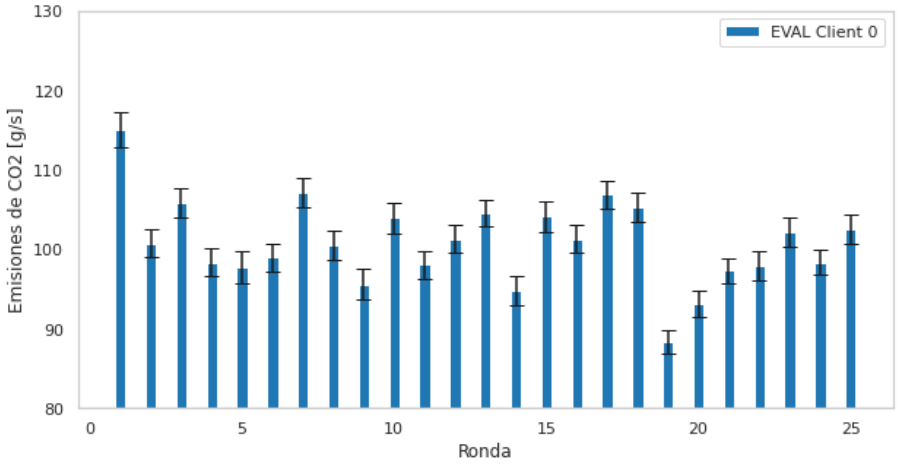


Figura 4.21: Evaluación de Emisiones CO_2 del Modelo Recibido en el Cliente 0

4.4.3. Evaluación del Modelo

Como se ha mencionado, el aprendizaje federado implementado utiliza el algoritmo de *Q-learning* en cada cliente. Para evaluar el desempeño del modelo, se analiza la tendencia de las recompensas a lo largo de las rondas, verificando que estas tiendan a estabilizarse. En la Figura 4.22 se muestra la recompensa promedio por ronda, donde se presentan tanto los datos reales como una regresión polinómica de los mismos. Se observa que, a partir de la ronda 30, las recompensas tienden a estabilizarse, lo que indica la convergencia del modelo.

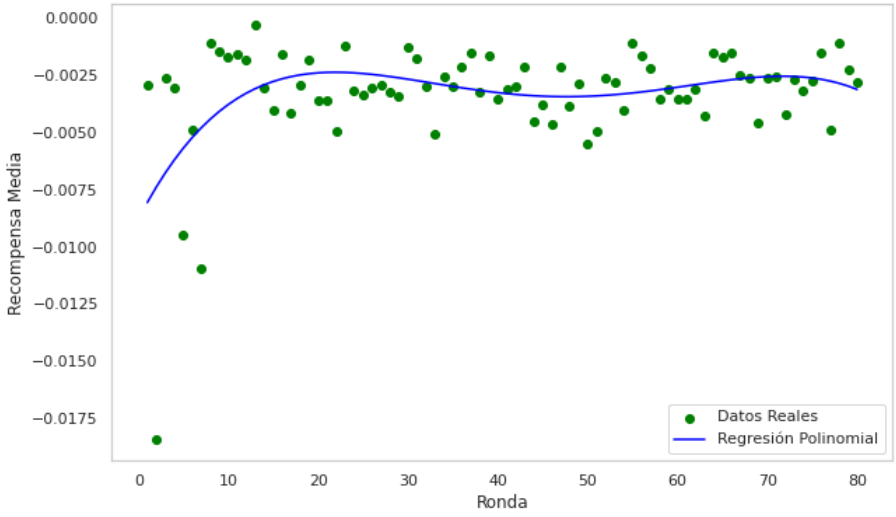


Figura 4.22: Recompensa Promedio por Ronda

En *Flower*, los clientes utilizan DP a través del módulo *LocalDpMod*. Desde el punto de vista de la privacidad, el uso de DP reduce de manera significativa la capacidad de inferir datos individuales de los usuarios al añadir un ruido controlado. Esto no solo proporciona una mayor protección de la información sensible, sino que también permite demostrar que el modelo está implementando efectivamente mecanismos de privacidad. La Figura 4.23 muestra una gráfica de la precisión de las recompensas para diferentes valores de ϵ, con el ruido añadido según la Ecuación 2.2. La precisión se calcula como la diferencia entre los datos obtenidos con DP y aquellos sin utilizar DP. La variación del parámetro ϵ revela una relación entre privacidad y precisión. Valores más bajos de ϵ mejoran la privacidad, pero reducen la precisión del modelo debido al mayor nivel de ruido añadido. En cambio, valores más altos de ϵ incrementan la precisión del modelo, aunque disminuye la privacidad. Los resultados previamente presentados se realizaron considerando un ϵ de 100, que como se muestra en la Figura 4.23 permite mantener una precisión superior al 85 % del modelo mientras se añade privacidad. Este ajuste del valor de ϵ permite modificar el nivel de privacidad según los requisitos específicos, equilibrando la

precisión del modelo con la protección de datos.

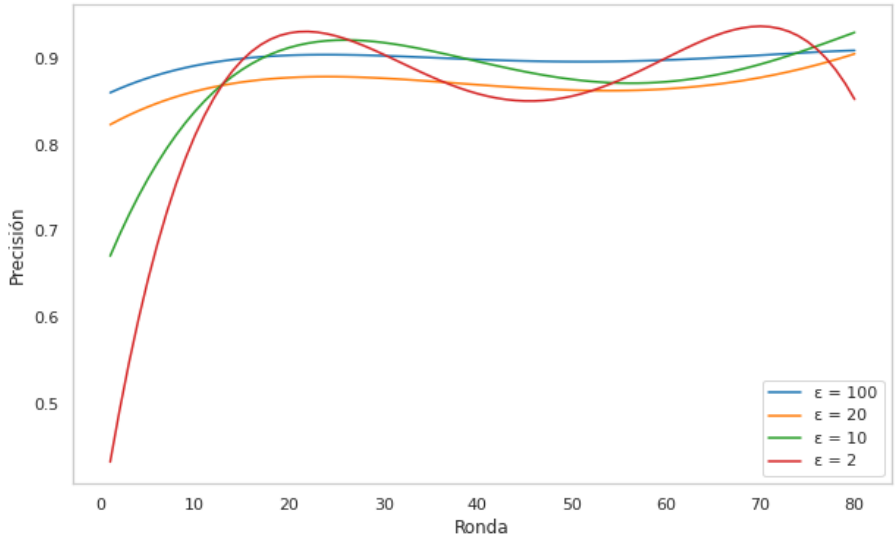


Figura 4.23: Precisión del Modelo para Distintos Valores de ϵ

4.5. Comparación Aprendizaje por Refuerzo vs Aprendizaje Federado

Para la comparación entre el Aprendizaje por Refuerzo y el Aprendizaje Federado se empezará evaluando el porcentaje de uso de CPU y el porcentaje de uso de la memoria RAM. Para esta comparación, se tendrá las condiciones iniciales mostradas en la Tabla 4.9.

Parámetros	Valor
Duración de la Simulación	86400
Densidad Vehicular	0.15
Número Episodios	5
Número de Rondas (Federado)	1
Runs	1

Tabla 4.9: Valores Utilizados para Realizar la Comparación

En las Figuras A.16 y A.17 se observa el comportamiento del porcentaje de uso del CPU y de la memoria RAM durante la simulación del RL y FL.

En la Figura 4.24, se presenta el promedio del porcentaje de uso de CPU y Memoria RAM para los clientes 0, 1 y 2, comparado con el RL, utilizando diagramas de bloques con un intervalo de confianza del 95 %. Este intervalo de confianza es pequeño, lo que indica que hay menos incertidumbre sobre el valor real del parámetro. Se observa que el uso promedio de CPU en los clientes es similar, con valores de 41.67 %, 41.26 % y 41.55 % para los clientes 0, 1 y 2 del

FL, respectivamente. En contraste, el RL muestra un uso promedio de CPU significativamente menor, alcanzando solo el 16.69 %. En cuanto al uso promedio de Memoria RAM, los clientes del FL presentan un valor uniforme de 47.25 %, mientras que el RL tiene un uso promedio inferior, de 38.73 %.

