

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



**Estimación de tiempos de llegada de Transmetro mediante  
modelos de aprendizaje automático y visualización interactiva**

Trabajo de graduación presentado por Pablo Andrés Zamora Vásquez  
para optar al grado académico de Licenciado en Ingeniería en Ciencias  
de la Computación y Tecnologías de la Información

Guatemala,

2025







UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



**Estimación de tiempos de llegada de Transmetro mediante  
modelos de aprendizaje automático y visualización interactiva**

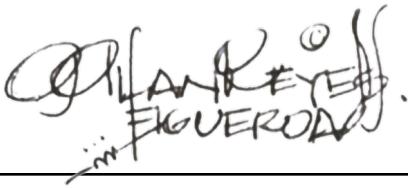
Trabajo de graduación presentado por Pablo Andrés Zamora Vásquez  
para optar al grado académico de Licenciado en Ingeniería en Ciencias  
de la Computación y Tecnologías de la Información

Guatemala,

2025

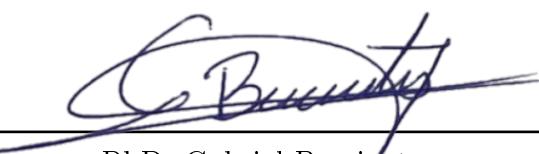


Vo.Bo.:

(f)   
PhD. Alan Reyes

Tribunal Examinador:

(f)   
PhD. Alan Reyes

(f)   
PhD. Gabriel Barrientos

Fecha de aprobación: Guatemala, noviembre de 2025.



---

## Dedicatoria

---

A Dios y a la Virgen María, por ser mi guía y fortaleza en cada uno de mis pasos.

A mi bisabuela Marieta, a mi abuela Chary, a mi madre Claudia y a mi hermana María José, por su amor incondicional, apoyo constante y por ser mi inspiración diaria para alcanzar mis metas.

A mis amigos y compañeros tanto dentro como fuera de la universidad, por su compañía, motivación y por compartir conmigo tantos momentos inolvidables.

A todos mis profesores y mentores a lo largo de mi formación académica, por su dedicación, paciencia y por transmitirme el conocimiento que me ha permitido crecer tanto personal como profesionalmente.

A la Fundación Juan Bautista Gutiérrez, por brindarme la oportunidad de acceder a una educación de calidad y colocar su confianza en mí a través de la beca que me fue otorgada.

A todos ustedes, gracias por ser parte de mi vida y de este logro tan importante para mí.



---

## Agradecimientos

---

A Alejandra Paiz y a la Empresa Municipal de Transporte, especialmente a Álvaro Pú, por su colaboración al proporcionar acceso a los registros históricos de posiciones GPS de las unidades de Transmetro.

En especial, a mi asesor, PhD. Alan Reyes, y a mi supervisor de trabajo de graduación, PhD. Gabriel Barrientos, por su guía, apoyo y paciencia a lo largo de todo el desarrollo de este trabajo.



---

## Índice

---

<b>Dedicatoria</b>	<b>V</b>
<b>Agradecimientos</b>	<b>VII</b>
<b>Lista de figuras</b>	<b>XIV</b>
<b>Lista de cuadros</b>	<b>XV</b>
<b>Resumen</b>	<b>XVII</b>
<b>Abstract</b>	<b>XIX</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Justificación</b>	<b>3</b>
<b>3. Objetivos</b>	<b>5</b>
3.1. Objetivo general . . . . .	5
3.2. Objetivos específicos . . . . .	5
<b>4. Alcance</b>	<b>7</b>
<b>5. Marco teórico</b>	<b>9</b>
5.1. Movilidad Urbana . . . . .	9
5.1.1. Movilidad urbana sostenible . . . . .	9
5.1.2. Transporte Público y Autobús de Tránsito Rápido . . . . .	9
5.1.3. Movilidad urbana en Guatemala . . . . .	10
5.2. Sistema de Transmetro . . . . .	11
5.3. Tiempo estimado de llegada . . . . .	13
5.4. Sistemas de información geográfica . . . . .	13
5.4.1. Cálculo de distancias geodésicas . . . . .	13
5.4.2. Proyecciones cartográficas locales . . . . .	14
5.4.3. Estructuración de rutas mediante polilíneas . . . . .	14
5.4.4. Proyección punto-segmento . . . . .	15

5.5.	Modelos de Markov ocultos . . . . .	15
5.5.1.	Algoritmo de Viterbi . . . . .	16
5.6.	Modelos de aprendizaje de gradiente . . . . .	17
5.6.1.	XGBoost . . . . .	17
5.6.2.	LightGBM . . . . .	17
5.7.	Evaluación y métricas . . . . .	18
5.7.1.	Métricas de error en regresión . . . . .	18
5.7.2.	Validación cruzada . . . . .	18
5.8.	Arquitectura de integración de modelos . . . . .	19
5.8.1.	Backend . . . . .	19
5.8.2.	Frontend . . . . .	19
<b>6.</b>	<b>Metodología</b>	<b>21</b>
6.1.	Recolección de datos . . . . .	21
6.1.1.	Automatización de la recolección de datos . . . . .	21
6.2.	Limpieza de datos . . . . .	22
6.2.1.	Unificación de informes . . . . .	22
6.2.2.	Transformaciones . . . . .	23
6.2.3.	Filtrado de trayectorias útiles . . . . .	23
6.3.	Análisis exploratorio de datos . . . . .	26
6.3.1.	Viajes multilínea . . . . .	26
6.3.2.	Tiempo de permanencia en estaciones . . . . .	27
6.3.3.	Consistencia temporal de los registros . . . . .	28
6.4.	Ingeniería de características . . . . .	28
6.4.1.	Línea en la que está operando la unidad . . . . .	28
6.4.2.	Siguiente estación teórica y distancia restante . . . . .	35
6.4.3.	Tiempo de permanencia . . . . .	37
6.4.4.	Velocidad instantánea . . . . .	37
6.4.5.	Características derivadas de tiempo . . . . .	38
6.5.	Cálculo de la variable objetivo . . . . .	38
6.6.	Descarte de datos atípicos . . . . .	39
6.7.	Muestreo representativo del conjunto de datos . . . . .	40
6.8.	Entrenamiento de modelos: LightGBM y XGBoost . . . . .	42
6.8.1.	Partición de datos para entrenamiento y validación . . . . .	42
6.8.2.	Validación cruzada temporal . . . . .	42
6.8.3.	Métricas de evaluación . . . . .	44
6.8.4.	Entrenamiento con LightGBM . . . . .	44
6.8.5.	Entrenamiento con XGBoost . . . . .	44
6.8.6.	Selección de número de iteraciones para los modelos finales . . . . .	45
6.9.	Complementación de los modelos con estadísticas históricas . . . . .	45
6.10.	Backend para interacción con los modelos de predicción . . . . .	47
6.10.1.	Archivos de información estática . . . . .	48
6.10.2.	Uso de modelos entrenados . . . . .	48
6.10.3.	Obtención de datos más recientes: . . . . .	49
6.10.4.	Definición de secuencia de estaciones siguientes por unidad . . . . .	49
6.10.5.	Cálculo de tiempos de llegada acumulados para una secuencia de estaciones . . . . .	51
6.10.6.	Cálculo de ETA por estación y selección del mejor candidato . . . . .	51

6.10.7. Cálculo del tiempo estimado de duración de un viaje entre estaciones . . . . .	52
6.10.8. Inicialización de la aplicación FastAPI y exposición de endpoints . . . . .	52
6.11. Frontend para interacción con los modelos de predicción . . . . .	55
6.11.1. Prototipado de vistas . . . . .	55
6.11.2. Implementación del frontend . . . . .	55
<b>7. Resultados</b>	<b>57</b>
7.1. Resultados de la recolección y limpieza de datos . . . . .	57
7.2. Resultados de la ingeniería de características . . . . .	58
7.3. Resultados del entrenamiento de modelos . . . . .	62
7.3.1. LightGBM . . . . .	62
7.3.2. XGBoost . . . . .	64
7.3.3. Resultados del cálculo de tiempos de viaje históricos entre estaciones . . . . .	67
7.4. Resultados de la implementación de los modelos . . . . .	67
<b>8. Discusión</b>	<b>71</b>
<b>9. Conclusiones</b>	<b>77</b>
<b>10. Recomendaciones</b>	<b>79</b>
<b>11. Bibliografía</b>	<b>81</b>
<b>12. Anexos</b>	<b>85</b>
12.1. Polilínea generada para cada línea de Transmetro . . . . .	85
12.2. Prototipos realizados en Figma . . . . .	92
12.3. Repositorios del proyecto . . . . .	92



---

## Lista de figuras

---

1.	Mapa oficial del sistema Transmetro de la Ciudad de Guatemala. Fuente: Municipalidad de Guatemala [21] . . . . .	12
2.	Esquema de validación cruzada temporal para evaluación de modelos predictivos. . . . .	19
3.	Ejemplo del formato original de los informes descargados. . . . .	22
4.	Ejemplo del formato de los <i>dataframes</i> una vez aplicadas las transformaciones. . . . .	23
5.	Histograma de velocidades de los registros de posición de la unidad 049. . . . .	24
6.	Ejemplo de las estaciones georreferenciadas. . . . .	25
7.	Ejemplo de la serie de estaciones diarias visitadas por la unidad 049. . . . .	26
8.	Comparación del viaje 146 de la unidad 204 con el mapa de las líneas 2 y 6 de Transmetro. . . . .	27
9.	Tiempos de permanencia de la unidad 098 en la línea 18 - Atlántida - Paraíso. . . . .	27
10.	Diferencia de tiempo entre registros consecutivos para la unidad 098. . . . .	28
11.	Ejemplo de estaciones de diferentes líneas de Transmetro en trayectorias similares. . . . .	29
12.	Asignación de puntos GPS de un viaje de la unidad 049 a la polilínea holgada de la línea 12. . . . .	30
13.	Segmentos de la polilínea densa de la línea 7 que representan el mismo trayecto pero en sentido opuesto. . . . .	31
14.	Estación terminal con giro en U. . . . .	31
15.	Ejemplo de estaciones únicas de la línea 6 de Transmetro. Fuente: Municipalidad de Guatemala [21]. . . . .	32
16.	Mapas de registros en los que la unidad 204 no realizó movimientos que reflejan su comportamiento operativo. . . . .	40
17.	Ejemplo de respuesta del <i>endpoint GET /stations</i> . . . . .	53
18.	Ejemplo de respuesta del <i>endpoint POST /eta</i> . . . . .	54
19.	Diagrama simplificado del <i>backend</i> y sus interacciones. . . . .	55
20.	Mapa de las trayectorias no útiles para las unidades 049, 401 y 216, respectivamente. . . . .	57

21. Comparación de una de las trayectorias útiles de la unidad 049 con el mapa de la línea 12 de Transmetro en la página oficial de la Municipalidad de Guatemala [21]. . . . .	58
22. Comparación del viaje 1 de la unidad 113 con el mapa de la línea 18 de Transmetro. . . . .	58
23. Puntos GPS consecutivos del viaje 1 de la unidad 113 con las estaciones teóricas inferidas. . . . .	60
24. Matriz de correlación entre las características numéricas generadas y la variable objetivo <code>ETA_proxima_est_s</code> . . . . .	61
25. Distribución de la variable objetivo <code>ETA_proxima_est_s</code> en el conjunto de datos final. . . . .	62
26. Curva de pérdida del modelo LightGBM durante el entrenamiento del modelo final. . . . .	63
27. Desempeño del modelo LightGBM por línea y dirección (MAE). . . . .	63
28. Desempeño del modelo LightGBM por línea y dirección (RMSE). . . . .	64
29. Importancia de características en el modelo LightGBM. . . . .	64
30. Curva de pérdida del modelo XGBoost durante el entrenamiento del modelo final. . . . .	65
31. Desempeño del modelo XGBoost por línea y dirección (MAE). . . . .	66
32. Desempeño del modelo XGBoost por línea y dirección (RMSE). . . . .	66
33. Importancia de características en el modelo XGBoost. . . . .	67
34. Obtención de candidatos para la estación “MONTÚFAR”, línea 13, dirección “VUELTA” en el <i>backend</i> implementado. . . . .	68
35. Predicción del tiempo estimado de llegada a la próxima estación teórica y consulta de tiempos históricos A2A en el <i>backend</i> implementado. . . . .	68
36. Respuesta final del <i>backend</i> implementado para la consulta del tiempo estimado de llegada a la estación “MONTÚFAR”, línea 13, dirección “VUELTA” el 13 de mayo de 2024 a las 14:38:00 y su despliegue en el <i>frontend</i> . . . . .	69
37. Respuesta final del <i>backend</i> implementado para la consulta del tiempo estimado de llegada a la estación “MONTÚFAR”, línea 13, dirección “VUELTA” el 13 de mayo de 2024 a las 23:00:00 y su despliegue en el <i>frontend</i> . . . . .	69
38. Predicción del tiempo estimado de duración del viaje entre las estaciones “MONTÚFAR” y “FUERZA AÉREA” el 13 de mayo de 2024 a las 14:38:00. . . . .	70
39. Respuesta final del <i>backend</i> implementado para la consulta del tiempo estimado de duración del viaje entre las estaciones “MONTÚFAR” y “FUERZA AÉREA” el 13 de mayo de 2024 a las 14:38:00 y su despliegue en el <i>frontend</i> . . . . .	70
40. Polilínea generada para la línea 1 de Transmetro. . . . .	85
41. Polilínea generada para la línea 2 de Transmetro. . . . .	86
42. Polilínea generada para la línea 6 de Transmetro. . . . .	87
43. Polilínea generada para la línea 7 de Transmetro. . . . .	88
44. Polilínea generada para la línea 12 de Transmetro. . . . .	89
45. Polilínea generada para la línea 13-A de Transmetro. . . . .	90
46. Polilínea generada para la línea 13-B de Transmetro. . . . .	91
47. Polilínea generada para la línea 18-A de Transmetro. . . . .	91
48. Polilínea generada para la línea 18-B de Transmetro. . . . .	92

---

## Lista de cuadros

---

1.	Distribución del parque vehicular por tipo de vehículo según cifras de la SAT (31 de mayo de 2025) [19]. . . . .	11
2.	Número de registros por línea en el conjunto de datos preprocesado. . . . .	41
3.	Resumen del muestreo estratificado por línea. . . . .	42
4.	Configuración principal de LightGBM. . . . .	44
5.	Configuración principal de XGBoost. . . . .	45
6.	Resumen del conjunto de datos antes y después de la limpieza. . . . .	57
8.	Resultado del algoritmo de Viterbi para el viaje 1 de la unidad 113. . . . .	58
7.	Resumen de las características generadas. . . . .	59
9.	Resultado del algoritmo de Viterbi para el viaje 146 de la unidad 204. . . . .	60
10.	Resumen del conjunto de datos antes y después de la limpieza con las características generadas. . . . .	61
11.	Desempeño del modelo LightGBM. . . . .	62
12.	Umbrales de error absoluto del modelo LightGBM. . . . .	62
13.	Desempeño del modelo LightGBM en la última iteración del entrenamiento. .	63
14.	Desempeño del modelo XGBoost. . . . .	65
15.	Umbrales de error absoluto del modelo XGBoost. . . . .	65
16.	Desempeño del modelo XGBoost en la última iteración del entrenamiento. .	65
17.	Formato de las estadísticas A2A históricas. . . . .	67
18.	Formato de las estadísticas D2A históricas. . . . .	67



---

## Resumen

---

El crecimiento acelerado de las ciudades ha planteado grandes retos en cuanto a movilidad urbana. En particular, la Ciudad de Guatemala enfrenta diariamente un alto nivel de congestionamiento vehicular que impacta negativamente la calidad de vida de sus habitantes. El parque vehicular, que en enero de 2025 alcanzó las 2,478,946 unidades, continúa en aumento, lo cual refuerza la necesidad de fortalecer el transporte público como una alternativa viable y eficiente frente al uso de vehículos particulares.

Entre las iniciativas impulsadas para atender esta necesidad, el sistema Transmetro se ha posicionado como una opción estratégica de movilidad urbana al contar con carriles exclusivos y rutas planificadas. No obstante, el sistema carece de una fuente centralizada de información que le permita a los usuarios conocer en tiempo real la ubicación de las unidades, los tiempos de espera o la duración estimada de un viaje entre estaciones. Esta falta de información genera incertidumbre, desincentiva su uso y limita el potencial del sistema como solución integral a los problemas de tráfico de la ciudad.

Ante esta situación, el presente proyecto propone el desarrollo de un sistema de predicción y visualización que utiliza datos reales del comportamiento de las unidades de Transmetro para estimar los tiempos de llegada de las unidades a las estaciones según su ubicación y la duración total de viaje entre estaciones pertenecientes a una misma línea. Para ello, se aprovechó la información histórica sobre la ubicación y velocidad de las unidades proporcionada por la Empresa Municipal de Transporte (EMT). Estos datos fueron procesados y utilizados para entrenar modelos de aprendizaje automático. Los resultados demostraron la utilidad práctica de dichos modelos para aplicaciones operativas y de información al usuario. Esta información se integró en una plataforma visual de fácil interacción para los potenciales usuarios.

Este proyecto busca no solo optimizar la experiencia de los actuales usuarios de Transmetro, sino también incentivar su adopción por parte de nuevos usuarios, contribuyendo así a una movilidad urbana más eficiente, sostenible y accesible en la Ciudad de Guatemala.



---

## Abstract

---

The accelerated growth of cities has posed significant challenges in terms of urban mobility. Particularly, Guatemala City faces high levels of traffic congestion on a daily basis, which negatively impacts the quality of life of its residents. As of January 2025, the number of automobiles reached 2,478,946 units and continues to grow, reinforcing the need to strengthen public transportation as a viable and efficient alternative to the use of private vehicles.

Among the initiatives aimed at addressing this need, the Transmetro system has positioned itself as a strategic urban mobility option due to its exclusive lanes and planned routes. However, the system lacks a centralized source of information that would allow users to know in real time the location of the buses, waiting times, or the estimated duration of a trip between stations. This lack of information generates uncertainty, discourages usage, and limits the system's potential as a comprehensive solution to the city's traffic problems.

In response to this situation, this project proposes the development of a prediction and visualization system that uses real data on Transmetro unit behavior to estimate the arrival times of units based on their location, as well as the total trip duration between stations on the same line. To achieve this, historical data on bus location and speed, provided by the Municipal Transport Company (EMT), was processed and used to train machine learning models. The results demonstrated the practical utility of these models for operational applications and user information. This information was integrated into an easily interactive visual platform for potential users.

This project aims not only to optimize the experience of current Transmetro users, but also to encourage adoption by new users, thereby contributing to more efficient, sustainable, and accessible urban mobility in Guatemala City.



# CAPÍTULO 1

---

## Introducción

---

El crecimiento poblacional de una ciudad conlleva inevitablemente un incremento en la congestión vehicular, influyendo significativamente en las actividades económicas de su población, el impacto ambiental y la calidad de vida de los habitantes [1]. La Ciudad de Guatemala no es la excepción: su parque vehicular continúa en aumento [2], [3], lo que hace cada vez más urgente la implementación de un sistema de transporte público eficiente.

El sistema de Transmetro, implementado como una de las principales alternativas de transporte público para mitigar el congestionamiento vehicular en la Ciudad de Guatemala, moviliza diariamente a cientos de miles de personas [4]. No obstante, la información de sus unidades, estaciones y rutas no se encuentra centralizada ni disponible en tiempo real para los usuarios. Esta carencia dificulta la planificación de viajes, reduce la previsibilidad del sistema y puede desincentivar su uso tanto por parte de usuarios actuales como de potenciales nuevos pasajeros [5].