Comparando los dos tipos de aprendizaje, se observa que el FL presenta un mayor consumo de recursos que el RL, debido a la necesidad de realizar más procesos para el envío y recepción de nuevos parámetros entre los clientes y el servidor. La diferencia en el uso de CPU es considerable, con el RL utilizando un promedio de 16.69 % en comparación con el 41.49 % de los clientes del FL, lo que representa una diferencia de aproximadamente 24.8 puntos porcentuales. En cuanto al uso de Memoria RAM, el FL también consume más, con un incremento de aproximadamente 8.52 puntos porcentuales en comparación con el RL.

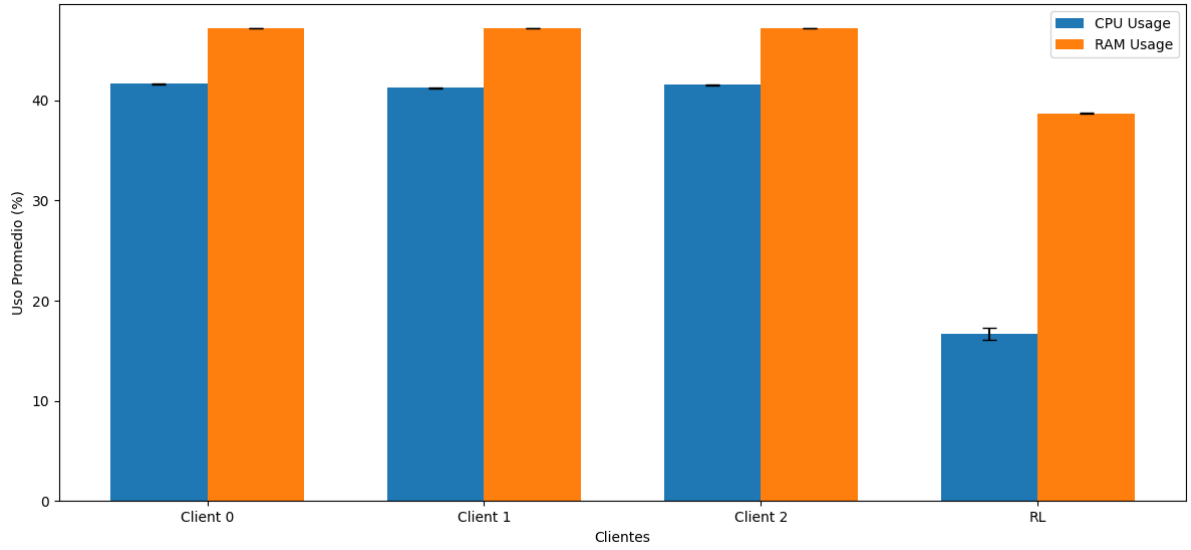


Figura 4.24: Uso Promedio de CPU y RAM

Otra manera en la que se compararon los diferentes métodos de aprendizaje fue mediante el tiempo de simulación. Los resultados de estos tiempos de simulación se presentan en la Tabla 4.10. Se observa que el tiempo de simulación del FL es casi cuatro veces mayor que el del RL, con una duración de 1797.66 segundos frente a los 468 segundos del RL. Esto indica que el aprendizaje federado es significativamente más lento y más demorado en comparación con el aprendizaje por refuerzo.

Estas diferencias en el tiempo de simulación pueden atribuirse a varios factores. En primer lugar, él FL involucra la comunicación y la agregación de modelos entre múltiples clientes y un servidor central, lo que añade una sobrecarga computacional y de comunicación. Además,

la heterogeneidad de los datos y modelos en los diferentes clientes puede llevar a mayores tiempos de convergencia.

Tipo de Aprendizaje	Tiempo [s]
RL	468
FL	1797.66

Tabla 4.10: Tiempo de Duración de cada Simulación

5. Conclusiones y Recomendaciones

En este capítulo se presentan las principales conclusiones y recomendaciones del estudio, el cual se enfocó en el diseño y evaluación de una arquitectura basada en técnicas de aprendizaje automático para mejorar la privacidad en los Sistemas de Transporte Inteligente (ITS). Los resultados obtenidos abordan la implementación de diversas técnicas de aprendizaje, incluyendo aprendizaje no supervisado, supervisado, por refuerzo y federado, con el objetivo de optimizar el tráfico vehicular y reducir las emisiones de CO_2 en la ciudad de Cuenca, Ecuador, mientras se protege la privacidad de los usuarios.

5.1. Conclusiones

Se recopiló información en la literatura científica acerca de los potenciales problemas a los que se enfrentan los ITS respecto a la privacidad de datos. Esta recopilación proporcionó una base sólida para entender las vulnerabilidades y desafíos que se deben abordar para proteger la privacidad de los usuarios en los ITS.

En el aprendizaje no supervisado, Se trabajó con 35 variables y 14,000 datos por cada una de ellas. La aplicación del Análisis de Componentes Principales (PCA) permitió identificar las componentes principales y las variables más importantes. Se compararon dos métodos de normalización: StandardScaler y MinMaxScaler. Con StandardScaler, se identificaron tres componentes principales que explicaron el 83.1 % de la varianza. Con MinMaxScaler, las tres componentes principales abarcaron el 86.5 % de la varianza. En ambos casos, se confirmó que tres componentes principales eran suficientes para una clasificación efectiva de los datos. La variable de *tiempo de espera total* se estableció como la base para la función de recompensa del algoritmo de Aprendizaje por Refuerzo (QL).

Con los datos transformados mediante PCA, se implementó el aprendizaje supervisado para clasificar el tiempo de espera y los niveles de CO_2 en categorías de alto, medio y bajo. Utilizando el algoritmo Random Forest, se determinó que la duración de la luz verde de los semáforos era una variable clave. La clasificación mostró una precisión general del 96 %, con una buena capacidad para predecir todas las clases, especialmente los niveles bajos y altos de congestión y emisiones de CO_2. Se observó que un tiempo de semáforo mínimo de 15 segundos era óptimo para reducir las emisiones y mejorar el flujo de tráfico.

En el Aprendizaje por Refuerzo con Q-Learning (QL), se configuró una función de recompensa

basada en el tiempo de espera total. La configuración inicial incluyó una tasa de aprendizaje (α) de 0.1 y un factor de descuento (γ) de 0.99. La simulación, que duró 86,400 segundos, permitió evaluar el comportamiento del sistema bajo diferentes densidades vehiculares (baja, media y alta). Los resultados mostraron una disminución en los tiempos de espera y emisiones de CO_2 en todos los escenarios, destacando la robustez del algoritmo.

El Aprendizaje Federado (FL) se implementó utilizando el Framework Flower, asegurando que los datos permanecieran en los dispositivos locales y solo se compartieran los parámetros del modelo. Se usó además el mecanismo de Privacidad Diferencial para añadir ruido en el envío de parámetros. Aunque el rendimiento del modelo FL fue inferior al del aprendizaje por refuerzo, se logró una mayor protección de la privacidad. Con cuatro clientes y un servidor central, cada cliente redujo los tiempos de espera y emisiones de CO_2 durante la fase de entrenamiento. En la fase de evaluación, el modelo general mantuvo la efectividad en la reducción de tiempos de espera y emisiones.

Al comparar RL y FL, se observó que FL tenía un mayor consumo de recursos. FL presentó picos de carga de CPU superiores al 60 % y un mayor uso de memoria RAM (47.10 %-47.50 %). En contraste, RL utilizó alrededor del 20 % del CPU y tuvo un uso de memoria más estable (38.6 %-39.1 %). Además, el tiempo de simulación de FL fue casi cuatro veces mayor que el de RL, indicando que el aprendizaje federado es más lento y demandante en términos de recursos.

Se demostró que es posible diseñar una arquitectura basada en técnicas de aprendizaje automático que mejora la privacidad de los usuarios en los ITS. Mediante la combinación de PCA, aprendizaje supervisado, por refuerzo y federado, se logró no solo optimizar la eficiencia del tráfico y reducir las emisiones de CO_2, sino también proteger la privacidad de los datos de los usuarios.

5.2. Recomendaciones

A continuación se presentan algunas recomendaciones que pueden ser consideradas al momento de mejorar la arquitectura propuesta.

- Evaluar y experimentar con otros métodos de normalización, además de $\mathrm{StandardScaler}$ y $\mathrm{MinMaxScaler}$, para determinar si se puede mejorar aún más la varianza explicada por las componentes principales.

- Considerar la optimización de los parámetros de simulación y aprendizaje en el marco de Flower para reducir el consumo de recursos y el tiempo de simulación. Esto podría incluir la experimentación con diferentes tasas de aprendizaje, factores de descuento, y configuraciones de red.
- Validar los resultados del estudio en entornos reales y con datos de tráfico en tiempo real para evaluar la efectividad del modelo en condiciones prácticas.
- Reforzar las medidas de seguridad de datos para proteger aún más la privacidad de los usuarios durante la implementación del aprendizaje federado, considerando la encriptación de los parámetros del modelo.

Referencias

[1] N. Alsaffar, H. Ali, y W. Elmedany, “Smart transportation system: a review of security and privacy issues,” in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. IEEE, 2018, pp. 1–4.

[2] IEEE, “Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Amendment 6: Wireless access in vehicular environments,” *IEEE Std. 802.11p-2010*, Jun. 2010.

[3] J. B. Kenney, “Dedicated short-range communications (dsrc) standards in the united states,” *Proceedings of the IEEE*, vol. 99, num. 7, pp. 1162–1182, 2011.

[4] M. Alaneed, “Enabling joint sensing and communication for vehicle-to-everything networks,” Ph.D. dissertation, Cleveland State University, 2023.

[5] A. L. Bazzan y F. Klügl, *Introduction to intelligent systems in traffic and transportation*. Springer Nature, 2022.

[6] Y. Wu, F. Gu, Y. Ji, S. Ma, y J. Guo, “Electric vehicle adoption and local pm2. 5 reduction: Evidence from china,” *Journal of Cleaner Production*, vol. 396, p. 136508, 2023.

[7] T. Baldi, G. Delnevo, R. Girau, y S. Mirri, “On the prediction of air quality within vehicles using outdoor air pollution: sensors and machine learning algorithms,” in *Proceedings of the ACM SIGCOMM Workshop on Networked Sensing Systems for a Sustainable Society*, 2022, pp. 14–19.

[8] D. A. Guastella y G. Bontempi, “Traffic modeling with sumo: a tutorial,” *arXiv preprint arXiv:2304.05982*, 2023.

[9] M. Behrisch, L. Bieker, J. Erdmann, y D. Krajzewicz, “Sumo–simulation of urban mobility: an overview,” in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[10] D. Gangwani y P. Gangwani, “Applications of machine learning and artificial intelligence in intelligent transportation system: A review,” *Applications of Artificial Intelligence and Machine Learning: Select Proceedings of ICAAAIML 2020*, pp. 203–216, 2021.

[11] S. Zhang, J. Li, L. Shi, M. Ding, D. C. Nguyen, W. Tan, J. Weng, y Z. Han, “Federated learning in intelligent transportation systems: Recent applications and open problems,”

IEEE Transactions on Intelligent Transportation Systems, vol. 25, num. 5, pp. 3259–3285, 2024.

[12] C. Tripp-Barba, P. Barbecho, L. Urquiza, y J. A. Aguilar-Calderón, “A comparison of vehicle emissions control strategies for smart cities,” *PeerJ Computer Science*, vol. 9, p. e1676, 2023.

[13] P. Barbecho Bautista, L. F. Urquiza-Aguiar, y M. Aguilar Igartua, “Privacy-aware vehicle emissions control system for traffic light intersections,” in *Proceedings of the 19th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, 2022, pp. 99–106.

[14] A. Sładkowski y W. Pamuła, *Intelligent transportation systems-problems and perspectives*. Springer, 2016, vol. 303.

[15] F. Cunha, G. Maia, C. Celes, D. Guidoni, F. de Souza, H. Ramos, y L. Villas, “Sistemas de transporte inteligentes: Conceitos, aplicações desafios,” *Livro de Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC’17)*, pp. 59–103, 2017.

[16] Movertis, “¿qué son los sistemas inteligentes de transporte?” 2024, accedido: 2024-06-23. [En línea]. Disponible: https://www.movertis.com/blog/sistemas-inteligentes-de-transporte

[17] M. Ayala-Chauvin, F. Avilés-Castillo, y J. Buele, “Exploring the landscape of data analysis: a review of its application and impact in ecuador,” *Computers*, vol. 12, num. 7, p. 146, 2023.

[18] A. Estevan, “Modelos de transporte y emisiones de co2 en españa,” *Revista de Economía Crítica*, num. 4, pp. 67–87, 2005.

[19] V. R. Bedoya, O. M. Sardà, y C. M. i Guasch, “Estimación de las emisiones de co2 desde la perspectiva de la demanda de transporte en medellín,” *Transporte y Territorio*, num. 15, pp. 302–322, 2016.

[20] Á. M. F. ARISTIZÁBAL, “06. aprendizaje automático,” *Coloquio de Investigación Formativa*, p. 32, 2021.

[21] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons. b*, vol. 4, num. 51-62, p. 56, 2017.

[22] A. S. Agama Espinoza, “Revisión de la literatura del comercio electrónico, el aprendizaje automático y sus aplicaciones en la industria y tiendas por departamento en línea,” 2021.

[23] J. Jiménez-Martínez y L. Mejía, “Multivariate stochastic analysis co2 emission factor for carbon sequestration and sustainable development for colombia. ugciencia. 2014; 20: 64–71.”

[24] L. P. Kaelbling, M. L. Littman, y A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[25] S. Naeem, A. Ali, S. Anam, y M. M. Ahmed, “An unsupervised machine learning algorithms: Comprehensive review,” *International Journal of Computing and Digital Systems*, 2023.

[26] M. Ahmed, R. Seraj, y S. M. S. Islam, “The k-means algorithm: A comprehensive survey and performance evaluation,” *Electronics*, vol. 9, num. 8, p. 1295, 2020.

[27] L. KPFRS, “On lines and planes of closest fit to systems of points in space,” in *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (SIGMOD)*, 1901, p. 19.

[28] H. Hotelling, “Analysis of a complex of statistical variables into principal components.” *Journal of educational psychology*, vol. 24, num. 6, p. 417, 1933.

[29] Ó. M. Molina y E. De los Monteros Pérez, “Rotación en análisis de componentes principales categórico: un caso práctico,” *Metodología de encuestas*, vol. 12, num. 1, pp. 63–88, 2010.

[30] I. Muhammad y Z. Yan, “Supervised machine learning approaches: A survey.” *ICTACT Journal on Soft Computing*, vol. 5, num. 3, 2015.

[31] W. S. Noble, “What is a support vector machine?” *Nature biotechnology*, vol. 24, num. 12, pp. 1565–1567, 2006.

[32] A. Natekin y A. Knoll, “Gradient boosting machines, a tutorial,” *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[33] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, y S. Homayouni, “Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 6308–6325, 2020.

[34] M. Chi y L. Bruzzone, “Semisupervised classification of hyperspectral images by svms optimized in the primal,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, num. 6, pp. 1870–1880, 2007.

[35] P. O. Gislason, J. A. Benediktsson, y J. R. Sveinsson, “Random forests for land cover classification,” *Pattern recognition letters*, vol. 27, num. 4, pp. 294–300, 2006.

[36] H. Byeon, “Advances in value-based, policy-based, and deep learning-based reinforcement learning,” *International Journal of Advanced Computer Science and Applications*, vol. 14, num. 8, 2023.