Esta falta de información oportuna genera incertidumbre en los usuarios, lo cual compromete tanto la eficiencia del sistema como la percepción de confiabilidad por parte de los usuarios [5]. Ante esta problemática, la posibilidad de incorporar herramientas tecnológicas que permitan estimar y visualizar el comportamiento de las unidades en tiempo real surge como una oportunidad para mejorar la experiencia del usuario de Transmetro y optimizar el servicio.

De hecho, la confiabilidad es un componente esencial en cualquier sistema de transporte, y uno de sus principales indicadores es la capacidad de prever con exactitud los tiempos de llegada de las unidades. Desde la perspectiva de los usuarios, la previsibilidad del transporte es generalmente medible por la precisión de un tiempo estimado de llegada (ETA, por sus siglas en inglés). [6] Los modelos de aprendizaje son especialmente adecuados para tareas de predicción como la estimación del tiempo de llegada, gracias a su capacidad de identificar patrones complejos y no lineales en un conjunto de datos históricos [7]. En cuanto a transporte urbano, algoritmos como árboles de decisión, random forest, XGBoost y redes neuronales recurrentes han sido utilizados exitosamente para capturar el comportamiento dinámico de este, incluso bajo condiciones cambiantes [8], [9].

A diferencia de métodos estadísticos tradicionales, que generalmente asumen distribuciones fijas y relaciones lineales entre variables, los modelos de aprendizaje automático permiten adaptarse a nuevos datos y mejorar su precisión con el tiempo, lo cual resulta útil en sistemas donde las condiciones operativas varían constantemente [7]. Esta flexibilidad y capacidad de generalización convierten al aprendizaje automático en una herramienta eficaz para capturar patrones complejos en el comportamiento del transporte público urbano [6].

En este contexto, diversos estudios han encontrado puntos importantes: primero, el ETA es un dato valorado por los usuarios de transporte, ya que les permite tomar las mejores decisiones para llegar de manera eficiente a su destino [10]; segundo, el entrenamiento de modelos predictivos con datos históricos acerca de las rutas de transporte público permite obtener predicciones acerca del ETA de sus unidades con una precisión razonable, incluso sin información acerca del flujo del tráfico que podría intervenir en su trayectoria [6].

En base a esto, el presente proyecto pretende contribuir a la modernización y eficiencia del sistema de transporte público en la Ciudad de Guatemala, promoviendo su uso como una alternativa viable frente al transporte particular. Su ejecución se basa en el desarrollo de un sistema de predicción y visualización, aprovechando datos históricos reales del comportamiento operativo de las unidades de Transmetro, proporcionados por la Empresa Municipal de Transporte (EMT).

A través de la implementación y entrenamiento de modelos de aprendizaje automático, se busca estimar el tiempo de llegada de las unidades a las estaciones, así como el tiempo de viaje entre estaciones pertenecientes a una misma línea. Estos resultados se integran en una plataforma visual que facilita la consulta de esta información por parte de potenciales usuarios.

## CAPÍTULO 2

---

### Justificación

---

El acceso a servicios de transporte público adecuados está íntimamente relacionado al bienestar de áreas metropolitanas, dado que el desarrollo de la población urbana depende de su posibilidad de desplazarse para realizar actividades económicas, educativas y recreativas [1]. Debido a que los sistemas de transporte constituyen uno de los medios por los que una sociedad puede interactuar, un sistema de transporte público ineficiente limita las oportunidades económicas y sociales de esta [11].

En la Ciudad de Guatemala, donde se concentran las principales actividades económicas del país, la movilidad es un problema crítico que influye en la calidad de vida de los habitantes. Para el mes de enero de 2025, se registró un parque vehicular de 2,478,946 unidades, lo cual representa un aumento de 20,999 unidades respecto al mes de diciembre de 2024 [2] [3]. Este aumento de aproximadamente 0.85 % en la circulación de vehículos en la Ciudad de Guatemala en tan solo un mes destaca la importancia de contar con un sistema de transporte público accesible y eficiente, de manera que más habitantes de la ciudad decidan utilizarlo en lugar de transportarse en su propio vehículo particular.

Una de las iniciativas propuestas para solucionar el congestionamiento vehicular y satisfacer la necesidad de un transporte público eficiente y seguro en la Ciudad de Guatemala fue el Transmetro: un sistema de autobuses con carriles exclusivos y estaciones. En 2021, este sistema movilizaba a aproximadamente 450,000 personas diarias y, en años recientes, se ha consolidado como uno de los principales medios de transporte público en la ciudad [4]. Sin embargo, los usuarios de Transmetro (y de otros medios de transporte público) no cuentan con una plataforma centralizada que les proporcione información acerca de rutas, estaciones o tiempos de espera. Las aplicaciones disponibles que aportan información sobre el tráfico en la ciudad están diseñadas principalmente para conductores de vehículos particulares o no proporcionan información en tiempo real acerca de las unidades de transporte público. La ausencia de esta información dificulta la planificación de viajes a los usuarios de Transmetro, reduce la percepción de confiabilidad del sistema y puede desmotivar su uso por parte de pasajeros actuales y potenciales, reduciendo así la eficiencia del sistema de Transmetro [5].

Entre los factores que determinan la calidad de un sistema de transporte público destaca

la previsibilidad del servicio. En particular, el tiempo estimado de llegada (ETA, por sus siglas en inglés) es uno de los datos más valorados por los usuarios de transporte, ya que les permite tomar decisiones informadas para llegar a su destino de forma eficiente [6]. Desde el punto de vista del pasajero, poseer esta información en tiempo real puede influir en sus decisiones de transporte. Por ejemplo, pueden elegir entre dirigirse a una estación con menor tiempo de espera o tomar una ruta más rápida [12]. Además, con el auge de los teléfonos inteligentes y el uso generalizado de sistemas GPS en vehículos, se ha vuelto técnicamente viable y cada vez más común mostrar esta información en aplicaciones móviles o plataformas web [6].

La relevancia de brindar información precisa en tiempo real también ha sido respaldada por diversos estudios. Por ejemplo, una investigación realizada en Kuala Lumpur demostró que el uso de paneles de mensajes variables para informar a conductores del tiempo estimado de viaje influyó directamente en sus decisiones de ruta, resultando en una reducción del tiempo promedio de viaje [13].

Por otro lado, un estudio realizado por Kumar et al. demostró que los vehículos del transporte público pueden ser utilizados como sensores móviles para recolectar datos de tráfico en tiempo real, sin necesidad de infraestructura adicional. En este estudio, se propone un enfoque basado en modelos para predecir el tiempo de viaje utilizando autobuses como vehículos de sondeo, obteniendo resultados prometedores en la estimación del flujo vehicular urbano [10]. El enfoque utilizado en este estudio evidencia el potencial de aprovechar los datos de la circulación de transporte público como una fuente para comprender y modelar el comportamiento del sistema.

La situación actual en la Ciudad de Guatemala, combinada con la creciente demanda de movilidad eficiente y la disponibilidad de tecnologías de monitoreo, evidencia la necesidad de investigar nuevas formas de aprovechar los datos de circulación de transporte público existentes para mejorar la experiencia de los usuarios y la eficacia de este sistema.

# CAPÍTULO 3

---

## Objetivos

---

### 3.1. Objetivo general

Implementar modelos predictivos que permitan estimar los tiempos de llegada de las unidades de Transmetro a las estaciones en la Ciudad de Guatemala, así como la duración total de viajes entre estaciones pertenecientes a una misma línea.

### 3.2. Objetivos específicos

- Preprocesar los datos proporcionados por la Empresa Municipal de Transporte (EMT) acerca del recorrido de las unidades de Transmetro en la Ciudad de Guatemala, con el fin de adecuarlos a un formato estructurado que facilite la implementación de modelos predictivos
- Implementar modelos de aprendizaje automático para estimar los tiempos de llegada y la duración total de viajes, evaluando su desempeño con métricas correspondientes a tareas de regresión.
- Diseñar una interfaz que integre los modelos de aprendizaje automático, permitiendo visualizar de forma dinámica las predicciones realizadas por estos.



## CAPÍTULO 4

---

### Alcance

---

Este proyecto busca determinar la viabilidad de implementar modelos de aprendizaje automático para predecir tiempos de llegada de las unidades de Transmetro a estaciones dentro de las líneas en las que estas se encuentran operando. El alcance del proyecto incluye:

- Recolección de datos históricos de ubicación de las unidades de Transmetro.
- Análisis exploratorio de los datos para identificar patrones, tendencias y posibles desafíos.
- Limpieza y preprocesamiento de los datos para su uso en modelos de aprendizaje automático.
- Desarrollo y evaluación de modelos de aprendizaje automático para predecir los tiempos de llegada a las estaciones.
- Implementación de un prototipo funcional en una interfaz de cara a posibles usuarios finales que integre los modelos en un entorno simulado, permitiendo la evaluación del desempeño en condiciones cercanas a la producción.

El proyecto no incluye:

- Optimización del rendimiento de los modelos mediante la búsqueda avanzada de hiperparámetros.
- Implementación en un entorno de producción real.
- Integración con sistemas existentes de Transmetro.
- Desarrollo de modelos para otros sistemas de transporte.



# CAPÍTULO 5

---

## Marco teórico

---

### 5.1. Movilidad Urbana

La movilidad urbana se refiere al desplazamiento de personas y bienes dentro de las ciudades, independientemente del medio de transporte utilizado [14]. Constituye un servicio urbano fundamental que moldea la estructura espacial de las áreas urbanas, influyendo directamente en el acceso equitativo a oportunidades, servicios públicos y vivienda [15].

#### 5.1.1. Movilidad urbana sostenible

Un sistema de movilidad urbana sostenible es aquel que da respuesta a las necesidades actuales de movilidad de las ciudades sin comprometer la capacidad de las generaciones futuras para satisfacer sus propias necesidades [16]. En este contexto, a medida que las ciudades se expanden, los sistemas de transporte desempeñan un papel crucial en la promoción de la inclusión social y la productividad económica, al tiempo que contribuyen a la reducción del impacto ambiental [15]. Dentro de ese marco sostenible, el transporte público emerge como una de las principales estrategias para movilizar grandes volúmenes de población de forma eficiente y con bajo impacto ambiental.

#### 5.1.2. Transporte Público y Autobús de Tránsito Rápido

Ampliar la cobertura y frecuencia del transporte público para mejorar la movilidad en las ciudades trae beneficios incuestionables, ya que desplaza a más personas con menos vehículos y menos energía y ocupa menos espacio. Además, la inversión en este sector aporta beneficios económicos como la creación directa de empleo y el apoyo indirecto a la industria, la construcción y otras actividades económicas. En el aspecto social, el incremento de transporte público favorece el acceso a empleo, educación y a servicios básicos como salud. [16]

La cuota modal del transporte público ha disminuido o se ha estancado en la mayoría de ciudades de los países en desarrollo, que apenas cuentan con sistemas públicos eficientes. Por lo general, este transporte está gestionado por un número creciente de emprendedores o de pequeñas y medianas empresas. Sin embargo, se observan algunas tendencias esperanzadoras en ciertos países latinoamericanos que han introducido sistemas de autobús de tránsito rápido (BRT, por sus siglas en inglés) que han permitido ampliar los servicios de transporte público de manera significativa. [16]

### **Autobús de tránsito rápido**

El autobús de tránsito rápido es un modo de transporte masivo que combina la flexibilidad de un servicio de autobuses con la calidad de servicio de tránsito ferroviario [17]. Es decir, el autobús de tránsito rápido es un sistema que integra servicios y amenidades diseñadas para mejorar la velocidad y confiabilidad del tránsito de autobuses y, aunque la implementación de un sistema de autobús de tránsito rápido varía según las necesidades de la comunidad que lo utiliza, incorpora siete componentes fundamentales:

- Vehículos que transitan principalmente en vías exclusivas que eliminan interferencias con el tráfico general.
- Estaciones dedicadas de fácil acceso y convenientemente localizadas dentro de la comunidad que utiliza el sistema.
- Vehículos especializados de fácil acceso.
- Disponibilidad de servicio durante la mayor parte del día y de alta frecuencia.
- Estructura de rutas planificada y bien distinguida.
- Sistemas de recolección de tarifa rápidos y fáciles de usar, usualmente para pagar antes de subirse al vehículo.
- Uso de tecnologías digitales para mejorar la conveniencia de los usuarios, velocidad, confiabilidad y seguridad de operaciones.

[17]

#### **5.1.3. Movilidad urbana en Guatemala**

El sistema de transporte público en la Ciudad de Guatemala se compone de diferentes medios, incluyendo transporte público formal (Transmetro, Transurbano y TuBus) y transporte informal [18].

No obstante, el parque vehicular en la ciudad no deja de crecer: en diciembre de 2024 ascendía a 2,457,947 vehículos, y para mayo de 2025 ya sumaba 2,564,014, lo que representa un aumento de aproximadamente 4.32 % en solo seis meses [2] [3]. El cuadro 1 presenta la distribución nacional del parque vehicular según la SAT. Aunque estos datos son del total

del país, reflejan la tendencia de ocupación del espacio vial: sólo el 2.09 % corresponde a autobuses y microbuses, mientras que las motocicletas y los vehículos ligeros (automóviles, camionetas y pick-ups) suman más del 76 % del total [3].

Cuadro 1: Distribución del parque vehicular por tipo de vehículo según cifras de la SAT (31 de mayo de 2025) [19].

<b>Tipo de vehículo</b>	<b>Cantidad</b>	<b>% Representación</b>
Motocicletas	2,979,525	49.22 %
Automóviles	930,554	15.37 %
Camionetas, Camionetas y Paneles	855,555	14.13 %
Pick-Up	813,481	13.44 %
Camiones, Cabezal y Transporte de Carga	247,484	4.09 %
Autobuses, Buses, Microbuses	126,228	2.09 %
Furgones y Plataformas	37,456	0.62 %
Jeep	24,696	0.41 %
Carretas, Carretones, Remolques, etc.	13,472	0.22 %
Grúas	2,982	0.05 %
Tractores y Minitractores	758	0.01 %
Otros	21,165	0.35 %
<b>Total</b>	<b>6,053,356</b>	<b>100.00 %</b>

La situación del transporte público en la Ciudad de Guatemala, con una cuota modal tan reducida, enfatiza las deficiencias del sistema colectivo a pesar de los esfuerzos de la Municipalidad por implementar soluciones de BRT en los últimos 17 años [18]. A continuación, se describe el sistema Transmetro, su origen, arquitectura y principales características.

## 5.2. Sistema de Transmetro

La Empresa Municipal de Transporte (EMT) es la encargada de administrar, coordinar, controlar y operar el sistema de Transmetro [20]. El Transmetro es un sistema de transporte público tipo autobús de tránsito rápido que opera en la Ciudad de Guatemala desde febrero de 2007 y que, a la fecha, se ha convertido en una de las principales alternativas de transporte público para mitigar el congestionamiento vehicular en la ciudad [4].

Al tratarse de un autobús de tránsito rápido, el sistema de Transmetro cuenta con carriles exclusivos, estaciones accesibles y estratégicamente localizadas en la ciudad, rutas definidas y tecnologías inteligentes para su gestión. Como se muestra en la figura 1, actualmente, el sistema opera 7 líneas diferenciadas y transporta a cientos de miles de pasajeros diariamente [4].

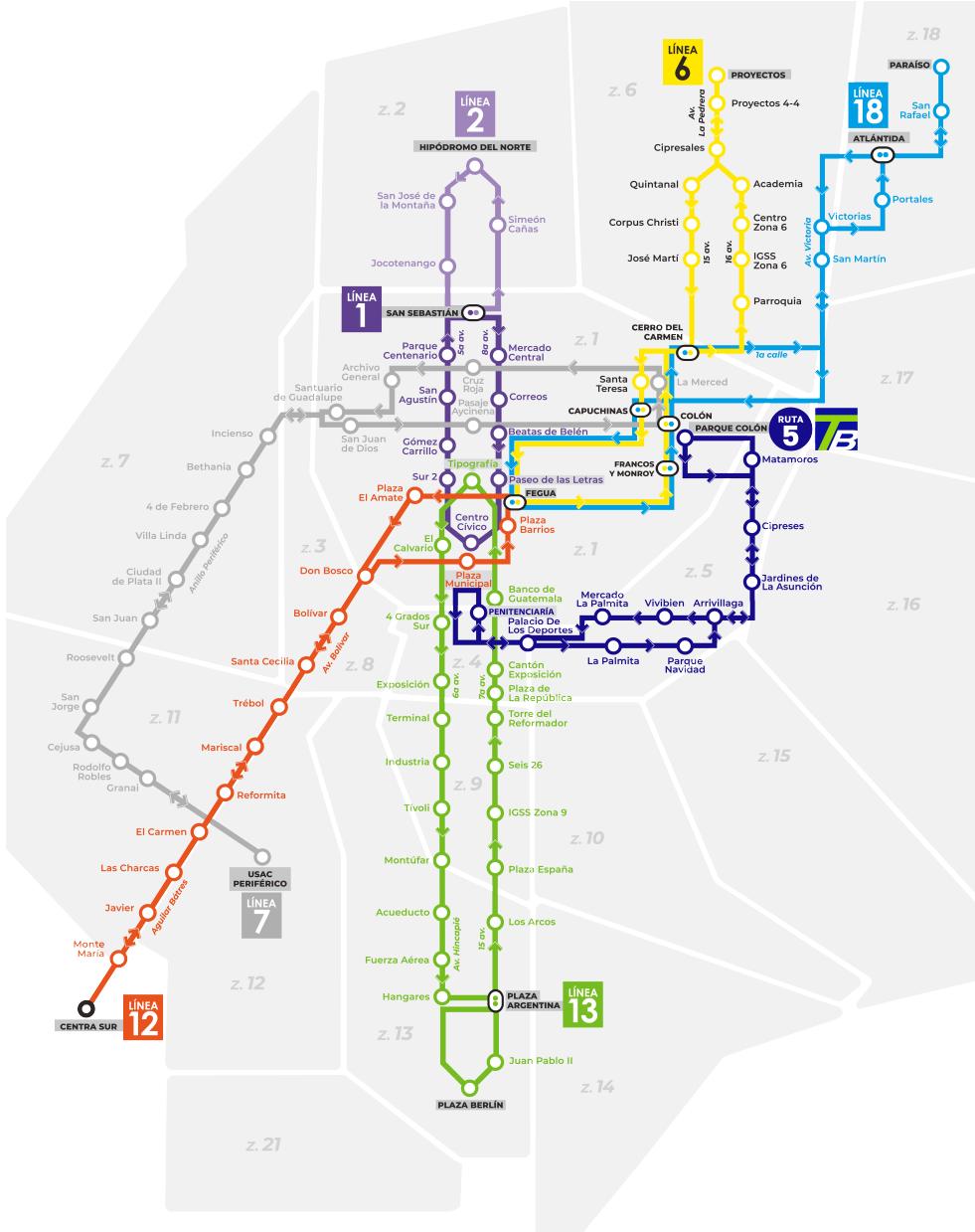


Figura 1: Mapa oficial del sistema Transmetro de la Ciudad de Guatemala. Fuente: Municipalidad de Guatemala [21].

No obstante, la información de sus unidades, estaciones y rutas no se encuentra centralizada ni disponible en tiempo real para los usuarios. La disponibilidad de tecnologías de información eficientes, precisas y accesibles es un factor clave en la percepción de calidad del servicio por parte de los usuarios [22]. En particular, se ha reportado que la información en tiempo real respecto a los tiempos de llegada de las unidades influye significativamente en la decisión de los usuarios para utilizar un autobús de tránsito rápido [23]. Por ello, es una característica importante de los sistemas de autobús de tránsito rápido.

### 5.3. Tiempo estimado de llegada

La información en tiempo real presentada a los usuarios de transporte público en las estaciones reduce significativamente la incertidumbre respecto al tiempo de llegada de las unidades y el tiempo de espera percibido [24]. En particular, el tiempo estimado de llegada es uno de los datos más valorados por los usuarios de transporte, ya que les permite tomar decisiones informadas para llegar a su destino de forma eficiente [6].

En aplicaciones de transporte y planificación, el tiempo estimado de llegada se define como el tiempo restante que se anticipa transcurrirá desde el momento actual hasta que un vehículo llegue a un punto de destino determinado [25].

Diversos estudios han demostrado que los modelos de aprendizaje automático son capaces de proporcionar estimaciones fiables de tiempo de llegada de unidades de transporte público, incluso en contextos de datos limitados y con características específicas de los sistemas de transporte público [6], [26].

### 5.4. Sistemas de información geográfica

Los sistemas de información geográfica (SIG) son herramientas computacionales diseñadas para capturar, almacenar, analizar y visualizar datos geoespaciales. En el contexto del transporte público, los SIG permiten gestionar y analizar información relacionada con la ubicación de rutas, estaciones y unidades en tiempo real, facilitando la toma de decisiones [27].

La EMT utiliza un SIG para monitorear y gestionar las operaciones del sistema Transmetro. Los dispositivos GPS como los que utilizan sus unidades para reportar su posición en tiempo real en coordenadas geográficas (latitud y longitud) se basan en WGS84, un sistema de referencia geodésico global que define un marco común para la representación de ubicaciones en la superficie terrestre [28].

#### 5.4.1. Cálculo de distancias geodésicas

Para calcular distancias entre dos puntos en la superficie terrestre utilizando sus coordenadas geográficas, como lo podrían ser la ubicación de una unidad y una estación del sistema Transmetro, es posible utilizar BallTree, una estructura de datos que resulta especialmente útil debido a su eficiencia en la búsqueda de vecinos más cercanos en grandes conjuntos de datos geoespaciales [29].

La librería scikit-learn de Python implementa BallTree con soporte para distancias geodésicas, empleando la fórmula del semiverseno, la cual aproxima la distancia entre dos puntos sobre una esfera a partir de sus coordenadas de latitud y longitud:

$$d = 2r \arcsin \left( \sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)} \right),$$

donde  $r$  es el radio medio de la Tierra,  $\varphi_1$  y  $\varphi_2$  representan la latitud y longitud del primer punto en radianes, y  $\lambda_1$  y  $\lambda_2$  representan latitud y longitud del segundo punto en radianes. [30]

#### 5.4.2. Proyecciones cartográficas locales

Para facilitar cálculos de distancia y análisis espaciales se utilizan proyecciones cartográficas que transforman las coordenadas geográficas (latitud y longitud) en coordenadas planas ( $X$ ,  $Y$ ) en un sistema de coordenadas proyectadas [31].

Una proyección cartográfica es una transformación de puntos longitudinales y latitudinales de la superficie curva de la Tierra a un plano bidimensional [28]. Existen diversas proyecciones cartográficas, pero en áreas geográficamente limitadas, como la Ciudad de Guatemala, es posible utilizar proyecciones equirectangulares locales, las cuales aproximan la superficie terrestre como plana y utilizan factores de conversión que dependen de la latitud de referencia:

$$m_\varphi = 111132.92 - 559.82 \cos(2\varphi) + 1.175 \cos(4\varphi) - 0.0023 \cos(6\varphi)$$

$$m_\lambda = 111412.84 \cos(\varphi) - 93.5 \cos(3\varphi) + 0.118 \cos(5\varphi)$$

donde  $m_\varphi$  y  $m_\lambda$  son los factores de conversión en metros por grado para latitud y longitud respectivamente, y  $\varphi$  es la latitud de referencia en grados.

A partir de estos factores, la transformación entre coordenadas geográficas y coordenadas planas locales se expresa como:

$$x = (\lambda - \lambda_0) m_\lambda(\varphi_0),$$

$$y = (\varphi - \varphi_0) m_\varphi(\varphi_0),$$

donde  $(\varphi_0, \lambda_0)$  representa la latitud y longitud del origen local. Esta aproximación permite operar directamente en unidades métricas, evitando el uso de proyecciones complejas cuando el área de estudio es pequeña. [31]

#### 5.4.3. Estructuración de rutas mediante polilíneas

Las rutas de las unidades de transporte público, como las líneas del sistema Transmetro, pueden representarse mediante polilíneas, que son secuencias ordenadas de puntos en el espacio bidimensional, conectados por segmentos lineales. Esto facilita la asociación de posiciones GPS a ubicaciones específicas a lo largo de las rutas, proceso conocido como *map-matching* [32].

#### 5.4.4. Proyección punto-segmento

La proyección punto-segmento es una técnica utilizada para determinar la ubicación de un punto en relación con una polilínea [32]. Al modelar un ruta como una polilínea y convertir las coordenadas geográficas a un sistema plano local, cada registro GPS puede asociarse al punto más cercano de dicha ruta mediante una proyección geométrica sobre la polilínea que la representa.

Dada una polilínea compuesta por segmentos consecutivos  $\overline{AB}$ , la proyección ortogonal  $P'$  de un punto  $P$  sobre uno de estos segmentos se obtiene como:

$$t = \frac{(P - A) \cdot (B - A)}{\|B - A\|^2}, \quad P' = A + t(B - A),$$

donde  $t$  es el parámetro escalar de la proyección. Si  $t \in [0, 1]$ , el punto proyectado  $P'$  se encuentra dentro del segmento; en caso contrario, se le asigna el extremo más cercano. Este procedimiento, repetido para todos los segmentos de la polilínea, permite encontrar el punto  $P'$  de la ruta cuya distancia a  $P$  es mínima. [33]

Una vez proyectado el punto  $P$  sobre la polilínea, es posible calcular la distancia acumulada a lo largo de la ruta desde un punto de referencia inicial hasta el punto proyectado  $P'$  (abcisa curvilínea), lo cual es útil para determinar la posición relativa de una unidad en la ruta. Si la polilínea está definida por los puntos  $P_0, P_1, \dots, P_k$ , la distancia acumulada  $s$  desde  $P_0$  hasta  $P'$  se calcula como:

$$s = \sum_{i=0}^{k-1} \|P_i - P_{i+1}\| + \|P_k - P'\|,$$

donde  $k$  es el índice del segmento en el que se encuentra  $P'$ .

### 5.5. Modelos de Markov ocultos

Los modelos de Markov ocultos (HMM, por sus siglas en inglés) son modelos estadísticos utilizados para representar sistemas donde el estado real no es observable directamente, sino que se infiere a partir de observaciones ruidosas.

Un HMM se define por tres componentes principales:

- Un conjunto finito de estados ocultos  $S = \{s_1, s_2, \dots, s_N\}$ .
- Un conjunto de observaciones  $O = \{o_1, o_2, \dots, o_M\}$ .
- Tres matrices de probabilidad:
  - La matriz de transición de estados  $A = [a_{ij}]$ , donde  $a_{ij} = P(s_j | s_i)$  es la probabilidad de transición del estado  $s_i$  al estado  $s_j$ .

- La matriz de emisión  $B = [b_j(o_k)]$ , donde  $b_j(o_k) = P(o_k|s_j)$  es la probabilidad de observar  $o_k$  dado el estado  $s_j$ .
- El vector de probabilidades iniciales  $\pi = [\pi_i]$ , donde  $\pi_i = P(s_i)$  es la probabilidad de iniciar en el estado  $s_i$ .

[34]

El problema de asignar secuencias de puntos GPS a líneas del sistema Transmetro puede abordarse mediante un HMM, donde los estados ocultos representan las posibles líneas del sistema y las observaciones corresponden a las posiciones GPS registradas.

Existen tres problemas fundamentales asociados a los HMM:

- **Evaluación:** Dada una secuencia de observaciones y un modelo HMM, calcular la probabilidad de que el modelo haya generado dicha secuencia.
- **Decodificación:** Dada una secuencia de observaciones y un modelo HMM, encontrar la secuencia más probable de estados ocultos que generó las observaciones.
- **Aprendizaje:** Dada una secuencia de observaciones, ajustar los parámetros del modelo HMM para maximizar la probabilidad de las observaciones.

[34]

El problema de interés en este trabajo es el de decodificación, ya que se busca la secuencia más probable de estados ocultos (líneas de Transmetro) que generó las observaciones (puntos GPS), para el cual se utiliza el algoritmo de Viterbi [35].

### 5.5.1. Algoritmo de Viterbi

El algoritmo de Viterbi es un método de programación dinámica utilizado para encontrar la secuencia más probable de estados ocultos en un HMM, dada una secuencia de observaciones [35]. Se basa en la construcción de una tabla de probabilidades que almacena la probabilidad máxima de llegar a cada estado en cada paso temporal, así como la ruta que conduce a ese estado.

Dada una secuencia de observaciones  $O = (o_1, o_2, \dots, o_T)$ , el algoritmo de Viterbi procede de la siguiente manera:

1. Inicialización: Para cada estado  $s_i$ , calcular la probabilidad inicial:

$$V_1(i) = \pi_i \cdot b_i(o_1)$$

y almacenar la ruta inicial.

2. Recursión: Para cada tiempo  $t = 2, 3, \dots, T$  y para cada estado  $s_j$ , calcular:

$$V_t(j) = \max_i [V_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t)$$

y actualizar la ruta que conduce al estado  $s_j$ .

3. Terminación: Encontrar la probabilidad máxima al final de la secuencia:

$$P^* = \max_i V_T(i)$$

y rastrear la ruta correspondiente para obtener la secuencia de estados más probable.

[35]

## 5.6. Modelos de aprendizaje de gradiente

Los modelos de *gradient boosting* representan una de las técnicas más efectivas para tareas de predicción supervisada en datos tabulares, combinando la potencia de múltiples modelos débiles (árboles de decisión) en una estructura de *ensamble* secuencial. Su principio fundamental consiste en construir iterativamente nuevos árboles que corrijan los errores cometidos por los árboles anteriores, optimizando una función de pérdida mediante gradientes. [36]

Sea un conjunto de datos  $\{(x_i, y_i)\}_{i=1}^N$  con variables predictoras  $x_i$  y valores objetivo  $y_i$ . El modelo inicial  $F_0(x)$  puede definirse como una constante, típicamente la media de los valores  $y_i$ . En cada iteración  $m$ , el algoritmo ajusta un nuevo árbol de decisión  $h_m(x)$  a las diferencias entre los valores reales y las predicciones actuales, actualizando el modelo como:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x)$$

donde  $\eta$  es la tasa de aprendizaje que controla el grado de corrección en cada paso. Este proceso continúa hasta alcanzar un número máximo de iteraciones o hasta que la mejora en la función de pérdida sea insignificante. El resultado final es un modelo compuesto por una suma ponderada de árboles de decisión, capaz de aproximar relaciones no lineales complejas en los datos. [36]

### 5.6.1. XGBoost

XGBoost introduce optimizaciones en la regularización de los árboles, la gestión de memoria y el paralelismo, lo que le permite entrenar modelos de gran tamaño con alta velocidad. Su algoritmo emplea una estrategia de crecimiento nivel por nivel, que equilibra la profundidad de los árboles y reduce el riesgo de sobreajuste. Además, implementa regularización  $L_1$  y  $L_2$  para controlar la complejidad del modelo y mejorar la generalización. [37]

### 5.6.2. LightGBM

LightGBM propone mejoras adicionales basadas en el uso de histogramas para agrupar valores numéricos y reducir el costo de cómputo en el cálculo de los mejores puntos de división. Asimismo, utiliza un enfoque de crecimiento de árbol basado en hojas en lugar de

niveles, priorizando así las hojas con mayor reducción de pérdida, lo que permite alcanzar mayor precisión con menos iteraciones. Además, implementa un manejo eficiente de variables categóricas, lo que lo hace especialmente adecuado para conjuntos de datos grandes y heterogéneos. [38]

## 5.7. Evaluación y métricas

La evaluación del rendimiento de los modelos de predicción de tiempo estimado de llegada (ETA, por sus siglas en inglés) se basa en métricas de error que cuantifican la discrepancia entre los tiempos reales y los estimados [6]. Estas métricas permiten comparar distintos modelos y establecer su capacidad de generalización hacia datos no observados.

### 5.7.1. Métricas de error en regresión

Las métricas más utilizadas para problemas de regresión son el error absoluto medio (MAE, por sus siglas en inglés) y la raíz del error cuadrático medio (RMSE, por sus siglas en inglés), definidas respectivamente como:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|, \quad \text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2},$$

donde  $y_i$  son los valores observados,  $\hat{y}_i$  las predicciones del modelo y  $N$  el número de observaciones.

El MAE mide el promedio de las desviaciones absolutas y es fácil de interpretar en las unidades originales del problema (en este caso, segundos), mientras que el RMSE penaliza con mayor intensidad los errores grandes debido al cuadrado de las diferencias. [39]

### 5.7.2. Validación cruzada

La calidad de un modelo no solo depende del error promedio, sino también de su capacidad para generalizar a viajes y períodos no vistos durante el entrenamiento. Para estimar esta capacidad, se utilizan esquemas de validación cruzada que dividen el conjunto de datos en particiones de entrenamiento y prueba. [30]

En problemas con dependencias temporales, como la predicción de tiempos de viaje, es fundamental preservar el orden temporal al crear las particiones. Una estrategia común es la validación cruzada temporal, que evalúa el modelo en intervalos de tiempo posteriores al conjunto de entrenamiento, evitando así fugas de información del futuro hacia el pasado (*data leakage*) [40].

La figura 2 ilustra un esquema de validación cruzada temporal donde se crean múltiples particiones de entrenamiento y prueba respetando la secuencia temporal de los datos.

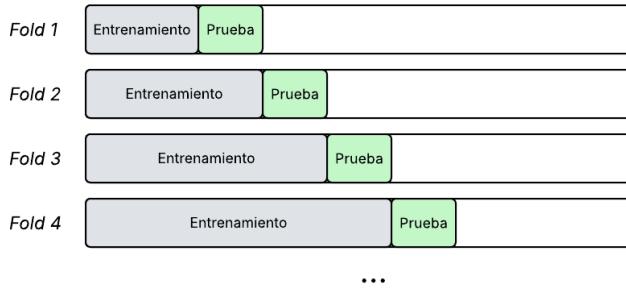


Figura 2: Esquema de validación cruzada temporal para evaluación de modelos predictivos.

## 5.8. Arquitectura de integración de modelos

La integración de modelos de aprendizaje automático dentro de un sistema de predicción de tiempos de llegada requiere componentes de software que permitan la ingestión de los datos más recientes, la ejecución de los modelos entrenados y la presentación de los resultados.

En este trabajo, se propone una arquitectura compuesta de dos capas: un servicio de *backend* encargado de la gestión de datos y la ejecución de los modelos, y una interfaz de usuario que presenta los resultados de forma accesible y comprensible.

### 5.8.1. Backend

El *backend* se implementó utilizando FastAPI, un marco de desarrollo en Python orientado a servicios web basados en transferencia de estado representacional (REST, por sus siglas en inglés). Permite definir y exponer puntos de acceso (*endpoints*) que reciben solicitudes HTTP con parámetros de entrada, ejecutan funciones de predicción y devuelven respuestas en formato JSON. [41]

Consiste de los siguientes componentes clave:

- **Ingesta de datos:** Módulos que recuperan y procesan los datos más recientes de las unidades del sistema Transmetro.
- **Ejecución de modelos:** Funciones que cargan los modelos entrenados y generan predicciones de tiempo estimado de llegada basadas en los datos ingresados.
- **API REST:** Puntos de acceso que permiten a la interfaz de usuario solicitar predicciones y recibir respuestas estructuradas.

### 5.8.2. Frontend

El *frontend* se desarrolló utilizando React, una librería de JavaScript orientada al desarrollo de interfaces de usuario dinámicas. La aplicación web consulta los *endpoints* expuestos

por el *backend* mediante peticiones asíncronas y presenta los resultados de manera interactiva. [42]

# CAPÍTULO 6

---

## Metodología

---

En este capítulo se describe la metodología empleada para la recolección y análisis de datos, el desarrollo de modelos de predicción del tiempo de llegada, la evaluación de su desempeño y la implementación de los resultados en una interfaz interactiva.

### 6.1. Recolección de datos

En primer lugar, se accedió a la plataforma indicada por la Empresa Municipal de Transporte (EMT), un sistema de información geográfica que permite visualizar y generar informes con los registros de posición de las unidades de Transmetro, así como consultar el último punto registrado de cada unidad.

Para descargar los datos que serían de utilidad para el entrenamiento de modelos de predicción, se utilizó la herramienta que ofrece la plataforma para la generación de informes de las posiciones históricas de cada unidad. Sin embargo, esta limitaba el rango de generación del informe de una unidad a un mes, lo que convertía la descarga manual de las posiciones históricas de cada unidad en un proceso lento y propenso a fallos.

Adicionalmente, la plataforma podía congelarse o no devolver resultados al generar el informe de las posiciones históricas de una unidad para fechas específicas. Por ello, para maximizar la cobertura de datos, era necesario identificar y omitir dichas fechas, continuando con el resto del período.

#### 6.1.1. Automatización de la recolección de datos

Para agilizar la recolección de datos se automatizó el proceso con Selenium WebDriver y ChromeDriver, que permitieron interactuar con el navegador para completar los formularios necesarios para la descarga de los informes de posiciones históricas de cada unidad.

Primero, se descargó el informe de “último punto” más reciente, el cual contenía todas las unidades visibles en la plataforma. Luego, se filtraron únicamente las placas correspondientes a unidades de Transmetro, ya que, según las indicaciones de la EMT, la plataforma posee registros tanto de unidades de Transmetro como de unidades de TuBus. Este reporte demostró que no todas las unidades de Transmetro registradas en la plataforma contaban con un último punto válido, ya sea porque fueron descontinuadas, o simplemente porque no habían aportado registros de su posición. Utilizando la librería pandas, se construyó un *dataframe* con el conjunto de placas válidas.

El script de automatización siguió la siguiente lógica:

1. Para todas las unidades, se estableció el 1 de enero de 2024 como la fecha límite de inicio de recolección de datos y el 30 de abril de 2025 como la fecha límite de fin.
2. Para cada placa válida en el *dataframe*, Selenium completó los campos correspondientes del formulario (placa, fecha de inicio y fecha de fin) y solicitó la generación del informe.
3. La confirmación de éxito se realizó verificando la presencia del archivo en el directorio de descargas.
4. En caso de fallo debido a una fecha inválida, se aplicó una estrategia de reducción progresiva del rango temporal: primero un mes (límite máximo), luego una semana y, finalmente, un solo día, hasta lograr la descarga.
5. Las fechas que no fueron posibles de descargar se omitieron y se continuó con el resto del período.
6. Al finalizar cada unidad, los informes descargados se organizaron en un directorio único para dicha unidad, facilitando su posterior preprocesamiento.

## 6.2. Limpieza de datos

Con los informes descargados y organizados por unidad, se procedió a la limpieza y transformación de los datos contenidos en estos, preparándolos para su posterior análisis y uso en el entrenamiento de modelos de predicción.

### 6.2.1. Unificación de informes

La figura 3 muestra el formato de los informes descargados. Para cada unidad, se descargaron varios de estos informes como archivos Excel, cada uno con posiciones históricas en un rango de fechas específico.