[37] M. M. Alam y S. Moh, “Survey on q-learning-based position-aware routing protocols in flying ad hoc networks,” *Electronics*, vol. 11, num. 7, p. 1099, 2022.

[38] M. Al-Emran, “Hierarchical reinforcement learning: a survey,” *International journal of computing and digital systems*, vol. 4, num. 02, 2015.

[39] P. Paavai Anand *y otros*, “A brief study of deep reinforcement learning with epsilon-greedy exploration,” *International Journal Of Computing and Digital System*, 2021.

[40] R. S. Sutton y A. G. Barto, *Reinforcement learning. An introduction*, 2da ed., ser. Adapt. Comput. Mach. Learn. Cambridge, MA: MIT Press, 2018.

[41] Y. Liu, B. Cao, y H. Li, “Improving ant colony optimization algorithm with epsilon greedy and levy flight,” *Complex & Intelligent Systems*, vol. 7, num. 4, pp. 1711–1722, 2021.

[42] J. Qi, Q. Zhou, L. Lei, y K. Zheng, “Federated reinforcement learning: Techniques, applications, and open challenges,” *arXiv preprint arXiv:2108.11887*, 2021.

[43] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, y N. D. Lane, “Flower: A friendly federated learning research framework,” 2022. [En línea]. Disponible: https://arxiv.org/abs/2007.14390

[44] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, y H. V. Poor, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE transactions on information forensics and security*, vol. 15, pp. 3454–3469, 2020.

[45] R. Hu, Y. Guo, H. Li, Q. Pei, y Y. Gong, “Personalized federated learning with differential privacy,” *IEEE Internet of Things Journal*, vol. 7, num. 10, pp. 9530–9539, 2020.

[46] J.-S. Kim, “Design of federated learning engagement method for autonomous vehicle privacy protection,” in *2022 Joint 12th International Conference on Soft Computing and*

Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS). IEEE, 2022, pp. 1–2.

[47] S. R. Pokhrel y J. Choi, “A decentralized federated learning approach for connected autonomous vehicles,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2020, pp. 1–6.

[48] Y. Lu, X. Huang, K. Zhang, S. Maharjan, y Y. Zhang, “Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles,” *IEEE Transactions on Vehicular Technology*, vol. 69, num. 4, pp. 4298–4311, 2020.

[49] Z. Qu, Y. Tang, G. Muhammad, y P. Tiwari, “Privacy protection in intelligent vehicle networking: A novel federated learning algorithm based on information fusion,” *Information Fusion*, vol. 98, p. 101824, 2023.

[50] R. S. K. Boddu, R. R. Chandan, M. Thamizharasi, R. Shaikh, A. A. Goyal, P. P. Gupta, y S. K. Gupta, “Using deep learning to address the security issue in intelligent transportation systems,” *Journal of Autonomous Intelligence*, vol. 7, num. 4, Mar. 2024.

[51] G. Meena, D. Sharma, y M. Mahrishi, “Traffic prediction for intelligent transportation system using machine learning,” in *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*. IEEE, 2020, pp. 145–148.

[52] T. Liu, F. Sabrina, J. Jang-Jaccard, W. Xu, y Y. Wei, “Artificial intelligence-enabled ddos detection for blockchain-based smart transport systems,” *Sensors*, vol. 22, num. 1, p. 32, Dic. 2021.

[53] Z. Lv, S. Zhang, y W. Xiu, “Solving the security problem of intelligent transportation system with deep learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, num. 7, pp. 4281–4290, 2020.

[54] K. Y. Anmol y M. S., “Role of cyber security on intelligent transportation systems,” *Insights2Techinfo*, p. 1, 2024.

[55] S. M. A. A. P. W. P. W. A. S. Vasudevan M, Townsend H y I. McManus, “Ai for its challenges and lessons learned report,” *Modern Education Forum*, vol. 1, num. 1, 2023.

[56] L. S. Iyer, “Ai enabled applications towards intelligent transportation,” *Transportation Engineering*, vol. 5, p. 100083, 2021.

[57] J. G. Bijalwan, J. Singh, V. Ravi, A. Bijalwan, T. J. Alahmadi, P. Singh, y M. Diwakar, “Navigating the future of secure and efficient intelligent transportation systems using ai and blockchain,” *The Open Transportation Journal*, vol. 18, num. 1, Mar. 2024.

[58] DLR, “Osm web wizard — eclipse sumo documentation,” https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html, 2023, accessed: 2024-07-16.

[59] L. N. Alegre, “SUMO-RL,” https://github.com/LucasAlegre/sumo-rl, 2019.

[60] B. McMahan, E. Moore, D. Ramage, S. Hampson, y B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[61] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, y P. Zhokhov, “Openai baselines,” https://github.com/openai/baselines, 2017.

[62] J. R. Beattie y F. W. Esmonde-White, “Exploration of principal component analysis: deriving principal component analysis visually using spectra,” *Applied Spectroscopy*, vol. 75, num. 4, pp. 361–375, 2021.

[63] D. Santamaría Álvarez, “Detección de ataques de tipo exploit en redes iot empleando algoritmos de clasificación de machine learning,” 2024.

A. Anexo

A.A. Algoritmo FedAvg

En este algoritmo el servidor inicializa el modelo global w_0 (línea 2). Este modelo se utiliza como punto de partida para las iteraciones de entrenamiento.

Para cada ronda t de entrenamiento, que va desde 1 hasta el número deseado de rondas (línea 3), se determina el número de clientes a seleccionar en esta ronda, m, que es el máximo entre $C \cdot K$ y 1, donde C es la fracción de clientes y K el número total de clientes (línea 4). Luego, se selecciona un conjunto aleatorio de m clientes, S_t (línea 5).

Para cada cliente k en el conjunto S_t(línea 6), se ejecuta la función *EV_ClientUpdate(k, w_t)* para actualizar su modelo local w_k^{t+1} (línea 7). Es decir, cada cliente seleccionado actualiza su modelo local utilizando su propio conjunto de datos.

Después de que todos los clientes seleccionados han actualizado sus modelos locales, el servidor actualiza el modelo global w_{t+1} promediando los modelos locales ponderados por el número de datos de cada cliente n_k (línea 9).

La función *EV_ClientUpdate(k, w)* se ejecuta en el cliente k y actualiza el modelo local w (línea 12). Primero, el conjunto de datos local P_k se divide en minibatches de tamaño B (línea 13). Luego, para cada epoch local i, desde 1 hasta E (línea 14), y para cada minibatch b en $\mathcal{B}$ (línea 15), se actualiza el modelo local w utilizando el gradiente de la pérdida respecto a w en el minibatch b, estableciendo una tasa de aprendizaje η (línea 16).

Una vez completadas las actualizaciones locales, la función devuelve el modelo local actualizado w al servidor (línea 19), completando así la ronda de entrenamiento. Este proceso se repite hasta cumplir con las rondas de entrenamiento establecidas.

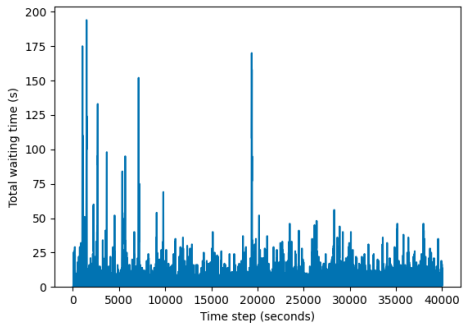
Algorithm 1 Los K clientes están indexados por k; B es el tamaño del minibatch local, E es el número de episodios locales, y η es la tasa de aprendizaje [60].