Fecha	Hora	Placa	Alias	Motivo	Velocidad (km/h)	Fecha de registro	Hora de registro	Georeferencia	Latitud	Longitud	Altitud (m)
2024-04-01	04:00:29	049		Por tiempo de lectura	0	2024-04-01	04:00:33	A 264 m. de comedor rey saul	14.561165	-90.562033	1399
2024-04-01	04:05:29	049		Por tiempo de lectura	0	2024-04-01	04:05:34	A 264 m. de comedor rey saul	14.561165	-90.562033	1399
2024-04-01	04:07:46	049		Por tiempo de lectura	0	2024-04-01	04:07:51	A 264 m. de comedor rey saul	14.561165	-90.562033	1399
2024-04-01	04:07:50	049		Encendido del vehículo	0	2024-04-01	04:07:51	A 272 m. de comedor rey saul	14.561085	-90.561993	1399
2024-04-01	04:08:50	049		Por tiempo de lectura	0	2024-04-01	04:08:55	A 272 m. de comedor rey saul	14.561085	-90.561993	1399

Figura 3: Ejemplo del formato original de los informes descargados.

Con el propósito de unificar y normalizar los datos, se desarrolló un script que:

- Lee todos los archivos Excel descargados en el directorio único de cada unidad y los combina en un único *dataframe*.
- Omite aquellos archivos en los que el informe consiste únicamente de encabezados; es decir, que no contienen registros de posición.
- Exporta el *dataframe* resultante a un archivo CSV para su posterior análisis.

### 6.2.2. Transformaciones

Para contar únicamente con las variables originales útiles para el análisis, se aplicaron las siguientes transformaciones al *dataframe* unificado de cada unidad:

- Se eliminó la columna “Alias”, ya que, como se observa en la figura 3, esta columna contiene, en su mayoría, datos nulos, y además, no es relevante para la estimación del tiempo de llegada.
- Se eliminaron las columnas “Fecha de registro” y “Hora de registro”, ya que, como se observa en la figura 3, la columna “Hora de registro”, indica la hora en la que se almacenó el registro, en lugar de la hora en la que se capturó la posición del vehículo, la cual suele suceder unos segundos antes. Por otro lado, la columna “Fecha de registro” es redundante, ya que la información de fecha ya se encuentra en la columna “Fecha”.
- Se eliminó la columna “Hora”, ya que esta ya se encuentra representada en la columna “Fecha”.
- Se convirtió la columna “Fecha” al tipo de dato `datetime` de pandas, para facilitar el manejo de fechas y horas en el análisis posterior.

La figura 4 muestra un ejemplo del formato de los *dataframes* una vez aplicadas las transformaciones anteriormente mencionadas.

	<b>Fecha</b>	<b>Placa</b>	<b>Motivo</b>	<b>Velocidad (km/h)</b>	<b>Latitud</b>	<b>Longitud</b>	<b>Altitud (m)</b>
0	2024-04-01 04:00:29	49	Por tiempo de lectura	0.0	14.561165	-90.562035	1399
1	2024-04-01 04:05:29	49	Por tiempo de lectura	0.0	14.561165	-90.562035	1399
2	2024-04-01 04:07:46	49	Por tiempo de lectura	0.0	14.561165	-90.562035	1399
3	2024-04-01 04:07:50	49	Encendido del vehiculo	0.0	14.561085	-90.561996	1399
4	2024-04-01 04:08:50	49	Por tiempo de lectura	0.0	14.561085	-90.561996	1399

Figura 4: Ejemplo del formato de los *dataframes* una vez aplicadas las transformaciones.

### 6.2.3. Filtrado de trayectorias útiles

Al graficar la distribución de los registros de posición en función de la velocidad, como se muestra en la figura 5, se observó que existía un pico en 0 km/h, indicando que había una gran cantidad de registros en los que la unidad estaba detenida. Estos registros pueden

corresponder a situaciones en las que la unidad está en una estación, en un semáforo, o simplemente detenida en el tráfico. Sin embargo, también pueden tratarse de registros durante los que la unidad ya estaba fuera de servicio, o bien, de registros que no corresponden a trayectorias reales, sino a trayectorias de prueba o de mantenimiento. Esto significa que las transformaciones realizadas anteriormente no garantizan que las trayectorias representadas por los registros de posición sean útiles, siendo necesario aplicar un filtrado adicional sobre estos registros.

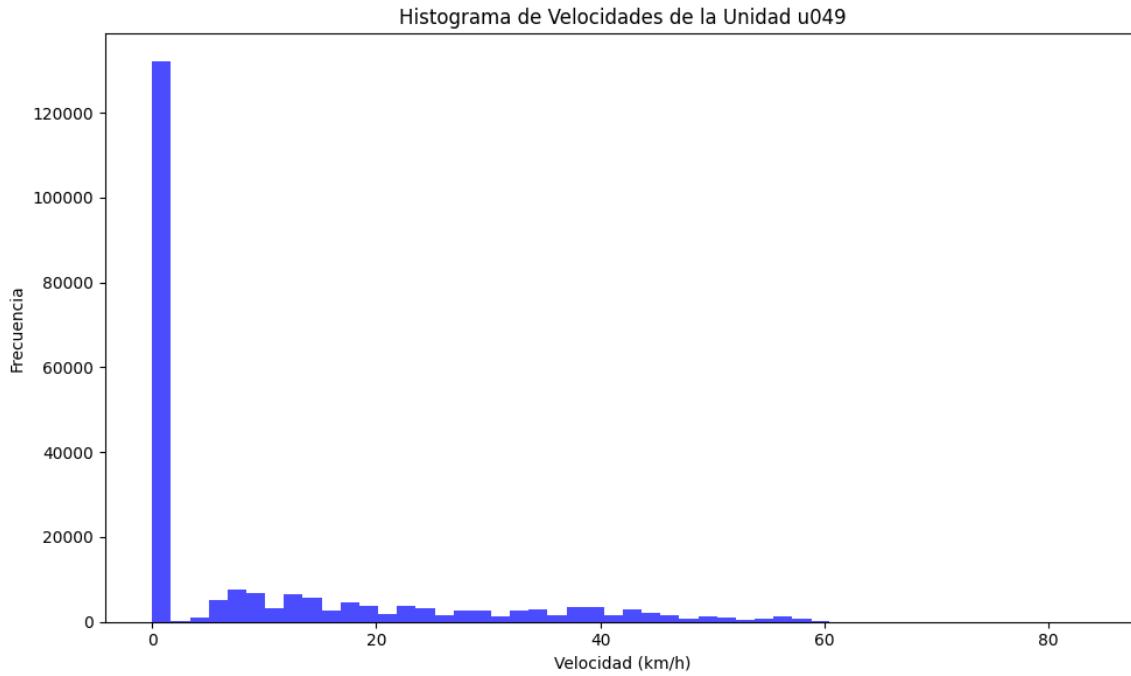


Figura 5: Histograma de velocidades de los registros de posición de la unidad 049.

Para determinar la utilidad de una trayectoria compuesta de registros de posición, es fundamental contar con la disposición geográfica de las estaciones del sistema de Transmetro, ya que esta permite evaluar si dicha trayectoria se alinea con el comportamiento de las unidades cuando están en servicio.

Se llevó a cabo un proceso manual de georreferenciación de las 107 estaciones de Transmetro indicadas en la página oficial de la Municipalidad de Guatemala [21], utilizando la herramienta de Google Maps. Se obtuvieron las coordenadas geográficas (latitud y longitud) de todas estas estaciones y se almacenaron en un archivo CSV. Cabe mencionar que las coordenadas de cada estación fueron verificadas y ajustadas manualmente siguiendo la referencia geográfica de las líneas de Transmetro que se encuentra en la página oficial de la Municipalidad de Guatemala, pero dichas coordenadas fueron obtenidas de Google Maps. La figura 6 muestra un ejemplo de las estaciones georreferenciadas.

Una vez que se contaba con la georreferenciación de las estaciones, para cada punto GPS de las trayectorias de cada unidad, se identificó la estación más cercana y se cuantificó cuántas estaciones distintas fueron visitadas por la unidad en cada día calendario. De no haberse acercado a ninguna estación en el día, se consideró que la trayectoria no era útil y sus registros fueron descartados. El procedimiento se detalla a continuación:

LINEA	ESTACION	POSICIÓN
Linea 1	BEATAS DE BELEN	14.634063226474238, -90.51267210423892
Linea 1	CENTRO CIVICO	14.627739591027462, -90.51487873355744
Linea 1	CORREOS	14.637128091375933, -90.51224041879135
Linea 1	GOMEZ CARRILLO	14.63455197922158, -90.51577607122474
Linea 1	MERCADO CENTRAL	14.642656825735923, -90.51147671685537

Figura 6: Ejemplo de las estaciones georreferenciadas.

**Emparejamiento punto–estación:** Se construyó un índice espacial BallTree con métrica de semiverseno a partir de las coordenadas en radianes de las estaciones georreferenciadas, haciendo uso de la librería scikit-learn. Lógicamente, las coordenadas de cada punto GPS también se transformaron a radianes y, para cada uno de ellos, se consultó el vecino (estación) más cercano ( $k = 1$ ). La distancia angular devuelta por el árbol se convirtió a metros usando un radio terrestre de 6,371 km según la fórmula:

$$d_m = d_{\text{rad}} \times 6,371,000.$$

donde  $d_m$  es la distancia en metros del punto GPS a la estación y  $d_{\text{rad}}$  es la distancia en radianes del punto GPS a la estación.

Entonces, para cada punto GPS se calculó (i) la estación más cercana y (ii) la distancia a dicha estación.

**Asignaciones confiables:** Con el fin de evitar asignaciones forzadas a estaciones lejanas, se estableció un umbral de confianza de 100 m. Es decir, si un punto quedaba a más de 100 m de su estación más cercana, se marcó su emparejamiento como no confiable y no se le asignó estación.

**Conteo diario de estaciones:** Se computó el número de estaciones distintas visitadas por cada unidad en cada fecha calendario indicada en la columna “Fecha”. El resultado fue una serie del conteo de estaciones diarias visitadas por unidad. La figura 7 muestra un ejemplo de esta serie.

**Criterio de filtrado:** Se consideraron **no útiles** las trayectorias de los días calendario en los que la unidad **no visitó ninguna estación**. Estos días se etiquetaron para exclusión del conjunto de datos y se cuantificaron sus puntos asociados. En caso de ser necesario, el umbral puede endurecerse (e.g., exigir  $\geq 2$  estaciones diarias) para eliminar recorridos muy cortos.

**Segmentación, recorte y etiquetado de trayectorias útiles:** Finalmente, se segmentaron las trayectorias **útiles** de cada unidad por día calendario, eliminando las observaciones con un tiempo de registro menor a 8 minutos antes de la primera estación a la que se acercó

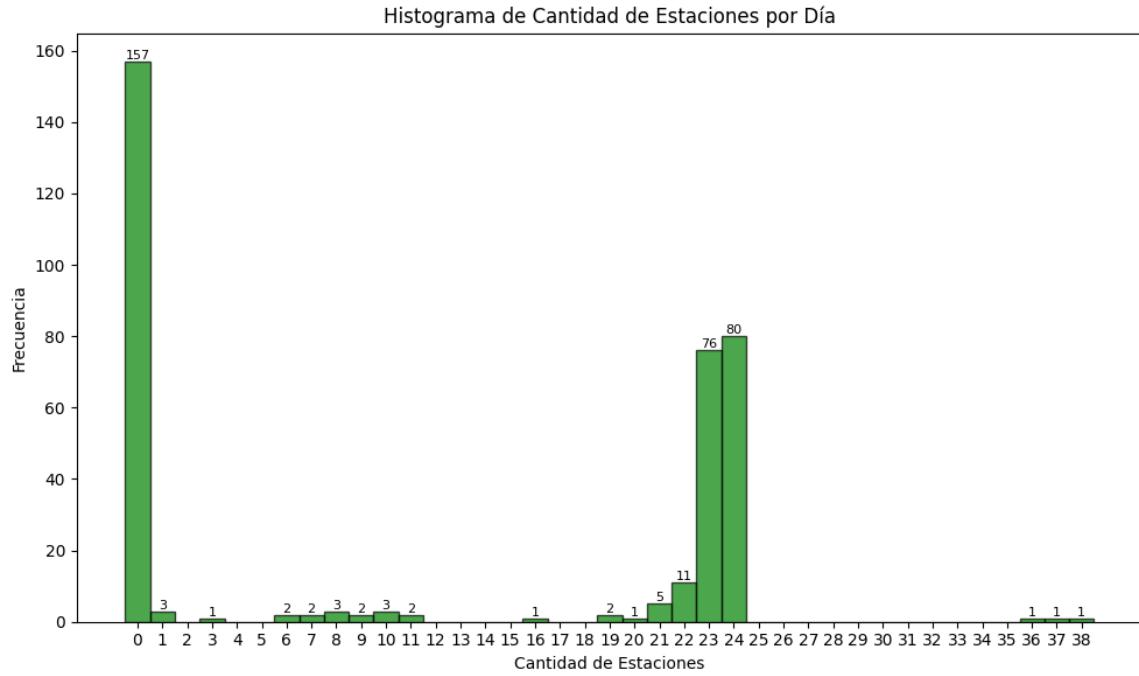


Figura 7: Ejemplo de la serie de estaciones diarias visitadas por la unidad 049.

la unidad en el día y las observaciones con un tiempo de registro mayor a 2 minutos después de que la unidad se alejara de la última estación del día, de manera que las observaciones antes y después de la ventana temporal en la que la unidad realmente está en servicio no afectaran el análisis. A cada trayectoria limpia por día se le etiquetó como **viaje**.

### 6.3. Análisis exploratorio de datos

El análisis exploratorio de datos se realizó con el propósito de comprender las características de los registros de posición de las unidades de Transmetro, identificar patrones en estos, y detectar posibles problemas o inconsistencias en los datos que podrían afectar el desarrollo de los modelos de predicción.

#### 6.3.1. Viajes multilínea

Al utilizar la herramienta de visualización de datos folium, se graficaron las trayectorias de las unidades en un mapa interactivo, lo que permitió observar la distribución espacial de los registros de posición. Gracias a este análisis, se identificaron casos en los que las unidades realizan viajes que abarcan múltiples líneas del sistema Transmetro. Esto puede ser debido a cambios en la demanda de los usuarios o a la necesidad de optimizar las rutas en función de las condiciones del tráfico. En la figura 8 se muestra un viaje realizado por la unidad 204 en el que esta abarcó la línea 2 y la línea 6. Estos viajes multilínea deben ser tomados en cuenta durante el preprocesamiento de datos, ya que afectan la creación de características relacionadas con la línea en la que está operando la unidad, como la siguiente

estación teórica y la distancia restante a esta.

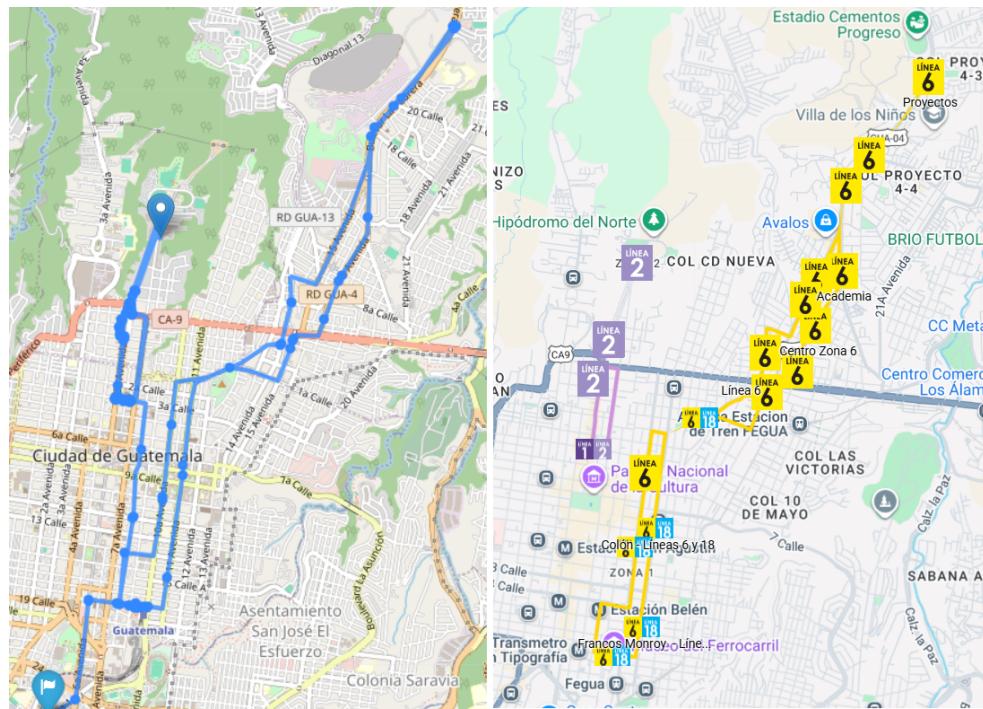


Figura 8: Comparación del viaje 146 de la unidad 204 con el mapa de las líneas 2 y 6 de Transmetro.

### 6.3.2. Tiempo de permanencia en estaciones

El tiempo de permanencia en las estaciones es un factor crítico que puede influir en el tiempo de llegada de una unidad de una estación a otra. Al analizar los registros de posición, se observó que las unidades poseen mayores tiempos de permanencia en estaciones terminales. La figura 9 grafica los puntos en los que la unidad 098 se detuvo por más de 30 segundos al abarcar la línea 18 - Atlántida - Paraíso, evidenciando que las estaciones terminales presentan los mayores tiempos de permanencia.



Figura 9: Tiempos de permanencia de la unidad 098 en la línea 18 - Atlántida - Paraíso.

### 6.3.3. Consistencia temporal de los registros

La consistencia temporal de los registros se refiere a la regularidad y continuidad de los datos a lo largo del tiempo. En el caso de las unidades de Transmetro, al analizar los registros de posición, se observó que la diferencia de tiempo entre registros consecutivos suele ser de 60 segundos, tal como se muestra en la figura 10. Aunque existen registros consecutivos con diferencias de tiempo menores a 60 segundos, estos son menos frecuentes.

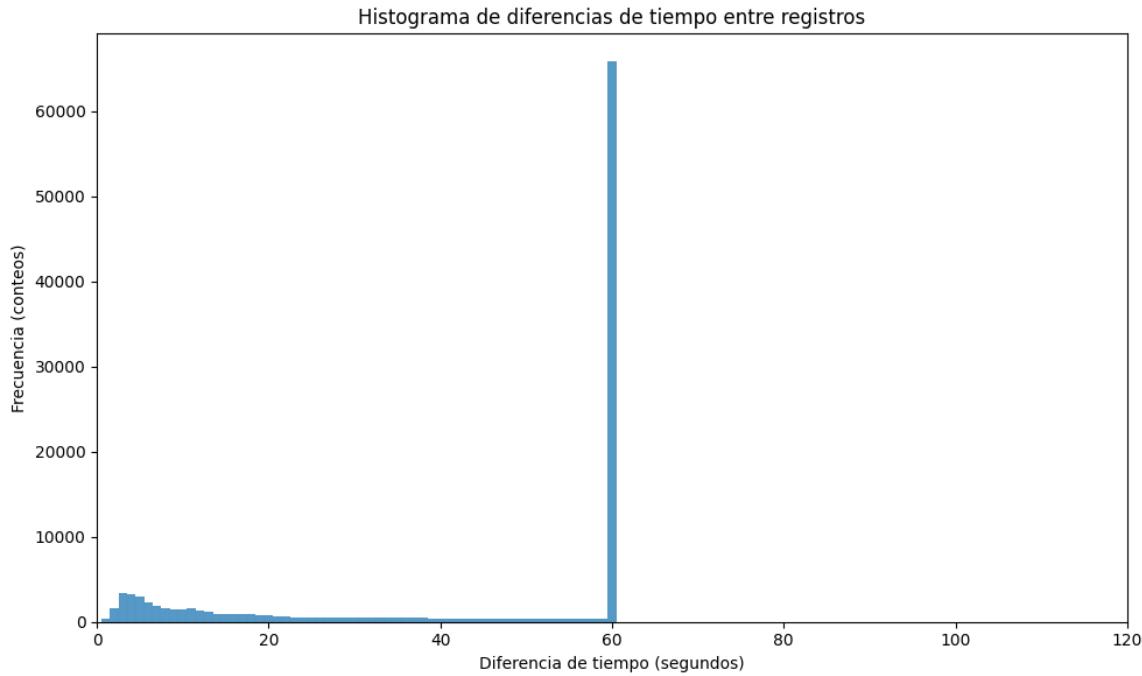


Figura 10: Diferencia de tiempo entre registros consecutivos para la unidad 098.

## 6.4. Ingeniería de características

La ingeniería de características consistió en la creación de nuevas variables a partir de los registros de posición de las unidades y su marca de tiempo correspondiente, con el objetivo de convertir estos datos en un formato adecuado para el entrenamiento de modelos de predicción del tiempo de llegada.

### 6.4.1. Línea en la que está operando la unidad

Se creó una nueva variable que indica la línea en la que está operando la unidad en un momento dado. Esta variable no solo le permite al modelo diferenciar el comportamiento de las distintas líneas, sino que también es la base para la creación de otras características relacionadas con la línea, como la siguiente estación teórica y la distancia restante. Como se mencionó en la sección del análisis exploratorio de datos, en ocasiones, las unidades realizan viajes que abarcan múltiples líneas del sistema Transmetro, por lo que es necesario tomar

en cuenta esta variabilidad. El procedimiento para asignar la línea a cada punto GPS fue el siguiente:

**Definición de una polilínea densa por línea de Transmetro:** Aunque las estaciones de Transmetro ya fueron georreferenciadas durante la limpieza de datos, estas por sí solas no son suficientes para determinar la línea que está realizando una unidad en un viaje específico, ya que, como se observa en la figura 11, la mayoría de las líneas de Transmetro cuentan con estaciones o trayectos que coinciden con otras líneas, ya sea porque se traten de estaciones de transbordo, o simplemente porque cuentan con tramos similares en ciertas partes de sus trayectorias. Por tanto, no es posible asignar de manera acertada una línea al viaje de una unidad basándose únicamente en las estaciones visitadas durante este.

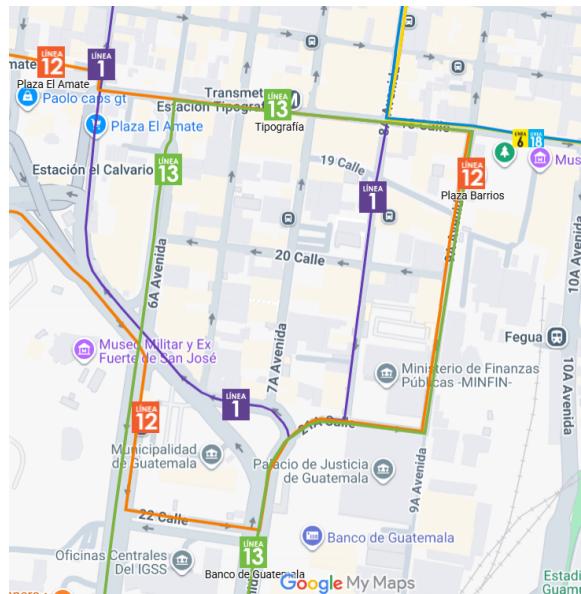


Figura 11: Ejemplo de estaciones de diferentes líneas de Transmetro en trayectorias similares.

Incluso definiendo una polilínea holgada para cada línea de Transmetro, que únicamente conecte las estaciones de dicha línea, la asignación de una línea al viaje de una unidad seguiría siendo ambigua y poco precisa en muchos casos, ya que, como se observa en la figura 12, la asignación de un punto GPS a un segmento u otro de la polilínea holgada no se alinea necesariamente al sentido real del viaje de la unidad, afectando la confiabilidad de la asignación de línea.

Para solventar esta limitación, se creó una polilínea densa para cada línea de Transmetro, haciendo uso de los archivos KML generados por Google Maps a partir de los mapas de las líneas de Transmetro disponibles en la página oficial de la Municipalidad de Guatemala [21]. Al descomponer estos archivos, se obtuvieron una serie vértices con longitud y latitud que representan la ruta completa de cada línea, entre los cuales se incluyeron los puntos correspondientes a las estaciones anteriormente georreferenciadas (véase el Anexo 12.1 para más detalles). Esto permite una asignación más confiable de la línea que está realizando una unidad en un viaje específico.

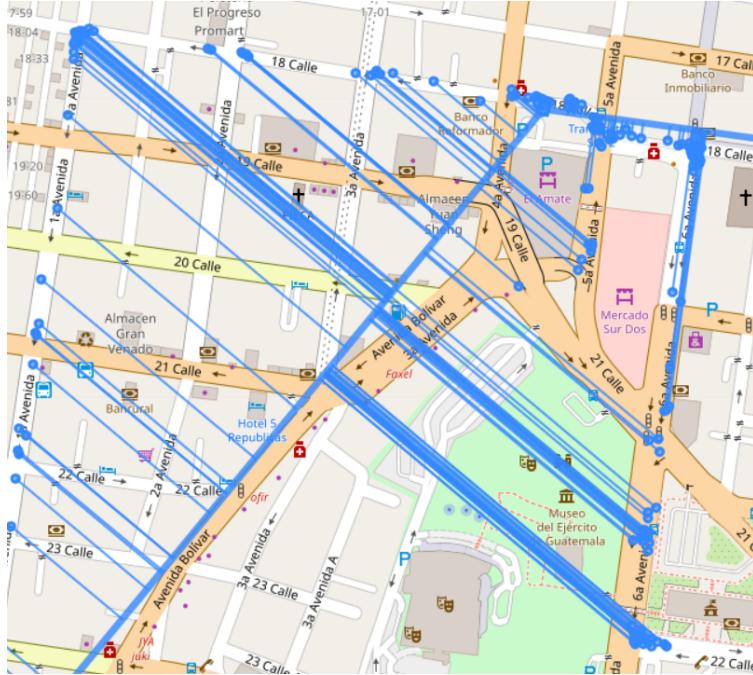


Figura 12: Asignación de puntos GPS de un viaje de la unidad 049 a la polilínea holgada de la línea 12.

**Definición de direcciones:** Asignar una dirección a las líneas de Transmetro es fundamental para evitar que el ruido en los datos asigne un punto GPS a un segmento de la polilínea densa que no corresponde al sentido real del viaje de la unidad, especialmente en aquellos segmentos que se encuentran particularmente cerca entre sí, como trayectos que se recorren en ambos sentidos o intersecciones, ya que esto afectaría la creación de características como la siguiente estación teórica y la distancia restante a esta. Por ejemplo, en la figura 13 se ilustran segmentos de la polilínea de la línea 7 que representan el mismo trayecto pero en sentido opuesto.

Por ello, se definió las direcciones “IDA”, “VUELTA” y “CIRCULAR” para cada línea de Transmetro, dependiendo de su configuración geográfica, diferenciando así aquellos segmentos de la polilínea que se recorren en ambos sentidos o que se encuentran particularmente cerca entre sí, como intersecciones. De esta manera, se evita que la asignación de un segmento a un punto GPS realice “saltos” irrealistas en direcciones opuestas. Esto se realizó asignándole una dirección a cada vértice de la polilínea densa, tomando en cuenta que la estación terminal de una dirección no debe tener más puntos en dicha dirección después de ella, ya que esto resultaría en un *delay* en la inferencia de la próxima estación teórica y la distancia restante, características que se describen en las siguientes secciones. Además, para aquellas estaciones terminales en las que la unidad realiza un giro en U para regresar por la misma ruta, se definió dicha estación terminal como la estación inicial de la dirección opuesta, tal como se observa en la figura 14, ya que esto facilita la inferencia de los cambios de dirección que también se describen en las siguientes secciones.

**Preselección de líneas candidatas:** Para hacer más eficiente el proceso de asignación de línea a cada punto GPS, se realizó una preselección de líneas candidatas según las estaciones

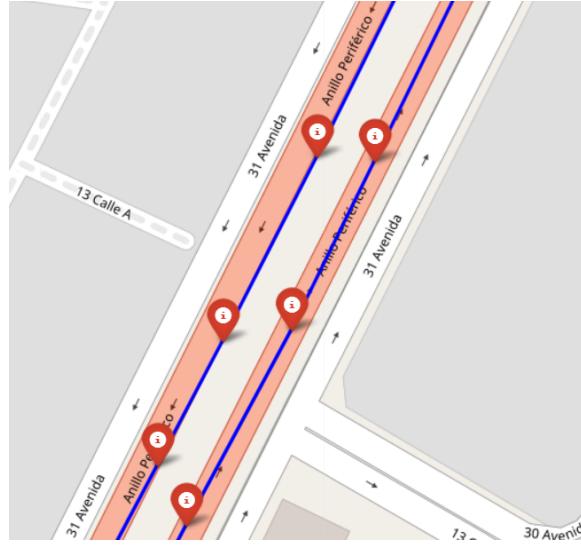


Figura 13: Segmentos de la polilínea densa de la línea 7 que representan el mismo trayecto pero en sentido opuesto.

```

1   kml_file,type,name,order,lat,lon,station,dir
186  doc.kml,Polygon,Linea_12,161,14.5643368,-90.5638924,CENMA,VUELTA
187  doc.kml,Polygon,Linea_12,162,14.5642837,-90.5636148,CENMA,IDA
188  doc.kml,Polygon,Linea_12,163,14.5652609,-90.5622429,IDA

```

Figura 14: Estación terminal con giro en U.

únicas de cada línea visitadas por la unidad en el viaje correspondiente; es decir, aquellas estaciones que no se encuentran en el trayecto de ninguna otra línea de Transmetro. De esta manera, se redujo el número de líneas a considerar para cada punto GPS. En la figura 15 se muestra un ejemplo de estaciones únicas de la línea 6.

Las estaciones únicas definidas para cada línea son las siguientes:

- Línea 1: San Agustín
- Línea 2: Jocotenango
- Línea 6: Centro zona 6
- Línea 7: Cejusa dirección la Merced, Cejusa dirección USAC, Villa Linda andén Norte y Villa Linda andén Sur
- Línea 13-A: Terminal, Industria, Tívoli y Torre del Reformador
- Línea 13-B: Plaza Berlín, Juan Pablo II y Plaza Argentina
- Línea 18-A: Atlántida
- Línea 18-B: San Rafael y Paraíso
- Línea 12: *Fallback* si no se visitan estaciones únicas de otras líneas

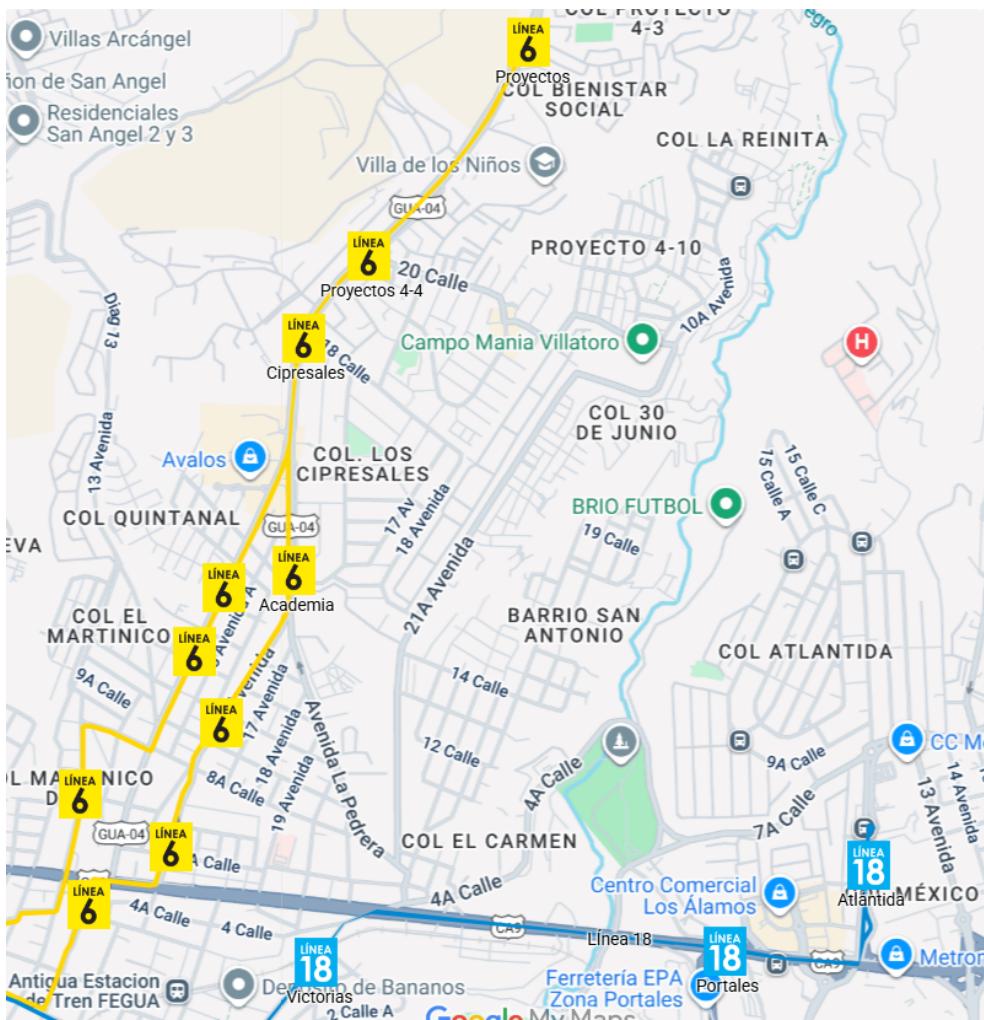


Figura 15: Ejemplo de estaciones únicas de la línea 6 de Transmetro. Fuente: Municipalidad de Guatemala [21].

**Proyección de puntos GPS sobre polilíneas densas:** El siguiente paso en la asignación de una línea a cada viaje de cada unidad fue proyectar cada punto GPS de este a la polilínea densa correspondiente de cada línea-dirección candidata. Primero, los vértices de cada línea-dirección candidata se transformaron a coordenadas cartesianas locales (metros) respecto a un origen  $(\varphi_0, \lambda_0)$ , tomado como el promedio de la polilínea densa de la línea-dirección correspondiente. Los puntos GPS de cada viaje se transformaron del mismo modo, reutilizando el  $(\varphi_0, \lambda_0)$  específico de cada línea-dirección candidata, para asegurar que punto y polilínea queden en el mismo marco local. Luego, utilizando la proyección punto-segmento, cada punto GPS se proyecta sobre la polilínea densa de cada línea-dirección candidata. También se computa la distancia acumulada (en metros) desde el inicio de la polilínea hasta el pie de proyección.

**Modelo de Markov oculto con distancias:** Con la distancia mínima de cada punto GPS a cada línea-dirección candidata y la distancia acumulada del pie de proyección, se define cada línea-dirección candidata como un estado  $k = (\text{línea}, \text{dirección})$ , con un costo de emisión:

$$e_{j,k} = \min(d_{j,k}, D_{max})^2$$

donde  $e_{j,k}$  es el costo de emisión para el estado  $k$  dado el punto GPS  $P_j$ ,  $d_{j,k}$  es la distancia del punto GPS  $P_j$  a la línea-dirección candidata  $k$  y  $D_{max} = 120$  m es un umbral de distancia máxima para evitar penalizaciones excesivas por lecturas ruidosas o puntos fuera de ruta.

La asignación punto a punto independiente es inestable cerca de los cruces entre líneas o segmentos paralelos mencionados durante la definición de polilíneas densas. Para introducir coherencia espacial y temporal en la secuencia de estados asignados a los puntos  $\{P_j\}_{j=1}^N$  de un viaje, se formula el problema como un modelo oculto de Markov (HMM) con:

- Estados: las líneas-direcciones candidatas  $k$ .
- Observaciones: los puntos GPS  $P_j$ .
- Costos de emisión:  $e_{j,k}$  definidos arriba.
- Costos de transición entre estados:

$$J(x_{1:N}) = \sum_{j=1}^N e_{j,x_j} + \sum_{j=2}^N \pi_j(x_{j-1}, x_j)$$

donde  $J(x_{1:N})$  es la función objetivo a minimizar (el costo total de la asignación de una línea-candidata  $x_j$  a cada punto GPS  $P_j$  desde el inicio del viaje hasta el punto  $P_N$ ) y  $\pi_j(x_{j-1}, x_j)$  es una penalización de cambio de línea-dirección que se detalla más adelante.

**Penalización de cambio de línea-dirección:** Para fomentar secuencias coherentes de línea-dirección, se parte de una penalización base  $\lambda_{sw} = 60$  para cualquier cambio  $x_j \neq x_{j-1}$

y 0 en caso contrario. Esta penalización se modula dinámicamente según la cercanía del punto proyectado  $P_j^*$  a la estación terminal de la línea–dirección actual  $x_{j-1}$ :

$$\pi_j(k \rightarrow s) = \begin{cases} 0, & \text{si } k = s, \\ \alpha_{j,k} \cdot \lambda_{sw}, & \text{si } k \neq s, \end{cases}$$

$$\alpha_{j,k} = \begin{cases} 0.3, & \text{si } L_k - s_{j,k} \leq W_{term} \\ 1.2, & \text{si } k \text{ y } s \text{ son de la misma línea pero diferente dirección} \\ 1.0, & \text{en otro caso} \end{cases}$$

donde  $k$  es la línea–dirección actual,  $s$  es la posible línea–dirección siguiente,  $L_k$  es la longitud total de la polilínea de la línea–dirección actual,  $s_{j,k}$  es la distancia acumulada del punto proyectado  $P_j^*$  sobre dicha polilínea, y  $W_{term} = 50\text{m}$  es una ventana alrededor de las estaciones terminales, donde se reduce la penalización para facilitar cambios de línea–dirección.

La solución óptima  $x_{1:N}^*$  que minimiza  $J(x_{1:N})$  se encuentra con el algoritmo de Viterbi.

**Segmentación del resultado:** El resultado de Viterbi es una secuencia de líneas - direcciones; sin embargo, para facilitar análisis posteriores, se colapsa esta secuencia en tramos contiguos de la misma línea, ignorando la dirección (histéresis por línea). Es decir, se agrupan segmentos consecutivos asignados a la misma línea, independientemente de su dirección, para formar tramos homogéneos, ya que en la creación de las siguientes características (siguiente estación teórica y distancia restante) se realiza una inferencia de dirección por registro más robusta:

1. Se agrupan  $[i_0, i_1]$  de índices consecutivos de cada viaje de cada unidad con la misma **línea** indicada por el resultado de Viterbi (aunque la dirección varíe).
2. Se descartan bloques cortos (menos de registros) y con bajo progreso sobre la polilínea  $\Delta s < 300\text{m}$ , donde

$$\Delta s = \max(s_{j,k}) - \min(s_{j,k})$$

para todos los puntos  $P_j$  dentro del bloque  $k$ .

3. Para cada bloque aceptado se define una dirección de arranque **DIR\_init** como la moda de las direcciones observadas en una ventana inicial de  $W = 40$  puntos. Esta etiqueta sirve para encadenar, en pasos posteriores, la siguiente estación teórica y la distancia restante a esta.