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
    $m \leftarrow$ máx$(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
        $w_{t+1}^k \leftarrow$ EV_ClientUpdate(k, w_t)
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \cdot w_{t+1}^k$
end for

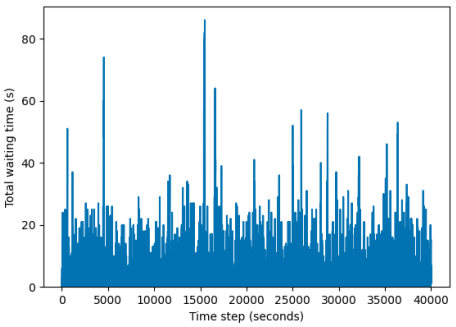
function EV_ClientUpdate(k, w) ▷ Run on client k
    $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches size B)
    for each local epoch i from 1 to E **do**
        for each batch $b \in \mathcal{B}$ **do**
            $w \leftarrow w - \eta \nabla l(w; b)$
        end for
    end for
    return w to server
end function

A.B. Monitoreo de Convergencia

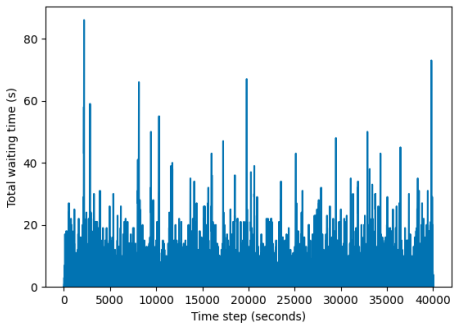
A.B.A. Densidad Baja



(a) Episodio 1



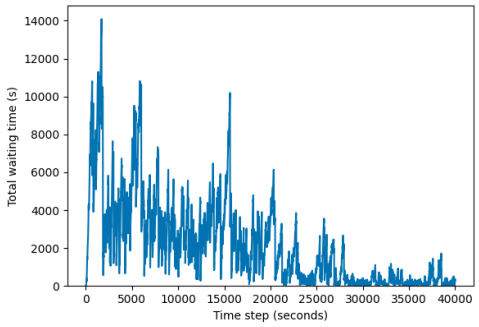
(b) Episodio 2



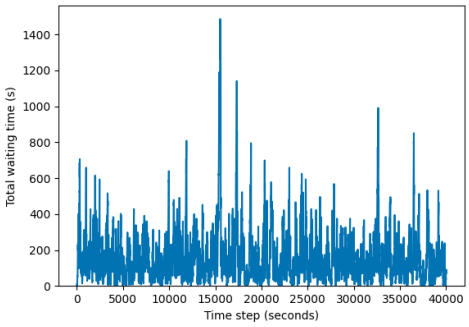
(c) Episodio 3

Figura A.1: Episodios Realizados para Monitorear Convergencia

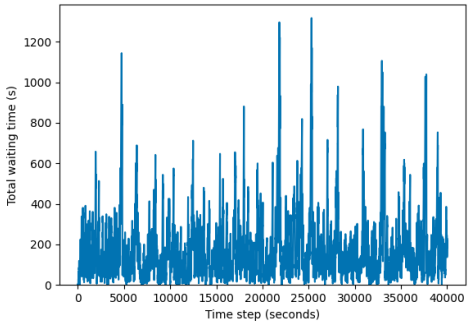
A.B.B. Densidad Media



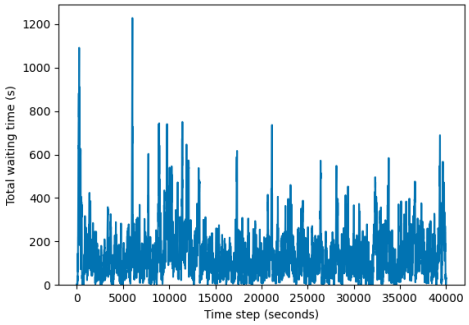
(a) Episodio 1



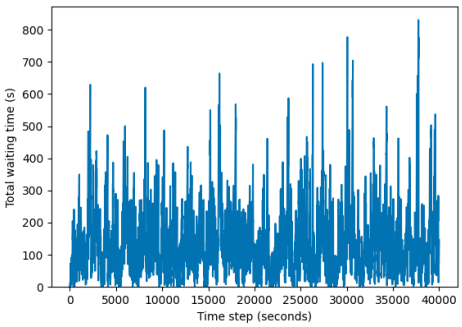
(b) Episodio 2



(c) Episodio 3



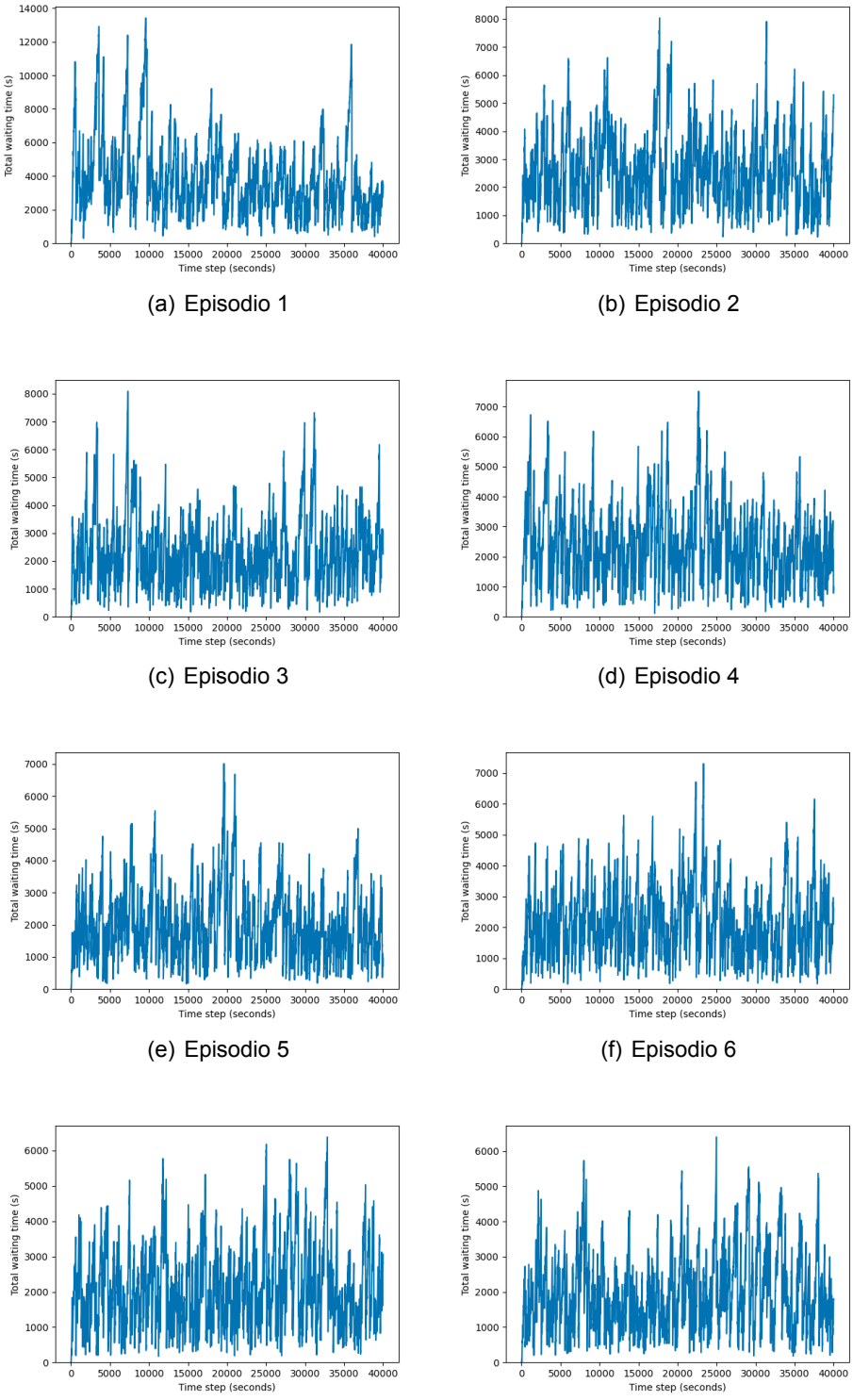
(d) Episodio 4



(e) Episodio 5

Figura A.2: Episodios Realizados para Monitorear Convergencia

A.B.C. Densidad Alta



(a) Episodio 1 (b) Episodio 2

(c) Episodio 3 (d) Episodio 4

(e) Episodio 5 (f) Episodio 6

(g) Episodio 7 (h) Episodio 8

Figura A.3: Episodios Realizados para Monitorear Convergencia

A.C. Clientes Aprendizaje Federado

A.C.A. Entrenamiento

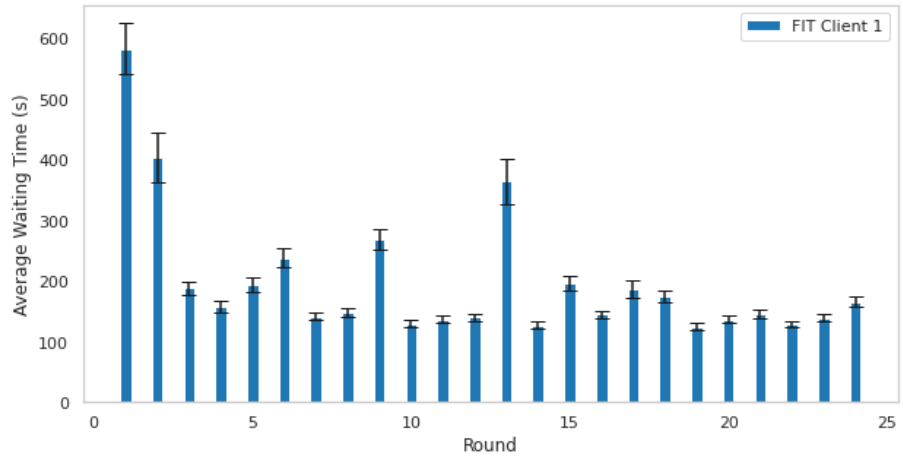


Figura A.4: Resultados Obtenidos de Tiempo de Espera en cada Entrenamiento Cliente 1

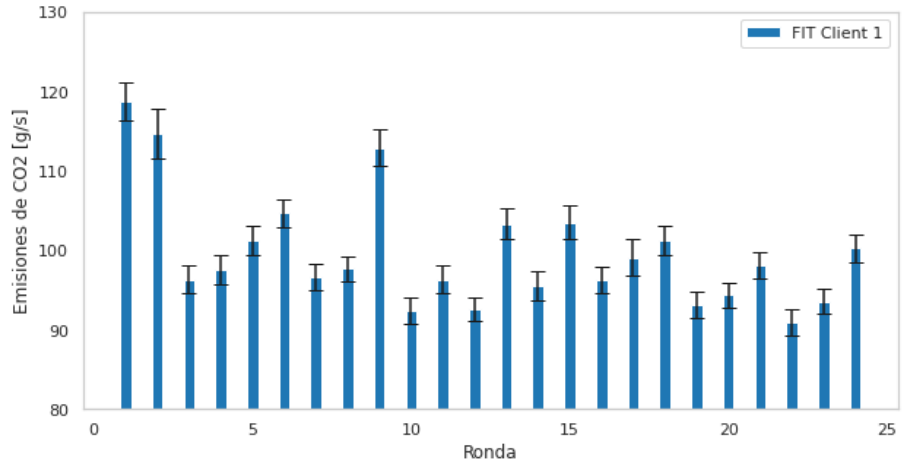


Figura A.5: Resultados Obtenidos de Emisiones de CO_2 en cada Entrenamiento Cliente 1

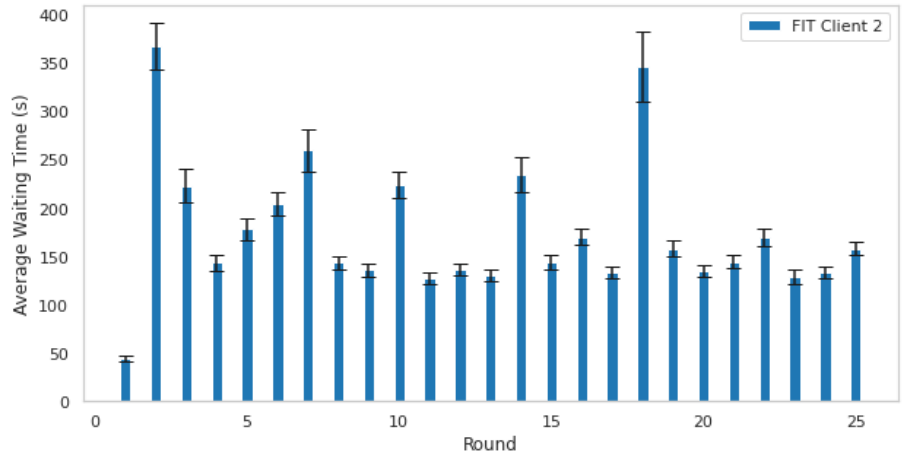


Figura A.6: Resultados Obtenidos de Tiempo de Espera en cada Entrenamiento Cliente 2

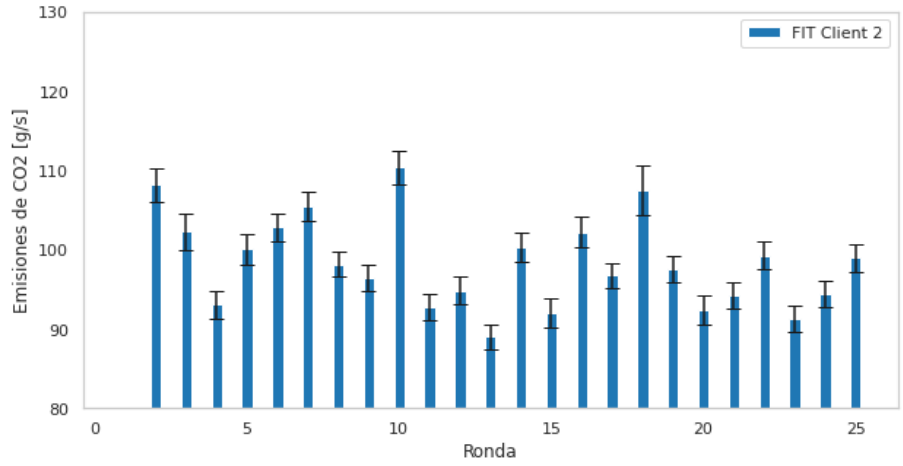


Figura A.7: Resultados Obtenidos de Emisiones de CO_2 en cada Entrenamiento Cliente 2

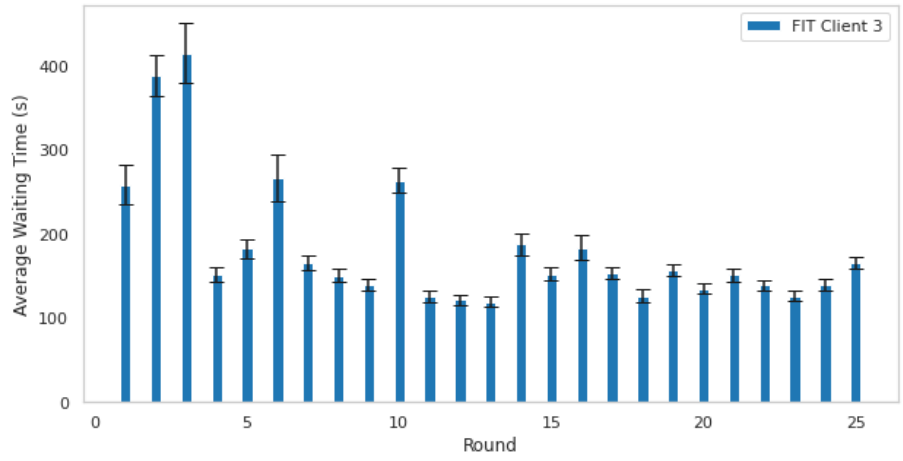


Figura A.8: Resultados Obtenidos de Tiempo de Espera en cada Entrenamiento Cliente 3