El artefacto intermedio almacenado por cada viaje de cada unidad contiene: **trip\_id**, **LINEA**, **DIR\_init**, **t\_start**, **t\_end**, **t\_start\_ts**, **t\_end\_ts**, **idx\_start**, **idx\_end**, **dur\_pts**, donde **trip\_id** es el identificador del viaje, **LINEA** es la línea asignada al bloque, **DIR\_init** es la dirección inicial del bloque, **t\_start** y **t\_end** son índices relativos al viaje que indican el inicio y fin del bloque, **t\_start\_ts** y **t\_end\_ts** son las marcas de tiempo de inicio y fin del bloque, **idx\_start** e **idx\_end** son los índices globales de inicio y fin del bloque, y **dur\_pts**

es la duración del bloque en número de registros. Cabe mencionar que la denotación **bloque** se refiere a un tramo contiguo de un viaje de una unidad en el que esta está operando en una misma línea.

Con este artefacto, se asignaron las líneas a cada punto GPS de los viajes de las unidades, utilizando los índices globales `idx_start` e `idx_end` para mapear los bloques a los puntos GPS correspondientes.

#### 6.4.2. Siguiente estación teórica y distancia restante

Con la línea asignada a cada punto GPS, se creó una nueva variable que indica la siguiente estación teórica a la que se dirige la unidad en función de su posición actual en la línea. Esta variable es fundamental para estimar el tiempo de llegada, ya que permite al modelo comprender el progreso de la unidad a lo largo de su ruta y anticipar las estaciones futuras. El procedimiento para calcular la siguiente estación teórica y la distancia restante fue el siguiente:

**Estaciones sobre las polilíneas densas:** Para cada posible línea-dirección  $(l, d)$ , se transformaron las coordenadas geográficas (latitud y longitud) de los vértices usando la misma transformación local en metros mencionada anteriormente. Para aquellos vértices que representan estaciones, se calculó su abscisa curvilínea  $s_{est} \in [0, L_{l,d}]$  sobre la polilínea densa correspondiente. El resultado es una secuencia ordenada

$$S_{l,d} = \{(est_1, s_{est_1}), \dots, (est_M, s_{est_M})\} \text{ con } 0 \leq s_{est_1} < \dots < s_{est_M} \leq L_{l,d},$$

donde  $est_i$  es el nombre de la estación  $i$ ,  $s_{est_i}$  su abscisa curvilínea,  $M$  el número de estaciones en la línea-dirección  $(l, d)$  y  $L_{l,d}$  la longitud total de la polilínea densa de dicha línea-dirección.

**Proyección del viaje y obtención de s:** Dado un bloque estable por línea  $l$  (sección previa), se asume una dirección inicial `DIR_init`  $\in \{\text{"IDA"}, \text{"VUELTA"}, \text{"CIRCULAR"}\}$  y se proyectan los puntos GPS del bloque sobre la polilínea densa de la línea-dirección  $(l, \text{DIR\_init})$  usando el mismo procedimiento descrito anteriormente, obteniendo, por punto  $P_j$ , la distancia lateral mínima  $d_j$  y la abscisa curvilínea  $s_j$ . Para robustez temporal, se usa una ventana local de segmentos alrededor del último segmento aceptado y se añade histéresis de no-retroceso:

$$s_j = \max(s_j, s_{j-1} - \epsilon_{back})$$

donde  $\epsilon_{back} = 20$  m es una tolerancia a pequeñas regresiones numéricas

**Regla de próxima estación:** Para cada  $s_j$ , se define la siguiente estación teórica  $k^*$  como la primera estación por delante en la misma dirección:

$$k^*(j) = \min\{k \in S_{l,\text{DIR\_init}} : s_k > s_j\}$$

y se asigna:

$$\text{next\_station}_j = est_{k^*(j)}$$

La distancia restante a dicha estación se calcula como:

$$\Delta_j = \begin{cases} s_{est_{k^*(j)}} - s_j, & \text{si } k^*(j) \leq M, \\ \max(0, s_{est_M} - s_j), & \text{si } j \text{ ya pasó la última estación en la ruta} \\ L_{l,\text{DIR\_init}} - s_j, & \text{si la ruta es circular} \end{cases}$$

En la práctica, se utilizó `searchsorted` de la librería numpy sobre  $s_{est}$  para encontrar eficientemente la siguiente estación teórica.

**Continuidad IDA/VUELTA y cambios de dirección:** Aunque la segmentación principal agrupa por **línea**, aquí se necesita continuidad por **dirección**. Para las líneas con direcciones “IDA” y “VUELTA”, el algoritmo inicia con **DIR\_init** y proyecta secuencias consecutivas hasta detectar un cambio plausible de dirección. Se consideran dos mecanismos complementarios:

- **Cambio en estación terminal diferente de estación inicial:** cuando el punto  $j$  está cerca del final de la polilínea actual ( $L_{l,d} - s_j \leq W_{end}$ , con  $W_{end} = 50\text{m}$ ) y la siguiente estación teórica es la estación inicial de la dirección opuesta, se comuta a dicha dirección.
- **Cambio en la misma estación (terminal compartida):** si la última estación de la dirección actual coincide con la primera de la dirección opuesta, se habilita un cambio en estación confirmado por una ventana corta de evidencia de movimiento en sentido contrario; i.e., al menos 3 vectores de desplazamiento con producto punto negativo respecto al último segmento de la polilínea correspondiente a la línea-dirección actual, evitando así disparos por ruido.
- **Proyección “muy atrás” respecto al historial:** sea  $j_{\min}$  el índice del segmento cuya proyección minimiza la distancia en el punto actual y  $j_{\text{prev}}$  el índice del segmento más reciente aceptado. Si

$$j_{\min} < j_{\text{prev}} - \kappa_{\text{back}},$$

con un umbral discreto  $\kappa_{\text{back}} = 3$  segmentos, se interpreta como evidencia de que el vehículo ya “giró” y la mejor proyección pertenece al tramo de la dirección opuesta. Esto resulta útil en giros inesperados de la unidad cuando no sigue la línea teórica, ya sea por registros muy lejos entre sí temporalmente o por cambios abruptos en la trayectoria.

Una vez comutada la dirección, se continúa proyectando el resto del bloque sobre la polilínea de la dirección opuesta y se repite la regla de próxima estación. Este esquema impide mezclar estaciones de “IDA” y “VUELTA” en un mismo marco  $s$ .

**Inicialización antes del punto estable:** Para los índices previos al inicio estable del bloque (primer índice con proyección confiable), se rellena la siguiente estación teórica y la distancia restante con los valores del primer punto estable, asegurando continuidad en toda la secuencia del viaje.

**Salida y columnas generadas:** Para cada registro del bloque se producen: `s_m`, `dist_m`, `proxima_est_teorica`, `dist_a_prox_m`, `DIR`, donde `s_m` es la abscisa curvilínea en dicho punto, `dist_m` es la distancia lateral mínima a la polilínea, `proxima_est_teorica` es la siguiente estación teórica, `dist_a_prox_m` es la distancia restante a dicha estación, y `DIR` es la dirección actual (“IDA”, “VUELTA” o “CIRCULAR”).

#### 6.4.3. Tiempo de permanencia

El tiempo de permanencia (*dwell time*) captura intervalos donde la unidad permanece detenida o avanza por debajo de un umbral mínimo, útil para identificar paradas en estaciones, congestión o *idling* operativo. Se computa directamente a partir de la cinemática de los puntos GPS, sin depender de eventos externos.

**Métrica de progreso y regla de no-avance:** Sea  $t_i$  el tiempo del punto  $i$  y  $(x_i, y_i)$  sus coordenadas locales. La distancia euclídea entre puntos consecutivos es

$$\Delta s_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}.$$

Se define  $\text{progreso}_i = |\Delta s_i|$  y un umbral  $m_{\text{prog}} = 8.0\text{m}$ . Entonces,

$$\text{is\_no\_progress}_i = \begin{cases} 1, & \text{si } \text{progreso}_i < m_{\text{prog}}, \\ 0, & \text{en otro caso.} \end{cases}$$

Si hubo progreso suficiente, el contador de permanencia se reinicia; de lo contrario, acumula el tiempo transcurrido:

$$\text{dwell\_time\_xy\_s}_i = \begin{cases} 0, & \text{si } \text{is\_no\_progress}_i = 0, \\ \text{dwell\_time\_xy\_s}_{i-1} + (t_i - t_{i-1}), & \text{si } \text{is\_no\_progress}_i = 1. \end{cases}$$

**Columnas generadas:** El procedimiento añade, por registro, la columna `dwell_time_xy_s` que indica el tiempo de permanencia en segundos y  $\text{is\_no\_progress} \in \{0, 1\}$  que indica si hubo progreso suficiente en el punto.

#### 6.4.4. Velocidad instantánea

A partir de la columna original del conjunto de datos “Velocidad (km/h)”, se calcula la velocidad instantánea en metros por segundo (m/s) utilizando la conversión:

$$\text{Velocidad (m/s)} = \frac{\text{Velocidad (km/h)}}{3.6}$$

Esta nueva columna, denominada `vel_mps`, es únicamente la velocidad instantánea en m/s de la unidad en cada registro. No describe el comportamiento de la unidad en un segmento de la ruta de manera tan robusta como lo haría una velocidad promedio, pero es una característica útil para los modelos de predicción, ya que proporciona información sobre la rapidez con la que la unidad se está desplazando en un momento y posición dados.

#### 6.4.5. Características derivadas de tiempo

Utilizando la columna original del conjunto de datos “Fecha”, que contiene la marca de tiempo de cada registro en formato de fecha y hora, se extrajeron varias características temporales que pueden ser relevantes para el análisis y predicción del tiempo de llegada de las unidades, especialmente en lo que respecta a patrones de tráfico y demanda en diferentes momentos del día y días de la semana. Las características derivadas son las siguientes:

- `hour`: La hora del día en formato de 24 horas (0-23).
- `dow`: El día de la semana (0-6), donde 0 representa el lunes y 6 el domingo.
- `is_weekend`: Una variable binaria que indica si el día es fin de semana (1) o día laborable (0).
- `is_peak_hour`: Una variable binaria que indica si la hora del día corresponde a una hora pico de tráfico (1) o no (0). Se definieron las horas pico como 7:00-9:00 y 16:00-19:00.

### 6.5. Cálculo de la variable objetivo

La variable objetivo a predecir es el tiempo restante (en segundos) para que una unidad llegue a su siguiente estación teórica, definida en la sección anterior. Para cada muestra temporal  $j$ :

$$\text{ETA}_j = t_{arr}(\text{prox\_estacion}_j) - t_j$$

donde  $t_j$  es el tiempo del punto  $j$  y  $t_{arr}(\text{prox\_estacion}_j)$  es el tiempo de llegada real a la siguiente estación teórica, obtenido buscando el primer punto en el viaje con diferente `prox_estacion` y tomando su tiempo  $t$ . En el conjunto de datos se registra en segundos como la columna `ETA_prox_est_s` y en minutos como `ETA_prox_est_min`.

**Agrupación y orden temporal:** El cálculo se realiza por grupos (`trip_id`, `block_id`), ordenando los puntos por `Fecha` ascendente para asegurar que la búsqueda de la siguiente estación teórica sea correcta.

Sea  $\text{prox\_estacion}[j]$  la etiqueta de la siguiente estación teórica en el punto  $j$ . Se descompone la secuencia del bloque en *runs* de valores  $\text{prox\_estacion}[j]$  iguales:

$$\underbrace{\text{prox\_estacion}_j = A, \text{prox\_estacion}_j = A \dots}_{\text{run 1}} \quad \underbrace{\text{prox\_estacion}_j = B, \text{prox\_estacion}_j = B \dots}_{\text{run 2}}$$

Cada run  $r$  se representa como un intervalo  $[a_r, b_r]$ . Esta estructura permite asignar un único instante de llegada a todo el run.

**Detección del arribo:** Para cada run  $r$  se toma el primer punto después del run con diferente `prox_estacion`. El último run no tiene punto siguiente, por lo que se asigna un ETA `none` a sus puntos. Además, ETAs negativas (posibles por ruido de ordenamiento o empates temporales) se marcan como no disponibles (`none`).

**Columnas generadas:** El procedimiento añade: `ETA_prox_est_s` y `ETA_prox_est_min`, que representan el tiempo restante para llegar a la siguiente estación teórica en segundos y minutos, respectivamente.

## 6.6. Descarte de datos atípicos

Una vez calculado el tiempo de permanencia y la variable objetivo, se procedió a descartar aquellos registros que presentaban valores atípicos o inconsistentes. En la figura 16 se grafican registros de un viaje de la unidad 204 que abarcan aproximadamente 2 horas, seguidos de registros que reflejan su trayectoria habitual. Esto indica que, durante este tiempo, la unidad no realizó movimientos que reflejan su comportamiento operativo.

Los criterios utilizados para identificar y eliminar registros que no representan el comportamiento habitual de las unidades fueron los siguientes:

- Registros con valores nulos o faltantes en la variable objetivo o en las características calculadas durante el proceso de ingeniería de características.
- Registros con tiempos de permanencia anormalmente altos, definidos como aquellos mayores a 300 segundos (5 minutos).
- En caso la unidad sí haya presentado movimiento durante 5 minutos, pero no se acerca a su siguiente estación teórica, tal como en la figura 16, se descartan aquellos registros con tiempos de llegada excesivamente altos, definidos como aquellos mayores a 7200 segundos (2 horas).

Es importante mencionar que estos descartes pueden replicarse cuando el modelo ya esté implementado y se encuentre realizando predicciones constantemente en base a registros nuevos, filtrando aquellos que no representen el comportamiento habitual de las unidades.

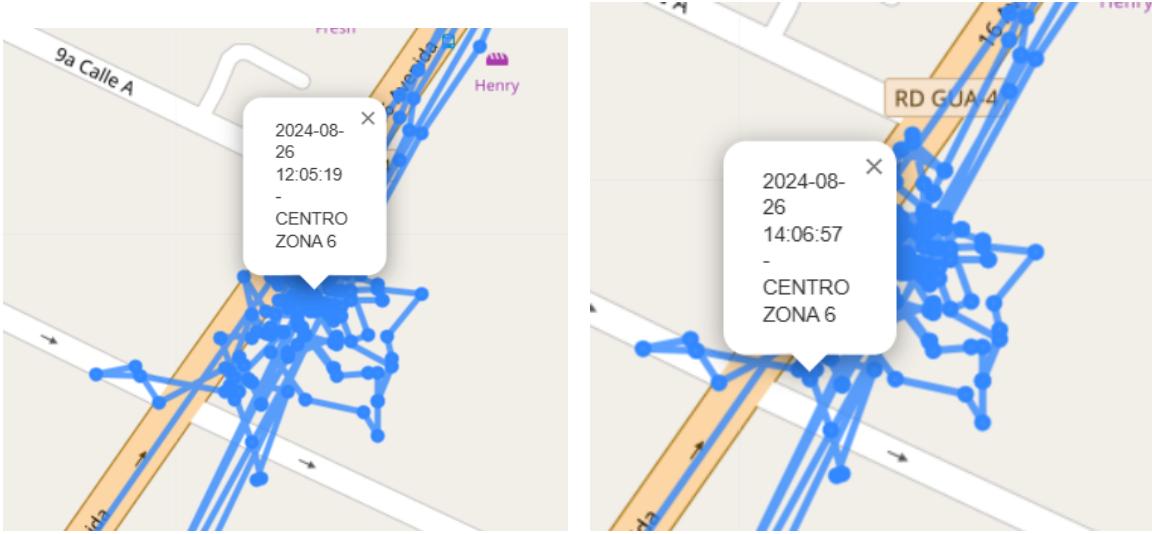


Figura 16: Mapas de registros en los que la unidad 204 no realizó movimientos que reflejan su comportamiento operativo.

## 6.7. Muestreo representativo del conjunto de datos

Dada la naturaleza repetitiva de los datos y a la gran cantidad de estos, se realizó un muestreo estratificado para reducir el tiempo de entrenamiento de los modelos. Este muestreo consideró tanto la línea de Transmetro indicada en cada registro como la secuencia temporal de los registros, asegurando una cantidad representativa de la proporción original de registros por línea y manteniendo viajes completos en el subconjunto seleccionado.

**Estrategia de estratificación:** Como se mencionó anteriormente, el muestreo está estratificado en dos niveles:

1. **Estrato por línea (LINEA):** garantiza cobertura mínima por cada línea del sistema.
2. **Unidad de muestreo:** se seleccionan bloques completos dentro de cada línea, en lugar de filas aisladas, para conservar la estructura secuencial y la dinámica de tránsito (velocidad, detenciones, progresión), permitiendo que los modelos aprendan patrones temporales y espaciales en días completos de operación. Un bloque es una secuencia contigua de registros pertenecientes a un mismo viaje de una unidad específica en una línea determinada.

**Cuotas por línea:** Sea  $C_\ell$  la capacidad de la línea  $\ell$  (número total de filas en todos sus bloques) y  $T = 5,000,000$  el tamaño de muestra objetivo (número total de filas a obtener). Debido a las diferencias entre las proporciones de registros correspondientes a cada línea que se observa en el cuadro 2, se define una base proporcional suavizada  $B_\ell = \sqrt{C_\ell}$  para reducir el dominio de aquellas líneas con las que se cuentan muchos más registros, evitando así que sobre-representen el comportamiento del sistema de Transmetro durante el entrenamiento

de los modelos. En base a esta se define una cuota preliminar  $\hat{q}_\ell$ :

$$\hat{q}_\ell = T \cdot \frac{B_\ell}{\sum_j B_j}.$$

Esta cuota preliminar se aproxima al menor entero y se le impone un mínimo garantizado de registros por línea  $m = 10,000$  y un tope por la capacidad de la línea correspondiente:

$$q_\ell = \min(\max(\lfloor \hat{q}_\ell \rfloor, m), C_\ell).$$

Cuadro 2: Número de registros por línea en el conjunto de datos preprocesado.

Línea	Registros	Proporción (%)
Línea 1	2,138,693	4.97
Línea 2	549,587	1.28
Línea 6	5,460,214	12.69
Línea 7	506,274	1.18
Línea 12	18,864,271	43.84
Línea 13-A	9,778,243	22.73
Línea 13-B	80,323	0.19
Línea 18-A	4,818,382	11.20
Línea 18-B	834,939	1.94
<b>Total</b>	<b>43,030,926</b>	<b>100.0</b>

**Selección aleatoria de bloques:** Dentro de cada línea  $\ell$  se seleccionan bloques completos de manera aleatoria hasta alcanzar su cuota  $q_\ell$  correspondiente. Sea  $n_b$  el número de registros en el bloque  $b$  y  $q_\ell$  la cuota de la línea  $\ell$ . Para cada línea  $\ell$ :

1. Se baraja aleatoriamente el orden de sus bloques (con una semilla SEED = 25 fija para reproducibilidad);
2. Se acumulan bloques completos (sumando sus tamaños  $n_b$ ) hasta cubrir  $q_\ell$ ;
3. Se registra el total seleccionado y la diferencia entre el total seleccionado y la cuota previamente establecida.

Esta política *greedy* por bloques prioriza secuencias completas y evita romper continuidad temporal, a costa de pequeñas desviaciones respecto a  $q_\ell$  cuando los bloques seleccionados son muy heterogéneos en longitud. En el cuadro 3 se resumen los resultados del muestreo por línea.

Cuadro 3: Resumen del muestreo estratificado por línea.

Línea	Cuota	Seleccionados	Diferencia
Línea 1	453,759	453,762	3
Línea 2	230,022	230,817	795
Línea 6	725,030	725,071	41
Línea 7	220,772	220,959	187
Línea 12	1,347,632	1,347,834	202
Línea 13-A	970,245	970,476	231
Línea 13-B	13,042	13,042	0
Línea 18-A	681,086	682,399	1,313
Línea 18-B	283,517	283,820	303

**Construcción de la muestra fila a fila:** Una vez elegidos los bloques de cada línea, se utilizan los campos `{Placa, trip_id, block_id}` para filtrar todos los archivos CSV de las unidades, conservando únicamente las filas pertenecientes a esos bloques en la muestra final. Luego se estandarizan los tipos tanto de las variables originales, como de las características definidas anteriormente (categóricas, fechas, numéricas) antes de persistir la muestra en un archivo `sample_features.parquet`.