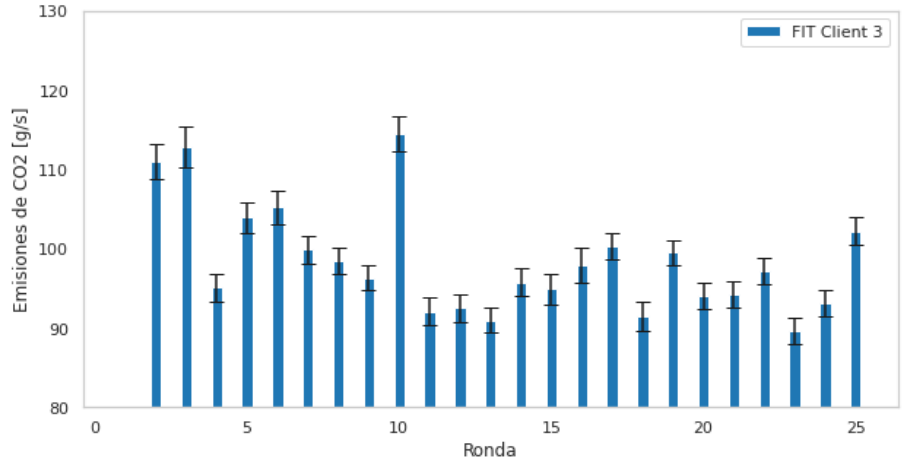


Figura A.9: Resultados Obtenidos de Emisiones de CO_2 en cada Entrenamiento Cliente 3