## 6.8. Entrenamiento de modelos: LightGBM y XGBoost

Se entrenaron modelos de regresión para estimar el tiempo de llegada a la próxima estación en segundos. Los algoritmos utilizados fueron LightGBM y XGBoost, algoritmos de *boosting* basados en árboles de decisión. Se utilizó validación cruzada para evaluar el desempeño de los modelos y evitar sobreajuste.

### 6.8.1. Partición de datos para entrenamiento y validación

En primer lugar, se dividió el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba. El conjunto de prueba se reservó para la evaluación final del modelo y no se utilizó durante el entrenamiento. El conjunto de entrenamiento se definió con datos desde el 1 de enero de 2024 hasta el 31 de marzo de 2025, mientras que el conjunto de prueba abarcó datos desde el 1 de abril de 2025 hasta el 30 de abril de 2025.

### 6.8.2. Validación cruzada temporal

Se realizó validación cruzada para (i) obtener una estimación robusta del error en comparación con la que se obtendría con un único conjunto de validación, (ii) elegir el número de iteraciones de *boosting* de cada modelo con evidencia y (iii) reducir el sobreajuste a un único conjunto de validación. La validación cruzada realizada es temporal; es decir, los *folds* de validación en los que se divide el conjunto de entrenamiento están basados en fechas, asegurando que los datos de entrenamiento siempre precedieran temporalmente a los datos

de validación. También cabe mencionar que la división del conjunto de datos se realizó a nivel de viaje completo (`trip_id`) para evitar fugas de información entre los conjuntos de entrenamiento y validación. Se definieron 15 *folds* de la siguiente manera:

- Fold 1: Entrenamiento con datos del 1 de enero de 2024 al 30 de enero de 2024, validación con datos del 31 de enero de 2024 al 4 de febrero de 2024.
- Fold 2: Entrenamiento con datos del 1 de enero de 2024 al 29 de febrero de 2024, validación con datos del 1 de marzo de 2024 al 5 de marzo de 2024.
- Fold 3: Entrenamiento con datos del 1 de enero de 2024 al 30 de marzo de 2024, validación con datos del 31 de marzo de 2024 al 4 de abril de 2024.
- Fold 4: Entrenamiento con datos del 1 de enero de 2024 al 29 de abril de 2024, validación con datos del 30 de abril de 2024 al 4 de mayo de 2024.
- Fold 5: Entrenamiento con datos del 1 de enero de 2024 al 29 de mayo de 2024, validación con datos del 30 de mayo de 2024 al 3 de junio de 2024.
- Fold 6: Entrenamiento con datos del 1 de enero de 2024 al 28 de junio de 2024, validación con datos del 29 de junio de 2024 al 3 de julio de 2024.
- Fold 7: Entrenamiento con datos del 1 de enero de 2024 al 28 de julio de 2024, validación con datos del 29 de julio de 2024 al 2 de agosto de 2024.
- Fold 8: Entrenamiento con datos del 1 de enero de 2024 al 27 de agosto de 2024, validación con datos del 28 de agosto de 2024 al 1 de septiembre de 2024.
- Fold 9: Entrenamiento con datos del 1 de enero de 2024 al 26 de septiembre de 2024, validación con datos del 27 de septiembre de 2024 al 1 de octubre de 2024.
- Fold 10: Entrenamiento con datos del 1 de enero de 2024 al 26 de octubre de 2024, validación con datos del 27 de octubre de 2024 al 31 de octubre de 2024.
- Fold 11: Entrenamiento con datos del 1 de enero de 2024 al 25 de noviembre de 2024, validación con datos del 26 de noviembre de 2024 al 30 de noviembre de 2024.
- Fold 12: Entrenamiento con datos del 1 de enero de 2024 al 25 de diciembre de 2024, validación con datos del 26 de diciembre de 2024 al 30 de diciembre de 2024.
- Fold 13: Entrenamiento con datos del 1 de enero de 2024 al 24 de enero de 2025, validación con datos del 25 de enero de 2025 al 29 de enero de 2025.
- Fold 14: Entrenamiento con datos del 1 de enero de 2024 al 23 de febrero de 2025, validación con datos del 24 de febrero de 2025 al 28 de febrero de 2025.
- Fold 15: Entrenamiento con datos del 1 de enero de 2024 al 26 de marzo de 2025, validación con datos del 27 de marzo de 2025 al 31 de marzo de 2025.

Este enfoque de validación temporal permite evaluar el desempeño del modelo en condiciones similares a las que enfrentaría en producción, donde la predicción se realiza sobre registros nuevos constantemente.

### 6.8.3. Métricas de evaluación

Para evaluar el desempeño de los modelos tanto durante la validación cruzada como en la evaluación final sobre el conjunto de prueba, se utilizaron las siguientes métricas:

- **Error absoluto medio:** Métrica principal utilizada por su interpretabilidad directa en segundos y su robustez frente a valores atípicos, los cuales pudieron observarse durante el cálculo del tiempo de permanencia y de la variable objetivo.
- **Error cuadrático medio:** Métrica complementaria que penaliza más fuertemente los errores grandes, proporcionando una visión adicional del desempeño del modelo.
- **Acuerdos de nivel de servicio:** Se calcularon los porcentajes de predicciones con errores absolutos menores a 60, 120 y 180 segundos. Estas métricas son relevantes para evaluar la utilidad práctica del modelo en escenarios operativos.

### 6.8.4. Entrenamiento con LightGBM

En el cuadro 4 se presentan los hiperparámetros utilizados para entrenar el modelo LightGBM, tanto en la validación cruzada como en el entrenamiento final sobre todo el conjunto de entrenamiento.

Hiperparámetro	Valor
objective	regression
metric	{mae, rmse}
learning_rate	0.05
num_leaves	64
max_depth	por defecto
min_data_in_leaf	200
feature_fraction	0.8
bagging_fraction	0.8
bagging_freq	1
lambda_l1, lambda_l2	0.1, 0.1
max_bin	255
seed	25

Cuadro 4: Configuración principal de LightGBM.

Por otro lado, durante la validación cruzada se definió un *early stopping* de 300 iteraciones y un *num\_boost\_rounds* de 3,000 iteraciones.

### 6.8.5. Entrenamiento con XGBoost

En el cuadro 5 se presentan los hiperparámetros utilizados para entrenar el modelo XGBoost, tanto en la validación cruzada como en el entrenamiento final sobre todo el conjunto de entrenamiento.

Hiperparámetro	Valor
objective	reg:squarederror
eval_metric	{mae, rmse}
learning_rate	0.05
max_depth	6
min_child_weight	200
subsample	0.8
colsample_bytree	0.8
reg_lambda, reg_alpha	0.1, 0.1
tree_method	hist
max_bin	255
random_state	25

Cuadro 5: Configuración principal de XGBoost.

Al igual que en el caso de LightGBM, durante la validación cruzada se definió un *early stopping* de 300 iteraciones y un *num\_boost\_rounds* de 3,000 iteraciones.

#### 6.8.6. Selección de número de iteraciones para los modelos finales

Para seleccionar el número óptimo de iteraciones para el entrenamiento de los modelos finales de LightGBM y XGBoost, se fijó *num\_boost\_rounds* a la mediana de las mejores iteraciones obtenidas en cada *fold* de la validación cruzada y se entrenó sobre todo el conjunto de entrenamiento. Luego, se evaluó el desempeño en el conjunto de prueba reservado inicialmente.

#### 6.9. Complementación de los modelos con estadísticas históricas

Para estimar el tiempo de llegada de una unidad a una estación más allá de su siguiente estación, es posible complementar la predicción de los modelos con tiempos históricos robustos por tramo. Utilizando el conjunto de datos preprocesado, se calculan dos nociones del tiempo de llegada entre estaciones:

- **A2A (*arrival to arrival*):** Tiempo desde la llegada a la `prox_est_teoricaj` hasta la llegada a la `prox_est_teoricaj+1`. Este tiempo incluye tanto el tiempo de viaje como el tiempo de permanencia en la estación intermedia.
- **D2A (*departure to arrival*):** Tiempo desde la salida de la `prox_est_teoricaj` hasta la llegada a la `prox_est_teoricaj+1`. Este tiempo excluye el tiempo de permanencia, considerando únicamente el tiempo de viaje entre estaciones.

**Segmentación de viajes en bloques de misma próxima estación teórica:** Se leen todos los archivos CSV preprocesados y los registros se ordenan estrictamente por LINEA,

`Placa`, `trip_id`, `Fecha` para preservar la secuencia temporal dentro de cada viaje. Luego, para cada viaje `LINEA`, `Placa`, `trip_id`, se generan bloques consecutivos de registros con la misma `prox_est_teorica`, de la misma manera en la que se hizo para calcular la variable objetivo. Con esto se obtienen bloques  $[a_r, b_r]$  donde todos los registros tienen la misma próxima estación teórica.

**Detección del instante de llegada a una estación:** Para cada bloque  $r$  se identifica el instante de llegada a la estación teórica correspondiente con reglas de robustez:

- Umbral de distancia a la próxima estación teórica:  $\text{dist\_a\_prox\_m} \leq D$ , con  $D = 20\text{ m}$ .
- Umbral de tiempo estimado de llegada:  $\text{ETA\_prox\_est\_s} \leq T$ , con  $T = 20\text{ s}$ .
- Si ninguno de los dos anteriores se cumple, como *fallback* se toma el tiempo del último punto del bloque como instante de llegada para garantizar continuidad.

Esto produce una secuencia ordenada de pares

$$\text{prox\_est\_teorica} t_{arr}(\text{prox\_est\_teorica})$$

para cada viaje.

**Detección del instante de salida de una estación:** Para cada bloque  $r$  se obtiene el primer registro con progreso después del instante de llegada identificado previamente. Este registro no debe presentar la bandera de bandera de no-progreso: `is_no_progress = 0`. El tiempo de este registro se toma como el instante de salida  $t_{dep}(\text{prox\_est\_teorica})$  de la estación teórica correspondiente. Esto genera una secuencia ordenada de pares

$$\text{prox\_est\_teorica} t_{dep}(\text{prox\_est\_teorica})$$

para cada viaje.

Si no se encuentra ningún registro que cumpla las condiciones mencionadas antes del  $t_{arr}(\text{prox\_est\_teorica})$ , se omite el cálculo de D2A para el par correspondiente a este bloque.

**Cálculo de estadísticas por tramo entre estaciones:** Con los instantes de llegada y salida a cada estación teórica, se calcula el tiempo A2A y D2A entre estaciones consecutivas para cada viaje.

- **A2A:** Para estaciones consecutivas `prox_est_teoricaj` y `prox_est_teoricaj+1`, el tiempo A2A se calcula como:

$$A2A = t_{arr}(\text{prox\_est\_teorica}_{j+1}) - t_{arr}(\text{prox\_est\_teorica}_j)$$

- **D2A:** Para estaciones consecutivas  $\text{prox\_est\_teorica}_j$  y  $\text{prox\_est\_teorica}_{j+1}$ , el tiempo D2A se calcula como:

$$\text{D2A} = t_{arr}(\text{prox\_est\_teorica}_{j+1}) - t_{dep}(\text{prox\_est\_teorica}_j)$$

Al cálculo de cada par consecutivo se le asocian las etiquetas `LINEA`, `estacion_actual`, `siguiente_estacion`, donde `estacion_actual` es la estación de partida  $j$  y `siguiente_estacion` es la estación de llegada  $j + 1$ , obteniendo así un conjunto de tiempos A2A y D2A por cada tramo entre estaciones.

**Agregación de tiempos por tramo:** El conjunto de todas las observaciones de tiempos A2A y D2A se agrupa por las etiquetas `LINEA`, `estacion_actual`, `siguiente_estacion`, es decir, por tramo entre estaciones de una misma línea. Para cada tramo, se calcularon tiempos típicos de recorrido a partir de todas las observaciones disponibles.

Con el fin de evitar que valores extremos (viajes anormalmente lentos o rápidos) afectaran los resultados, se aplicó un filtrado robusto basado en el rango intercuartílico (IQR): se eliminaron las observaciones fuera del intervalo  $[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$ , donde  $Q1$  y  $Q3$  son el primer y tercer cuartil, respectivamente.

Posteriormente, se calcularon las siguientes estadísticas para los tiempos A2A y D2A de cada tramo:

- Percentil 10
- Percentil 25
- Percentil 50 (mediana)
- Percentil 75
- Percentil 90
- Media recortada al 10 %, es decir, la media calculada excluyendo el 10 % de los valores más bajos y el 10 % de los valores más altos.
- Desviación estándar

Estas estadísticas fueron almacenadas en una tabla de referencia que puede ser consultada para complementar las predicciones de los modelos de machine learning, especialmente en escenarios donde se requiere estimar tiempos de llegada a estaciones más allá de la siguiente estación teórica.

## 6.10. Backend para interacción con los modelos de predicción

Se desarrolló un *backend* utilizando FastAPI, que expone una API REST para interactuar con los modelos de predicción. Esta permite recibir los datos de entrada que se necesitan del

usuario, utilizar los modelos para realizar predicciones basadas en los datos más recientes de cada unidad de Transmetro que está en operación, complementar dichas predicciones con estadísticas históricas cuando sea necesario y devolver los resultados al cliente. Cabe mencionar que la comunicación entre el *frontend* y el *backend* se realiza a través de solicitudes HTTP, utilizando JSON como formato de intercambio de datos.

#### 6.10.1. Archivos de información estática

Se crearon dos archivos JSON que contienen la información estática necesaria para el funcionamiento de la aplicación web:

- **Información de estaciones:** Un archivo que mapea cada estación del sistema de Transmetro con un identificador único, su nombre, la línea a la que pertenece, la dirección en la que se encuentra (ida o vuelta) y sus coordenadas geográficas (latitud y longitud). Esta información es consistente con la utilizada durante el preprocesamiento de datos y el entrenamiento de los modelos.
- **Orden canónico de estaciones:** Un archivo que contiene la secuencia ordenada de estaciones por línea y dirección. Este orden es utilizado para determinar las estaciones siguientes según la próxima estación teórica actual de una unidad.

#### 6.10.2. Uso de modelos entrenados

Se implementaron las siguientes funciones para cargar los modelos de predicción previamente entrenados y utilizarlos para realizar predicciones basadas en los datos de entrada proporcionados por el usuario y los datos más recientes de cada unidad en operación:

**load\_model():** Esta función carga en memoria los modelos entrenados a partir del directorio especificado. Soporta tanto un solo modelo final como un *ensamble* (p. ej., LightGBM y XGBoost).

**df\_from\_latest\_dict(data: dict):** Esta función recibe una solicitud de predicción en un diccionario de Python y lo transforma en un *dataframe* de pandas, que es el formato esperado por los modelos de predicción. Verifica la presencia de todas las columnas necesarias para la predicción y convierte los tipos de datos según lo requerido por los modelos entrenados:

- Columnas categóricas: Convertidas a tipo `category`.
- Columnas de números enteros: Convertidas a tipo `float32`.
- Columnas booleanas: Convertidas a tipo `bool`.

**predict\_boosters(boosters: list, data: dict):** Esta función recibe la lista de los modelos cargados en memoria y el diccionario de datos de entrada para realizar las predicciones. Primero, utiliza la función `df_from_latest_dict(data)` para transformar el diccionario en un *dataframe* de pandas. Luego, itera sobre cada modelo en la lista `boosters` y utiliza el método `predict()` de cada modelo para obtener las predicciones. Si hubiera más de un modelo, se calcula el promedio de las predicciones obtenidas por cada uno. Finalmente, Además, en caso de obtener predicciones negativas, estas se ajustan a cero para mantener la interpretabilidad del tiempo estimado de llegada.

#### 6.10.3. Obtención de datos más recientes:

Para obtener los datos más recientes de cada unidad de Transmetro en operación y transformarlos a un formato manejable en el *backend*, se implementaron las siguientes funciones auxiliares:

**load\_latest\_data\_per\_unit()** : Esta función auxiliar carga el archivo .parquet generado por el preprocesamiento de datos descrito en secciones anteriores. Se asume que este archivo es actualizado de manera periódica con los datos más recientes de las unidades en operación. Para fines de demostración, se utilizó un archivo con los datos preprocesados correspondientes al 13 de mayo de 2024, día en el que se registraron unidades en operación para todas las líneas del sistema de Transmetro.

Una vez cargados los datos, la función agrupa los registros por unidad y selecciona el registro más reciente para cada una. Si el último registro de alguna unidad tiene una edad mayor una hora, se descarta dicha unidad considerando que no está en operación.

La función devuelve un *dataframe* de pandas con los registros más recientes de cada unidad en operación.

**get\_latest\_data\_by\_line\_and\_dir(line, \_dir)** : Esta función auxiliar recibe como entrada la línea y dirección (ida, vuelta o circular) de interés e invoca a la función `load_latest_data_per_unit()` para obtener los datos más recientes de todas las unidades en operación. Luego, filtra los registros para conservar únicamente aquellos que pertenecen a la línea y dirección especificadas. Finalmente, devuelve un diccionario de Python donde las claves son la línea, dirección y próxima estación teórica de cada unidad y los valores son los registros correspondientes en formato de diccionario.

#### 6.10.4. Definición de secuencia de estaciones siguientes por unidad

Tomando en cuenta que puede solicitarse el tiempo estimado de llegada a una estación específica que no sea la próxima estación teórica más reciente de ninguna de las unidades que están operando en la línea a la que pertenece dicha estación, es necesario definir la secuencia ordenada de estaciones que dichas unidades deben seguir para llegar a la estación solicitada. Para ello, se definieron las siguientes funciones auxiliares:

**get\_station\_sequence(line, dir\_):** Esta función utiliza el archivo estático del orden canónico de estaciones mencionado anteriormente para obtener la secuencia ordenada de estaciones correspondiente a la línea y dirección especificadas.

**find\_stations\_between(origin\_stations, origin\_stations\_dir, target\_station, target\_line, target\_dir):** Esta función determina la secuencia de estaciones entre diferentes estaciones de partida y una estación objetivo. Recibe como entrada, en primer lugar, la línea a la que pertenecen tanto las estaciones de partida de la secuencia como la estación objetivo. También recibe una la dirección en la que se encuentran las estaciones de partida (por simplicidad, todas las estaciones de partida que reciba la función deben estar en la misma dirección). Por último, recibe la estación objetivo de la secuencia y la dirección en la que esta se encuentra.

Se obtiene la secuencia ordenada de estaciones para la línea y dirección de las estaciones de partida invocando a la función `get_station_sequence(line, dir_)`. Si la estación objetivo no se encuentra en la misma dirección que las estaciones de partida, también se obtiene la secuencia ordenada de estaciones para la dirección opuesta.

Finalmente, para cada estación de partida, la secuencia de estaciones que debe seguir para llegar a la estación objetivo se define según la cantidad de direcciones involucradas en la línea y el orden relativo entre la estación de partida y la objetivo.

Para líneas bidireccionales:

- **Estación objetivo en la misma dirección que la de partida:** Si la estación objetivo se encuentra después de la de partida, únicamente se consideran las estaciones siguientes a la de partida en esta dirección hasta llegar a la objetivo. Si la estación objetivo se encuentra antes que la de partida, el recorrido incluye las estaciones restantes de la dirección de la de partida, todas las de la dirección opuesta, y todas las de la misma dirección previas a la objetivo.
- **Estación objetivo en la dirección opuesta que la de partida:** El recorrido incluye las estaciones restantes de la dirección de la de partida y todas las de la dirección opuesta previas a la objetivo.