A.C.B. Evaluación

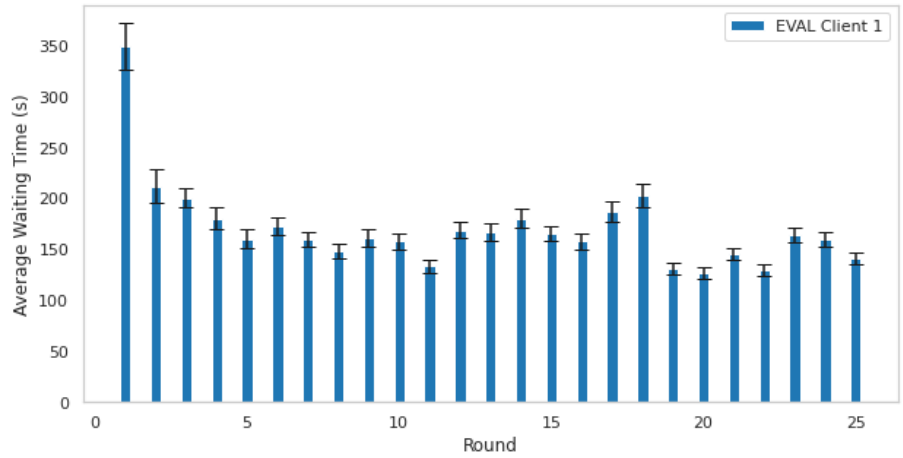


Figura A.10: Evaluación de Tiempo de Espera del Modelo Recibido en el Cliente 1

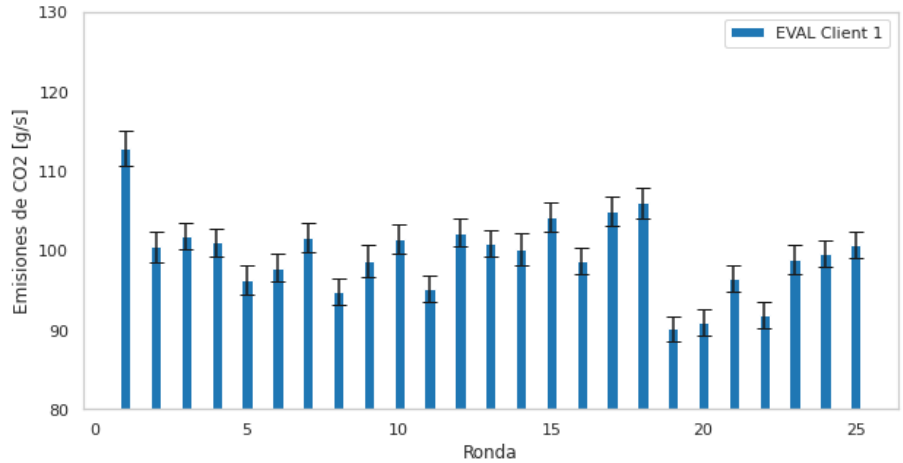


Figura A.11: Evaluación de Emisiones de CO_2 del Modelo Recibido en el Cliente 1

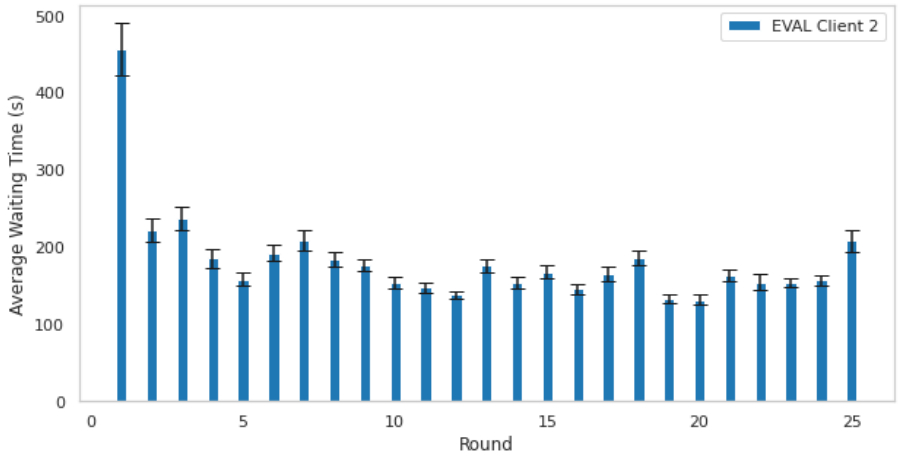


Figura A.12: Evaluación de Tiempo de Espera del Modelo Recibido en el Cliente 2

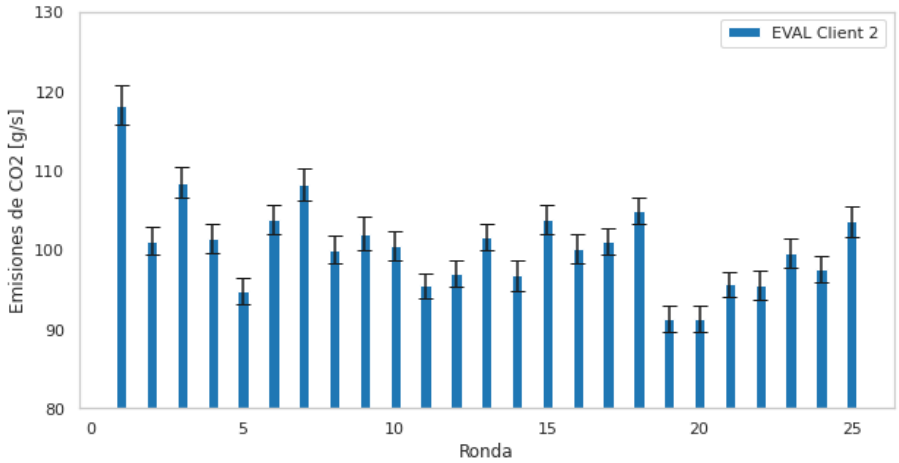


Figura A.13: Evaluación de Emisiones de CO_2 del Modelo Recibido en el Cliente 2

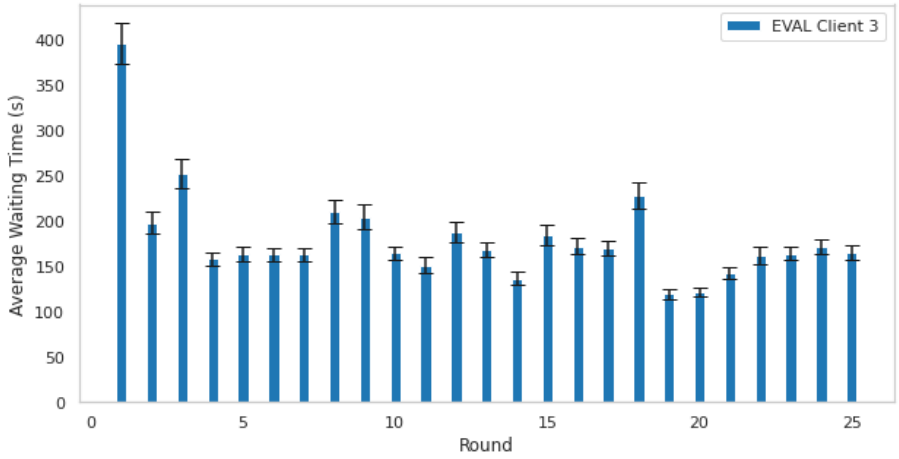


Figura A.14: Evaluación de Tiempo de Espera del Modelo Recibido en el Cliente 3

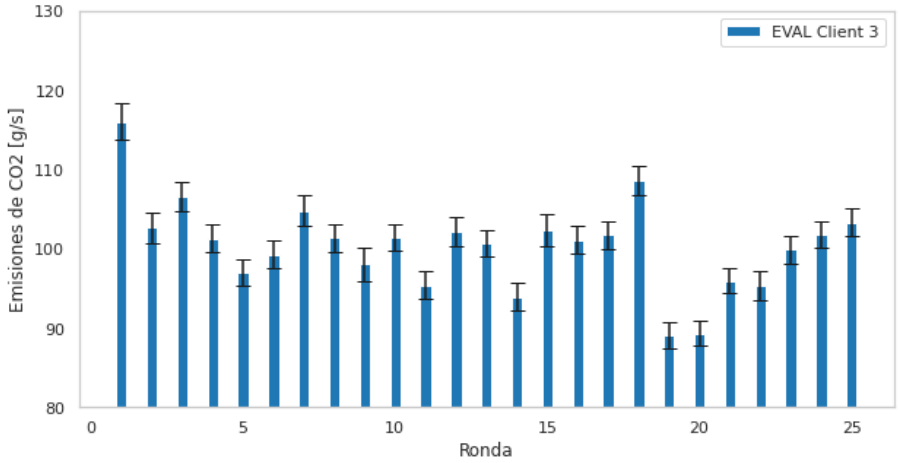


Figura A.15: Evaluación de Emisiones de CO_2 del Modelo Recibido en el Cliente 3

A.D. Comparación Aprendizaje por Refuerzo vs Aprendizaje Federado

Se presentan en las Figuras A.16 y A.17 el comportamiento del CPU y de la Memoria RAM a lo largo del tiempo de simulación para los casos de Aprendizaje por Refuerzo y Aprendizaje Federado.

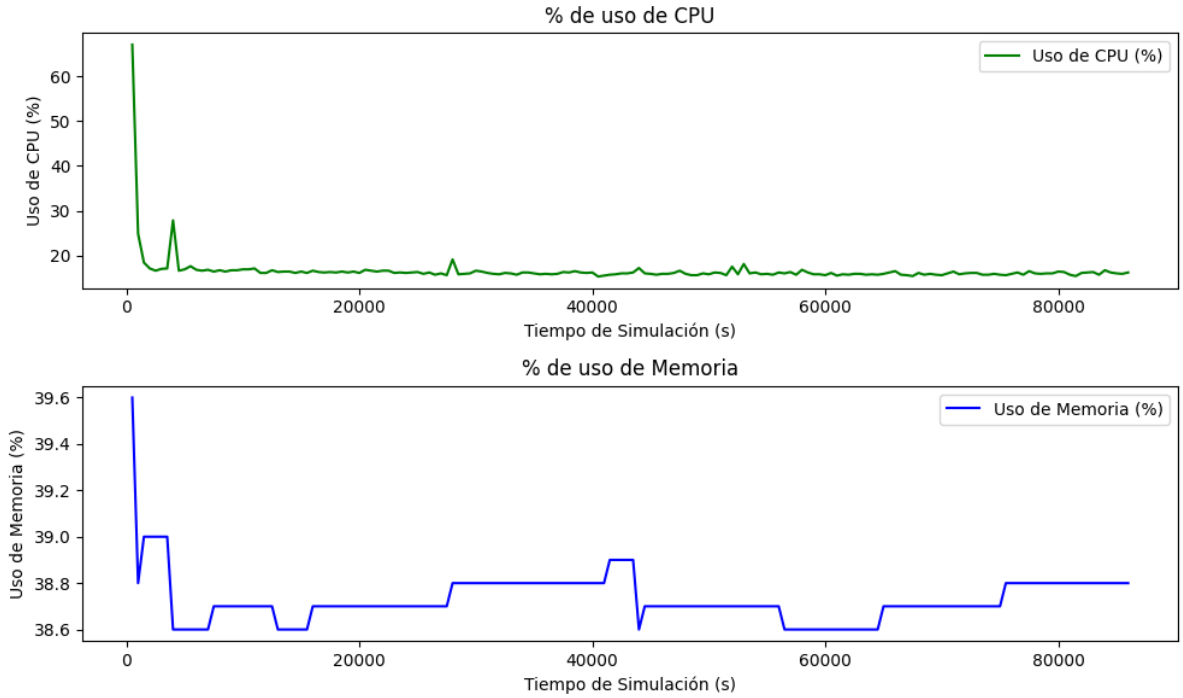


Figura A.16: Porcentaje de Uso de CPU y de Memoria RAM con RL

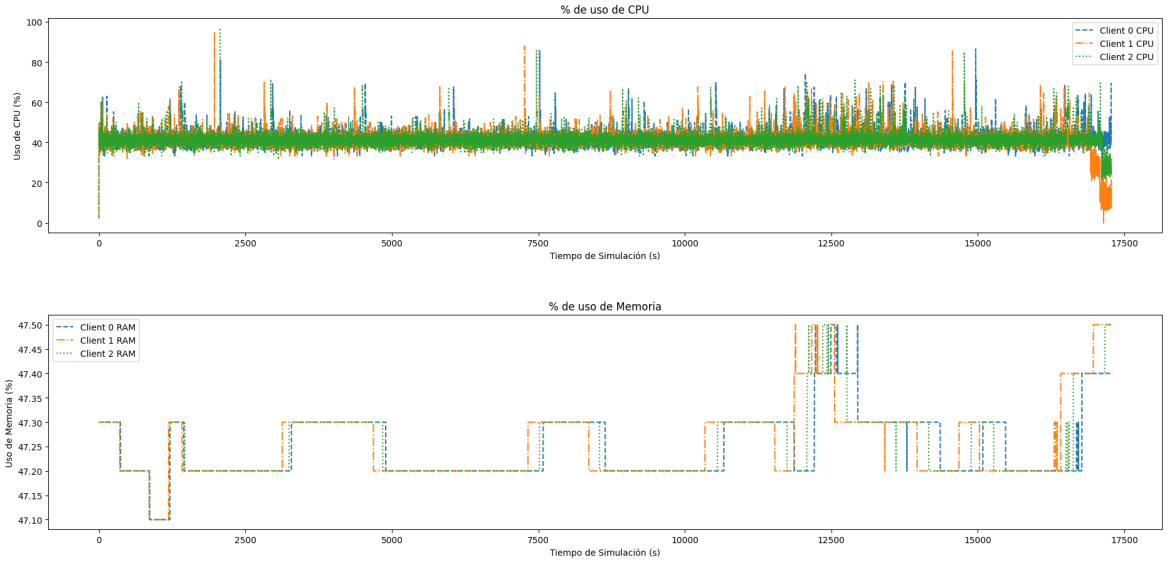


Figura A.17: Porcentaje de Uso de CPU y de Memoria RAM con FL

A.E. Repositorio

Los algoritmos desarrollados en el presente trabajo de titulación se encuentran disponibles en: https://github.com/davidsgonza/Privacy-V2X