Para líneas circulares:

- **Estación objetivo después de la de partida:** Únicamente se consideran las estaciones siguientes a la de partida hasta llegar a la objetivo.
- **Estación objetivo antes de la de partida:** El recorrido incluye las estaciones restantes después de la de partida y todas las estaciones previas a la objetivo.

El resultado de esta función es una lista de listas, donde cada sublista contiene la secuencia de estaciones que se deben seguir desde cada estación de partida hasta la estación objetivo.

### 6.10.5. Cálculo de tiempos de llegada acumulados para una secuencia de estaciones

Para calcular los tiempos de llegada acumulados para una secuencia de estaciones, se implementó la siguiente función auxiliar:

**cumulative\_ETA(stations\_sequence, line):** Esta función recibe como entrada una lista de estaciones que conforman la secuencia a seguir por una unidad y la línea a la que pertenecen dichas estaciones. Utiliza el archivo estático de tiempos históricos por tramo entre estaciones para obtener los tiempos típicos de recorrido entre cada par de estaciones consecutivas en la secuencia. Devuelve la suma total de estos tiempos, representando el tiempo total estimado de llegada desde la primera estación de la secuencia hasta la última. Cabe mencionar que, debido a que durante el preprocesamiento de datos un cambio de dirección en una misma estación se indicó incluyendo dicha estación en ambas direcciones, es posible que la secuencia de estaciones contenga pares de estaciones consecutivas idénticas. En estos casos, la función simplemente omite el par al calcular el tiempo total estimado de llegada.

### 6.10.6. Cálculo de ETA por estación y selección del mejor candidato

Para determinar el tiempo estimado de llegada a una estación solicitada, se implementó la siguiente función auxiliar:

**get\_closest\_ETA\_for\_station(line, dir\_, station):** Esta función recibe como entrada la estación objetivo, la línea a la que pertenece y la dirección en la que se encuentra. Primero, utiliza la función `get_latest_data_by_line_and_dir(line, dir_)` para obtener los datos más recientes de las unidades operando en la línea y dirección especificadas. Si la línea es bidireccional, se repite el proceso para la dirección opuesta y se combinan los resultados. Una vez obtenidos los datos más recientes de las unidades candidatas, para cada una de ellas:

- Se identifica su próxima estación teórica más reciente.
- Se determina la secuencia de estaciones que debe seguir para llegar a la estación objetivo utilizando la función `find_stations_between(...)`.
- Se calcula el tiempo estimado de llegada a su próxima estación teórica utilizando los modelos de predicción cargados en memoria a través de la función `predict_boosters(...)`.
- Se calcula el tiempo estimado de llegada acumulado desde la próxima estación teórica hasta la estación objetivo utilizando la función `cumulative_ETA(...)`.
- Se suman ambos tiempos para obtener el tiempo estimado de llegada total desde la posición actual de la unidad hasta la estación objetivo.

- Se almacena el tiempo estimado de llegada total para cada unidad candidata.

Finalmente, se selecciona la unidad con el tiempo estimado de llegada más bajo.

El resultado de la función incluye tanto el ETA estimado como la información de la unidad más cercana:

- **`prediction`**: tiempo estimado de llegada total (en segundos).
- **`closest_unit`**: diccionario con los datos de la unidad más cercana (placa, línea, dirección, próxima estación, latitud, longitud y estaciones intermedias).

#### 6.10.7. Cálculo del tiempo estimado de duración de un viaje entre estaciones

Para calcular el tiempo estimado de duración de un viaje entre dos estaciones específicas, se implementó la siguiente función auxiliar:

**`get_trip_duration_between_stations(line, origin_station, origin_direction, dest_station, dest_direction)`**: Esta función recibe como entrada la línea a la que pertenecen las estaciones de origen y destino, los nombres de ambas estaciones y sus respectivas direcciones (ida, vuelta o circular). El proceso que sigue consiste en:

1. Estimar el ETA actual hacia la estación de origen (según la unidad más cercana) utilizando `get_closest_eta_for_station()`.
2. Identificar las estaciones intermedias entre la estación de origen y la estación destino mediante `find_stations_between()`.
3. Sumar los ETAs históricos (`cumulative_eta()`) de los tramos entre las estaciones consecutivas.

El resultado de la función incluye el tiempo estimado total de duración del viaje entre las estaciones de origen y destino, así como la información de la unidad más cercana a la estación de origen, similar al de la función `get_closest_eta_for_station()`.

#### 6.10.8. Inicialización de la aplicación FastAPI y exposición de endpoints

Al iniciar la aplicación FastAPI, se cargan los modelos de predicción utilizando la función `load_model()`. Además, se exponen los siguientes *endpoints* para interactuar con la API REST:

**GET /lines**: Este *endpoint* devuelve la lista de líneas disponibles, según las estaciones definidas en el archivo estático de información de estaciones.

**GET /stations:** Este *endpoint* devuelve la lista completa de estaciones disponibles, indistintamente de la línea a la que pertenecen, según el archivo estático de información de estaciones. Tal como se muestra en la figura 17, cada estación incluye su identificador único, nombre, línea, dirección y coordenadas geográficas.

```
{
    "id": 1,
    "name": "CENMA",
    "line": "Linea_12",
    "direction": "IDA",
    "lat": 14.6345,
    "lon": -90.5061
},
```

Figura 17: Ejemplo de respuesta del *endpoint* GET /stations.

**GET /stations/line:** Este *endpoint* recibe como parámetro la línea de interés y su respuesta es similar a la del *endpoint* anterior, pero filtrada para conservar únicamente las estaciones que pertenecen a la línea especificada.

**GET /stations/station\_id:** Este *endpoint* recibe como parámetro el identificador único de una estación y su respuesta también es similar a la del *endpoint* GET /stations, pero filtrada para conservar únicamente la estación especificada.

**POST /eta:** Este *endpoint* recibe en el cuerpo de la solicitud un JSON con los siguientes campos:

- **line:** Línea a la que pertenece la estación objetivo.
- **direction:** Dirección en la que se encuentra la estación objetivo (ida, vuelta o circular).
- **station:** Nombre de la estación objetivo.

Utiliza la función `get_closest_ETA_for_station(...)` para calcular el tiempo estimado de llegada a la estación objetivo. El cuerpo de la respuesta se encuentra en formato JSON e incluye este valor en segundos, la información más reciente de la mejor unidad candidata seleccionada, la versión del modelo (o *ensamble*) utilizado, y la cantidad de modelos utilizados para realizar la predicción, tal como se muestra en la figura 18.

```
{
  "closest_unit": {
    "unit": "401",
    "line": "Linea_7",
    "direction": "VUELTA",
    "next_station": "CIUDAD DE PLATA DIRECCIÓN USAC",
    "cumulative_ETA": 59.98846314140643,
    "latitude": 14.639015,
    "longitude": -90.55039,
    "stations_between": [
      "CIUDAD DE PLATA DIRECCIÓN USAC"
    ]
  },
  "prediction": 59.98846314140643,
  "model_version": "2025-10-25",
  "n_models": 1
}
```

Figura 18: Ejemplo de respuesta del endpoint POST /eta.

**POST /trip:** Este endpoint recibe en el cuerpo de la solicitud un JSON con los siguientes campos:

- **line:** Línea a la que pertenecen las estaciones de origen y destino.
- **origin\_station:** Nombre de la estación de origen.
- **origin\_direction:** Dirección en la que se encuentra la estación de origen (ida, vuelta o circular).
- **dest\_station:** Nombre de la estación de destino.
- **dest\_direction:** Dirección en la que se encuentra la estación de destino (ida, vuelta o circular).

Utiliza la función `get_trip_duration_between_stations(...)` para calcular el tiempo estimado de duración del viaje entre las estaciones de origen y destino. El cuerpo de la respuesta se encuentra en formato JSON e incluye este valor en segundos, la información más reciente de la mejor unidad candidata seleccionada para la estación de origen, la versión del modelo (o *ensamble*) utilizado, y la cantidad de modelos utilizados para realizar la predicción, similar a la respuesta del endpoint POST /eta.

La figura 19 muestra un diagrama simplificado del flujo de datos y las interacciones entre los diferentes componentes del *backend*.

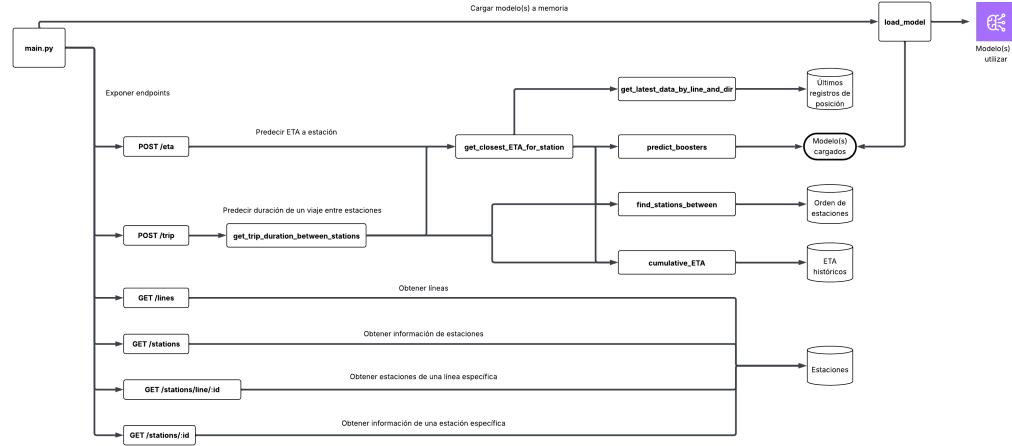


Figura 19: Diagrama simplificado del *backend* y sus interacciones.

## 6.11. Frontend para interacción con los modelos de predicción

Se desarrolló un *frontend* web utilizando ReactJS que permite a posibles usuarios interactuar con la API REST del *backend* para obtener tiempos estimados de llegada a estaciones específicas y duraciones de viajes entre estaciones.

### 6.11.1. Prototipado de vistas

Se diseñaron prototipos de las vistas de la aplicación web utilizando Figma. Estos prototipos sirvieron como guía para el desarrollo del *frontend*, procurando una experiencia de usuario intuitiva y coherente (véase el Anexo 12.2 para más detalles).

### 6.11.2. Implementación del frontend

Siguiendo los prototipos diseñados y tomando en cuenta las entradas del usuario requeridas por la API REST del *backend*, se implementó una estructura basada en rutas utilizando React Router. Se definieron las siguientes rutas:

- **/decision:** Vista que permite al usuario seleccionar si desea obtener el tiempo estimado de llegada a una estación específica o la duración estimada de un viaje entre dos estaciones.
- **/stations:** Vista que, al renderizarse, obtiene la información de todas las estaciones disponibles a través del endpoint GET `/stations` del *backend* y las muestra en una lista. Permite al usuario seleccionar cualquiera de estas estaciones para obtener el tiempo estimado de llegada a la misma. Al seleccionar una estación, el usuario es redirigido a la vista `/eta:station_id`, en donde `station_id` es el identificador único de la estación seleccionada.

- **/eta:stationId**: Vista que recibe como parámetro de ruta el identificador único de una estación seleccionada por el usuario. Al renderizarse, obtiene la información de dicha estación a través del *endpoint GET /stations/{station\_id}* del *backend* y muestra la información de esta al usuario. Además, una vez obtenida la información de la estación, envía una solicitud al *endpoint POST /eta* del *backend* para obtener el tiempo estimado de llegada a dicha estación. Finalmente, muestra el resultado de la predicción al usuario, junto con la placa de la unidad más cercana seleccionada y su posición en un mapa.
- **/trip**: Vista que inicialmente renderiza un formulario que permite al usuario seleccionar la línea en la que desea realizar el viaje, el cual muestra las líneas disponibles obtenidas a través del *endpoint GET /lines* del *backend*. Una vez seleccionada la línea, el formulario permite al usuario seleccionar la estación de origen y la estación de destino del viaje, mostrando las estaciones disponibles para la línea seleccionada a través del *endpoint GET /stations/{line}* del *backend*. Al enviar el formulario, se redirecciona al usuario a la vista */trip/eta/:originStationId/:destinationStationId*, en donde *originStationId* y *destinationStationId* son los identificadores únicos de las estaciones de origen y destino seleccionadas, respectivamente.
- **/trip/eta/:originStationId/:destinationStationId**: Vista que recibe como parámetros de ruta los identificadores únicos de las estaciones de origen y destino seleccionadas por el usuario. Al renderizarse, obtiene la información de ambas estaciones a través del *endpoint GET /stations/{station\_id}* del *backend* y muestra la información de estas al usuario. Además, una vez obtenida la información de ambas estaciones, envía una solicitud al *endpoint POST /trip* del *backend* para obtener la duración estimada del viaje entre las dos estaciones. Finalmente, muestra el resultado de la predicción al usuario, junto con la placa de la unidad más cercana seleccionada para la estación de origen y su posición en un mapa.

# CAPÍTULO 7

## Resultados

### 7.1. Resultados de la recolección y limpieza de datos

La recolección automatizada de datos permitió obtener los registros de posición de 159 unidades de Transmetro entre el 1 de enero de 2024 y el 30 de abril de 2025. El cuadro 6 resume la cantidad de registros antes y después del proceso de limpieza de datos basado en la identificación de trayectorias útiles.

Cuadro 6: Resumen del conjunto de datos antes y después de la limpieza.

Indicador	Antes	Después	Reducción (%)
Registros	63,257,434	51,261,106	11,996,328 (18.96 %)
Unidades	159	156	3 (1.89 %)

La figura 20, muestra los puntos GPS de los registros pertenecientes a las trayectorias no útiles de las unidades 049, 401 y 216 graficados en un mapa de Guatemala.

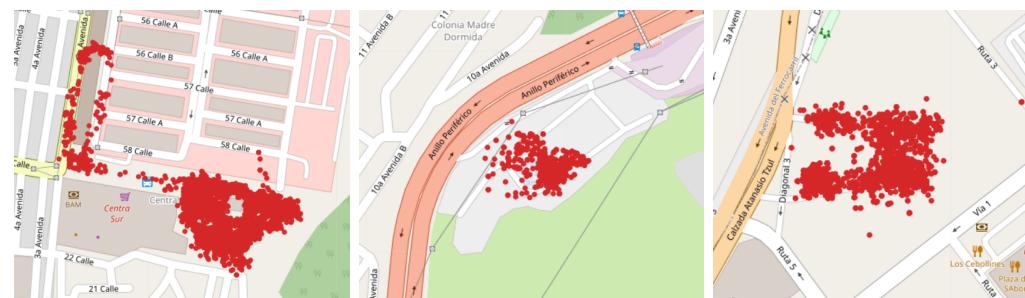


Figura 20: Mapa de las trayectorias no útiles para las unidades 049, 401 y 216, respectivamente.

Por otro lado, la figura 21 muestra los puntos GPS de los registros pertenecientes a una de las trayectorias útiles de la unidad 049 graficados en un mapa de Guatemala, en

comparación con el mapa de la línea 12 de Transmetro disponible en la página oficial de la Municipalidad de Guatemala.

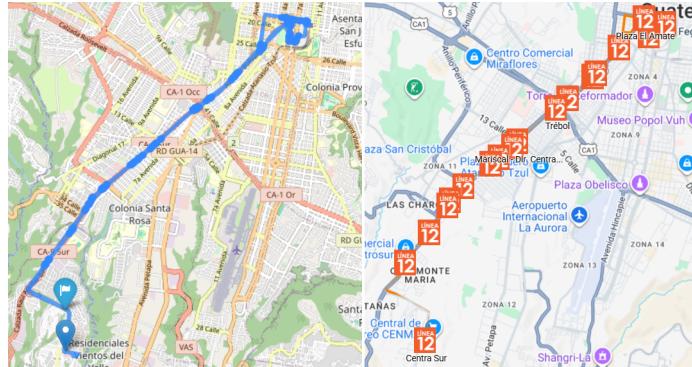


Figura 21: Comparación de una de las trayectorias útiles de la unidad 049 con el mapa de la línea 12 de Transmetro en la página oficial de la Municipalidad de Guatemala [21].

## 7.2. Resultados de la ingeniería de características

El cuadro 7 resume las características generadas para cada registro en el conjunto de datos final.

La figura 22 el viaje 1 de la unidad 113, graficado en comparación con el mapa de la línea 18 de Transmetro disponible en la página oficial de la Municipalidad de Guatemala [21], mientras que el cuadro 8 muestra el resultado del algoritmo de Viterbi aplicado a dicho viaje.

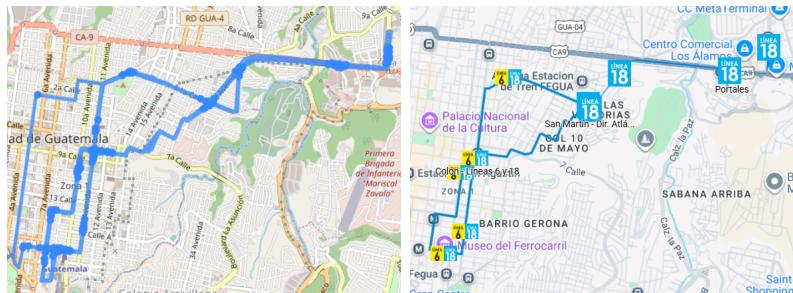


Figura 22: Comparación del viaje 1 de la unidad 113 con el mapa de la línea 18 de Transmetro.

Cuadro 8: Resultado del algoritmo de Viterbi para el viaje 1 de la unidad 113.

trip_id	LINEA	DIR_init	t_start
1	Línea_18-A	VUELTA	2024-01-02 04:14:08

El cuadro 9 muestra el resultado del algoritmo de Viterbi aplicado al viaje 146 de la unidad 204, graficado en la figura 8, en el que dicha unidad recorrió las líneas 1 y 6 de Transmetro.

Cuadro 7: Resumen de las características generadas.

Característica	Tipo	Descripción
LINEA	Categórica	Línea de Transmetro que está recorriendo la unidad
DIR	Categórica	Dirección de la línea que está recorriendo la unidad
proxima_est_teorica	Categórica	Nombre de la próxima estación teórica según la posición de la unidad sobre la polilínea
dist_a_prox_m	Numérica	Distancia restante de la abcisa curvilínea a la próxima estación teórica
dist_estacion_m	Numérica	Distancia ortodrómica a la estación más cercana
vel_mps	Numérica	Velocidad instantánea registrada por la unidad en metros por segundo
Altitud	Numérica	Altitud registrada por la unidad en metros sobre el nivel del mar
s_m	Numérica	Distancia recorrida en metros desde el inicio de la abcisa curvilínea de línea-dirección actual
dist_m	Numérica	Distancia euclíadiana desde el punto GPS del registro al punto más cercano de la polilínea de línea-dirección actual
time_diff_s	Numérica	Diferencia de tiempo en segundos entre el registro actual y el registro previo de la misma unidad
dwell_same_xy_s	Numérica	Tiempo en segundos que la unidad ha permanecido en la misma posición GPS
ETA_proxima_est_s	Numérica	Variable objetivo: Tiempo estimado en segundos para llegar a la próxima estación teórica
hour	Numérica	Hora del día en formato 1 a 24
day_of_week	Numérica	Día de la semana en formato 0 a 6
is_weekend	Binaria	Indicador binario de si el día es fin de semana (1) o día laborable (0)
is_no_progress	Binaria	Indicador binario de si la unidad no ha avanzado en su posición GPS (1) o sí ha avanzado (0)
is_peak	Binaria	Indicador binario de si el registro fue realizado entre 5 y 9 de la mañana o 4 y 7 de la tarde (1) o no (0)

Cuadro 9: Resultado del algoritmo de Viterbi para el viaje 146 de la unidad 204.

trip_id	LINEA	DIR_init	t_start
146	Linea_1	CIRCULAR	2024-03-01 05:30:24
146	Linea_6	IDA	2024-03-01 17:17:16

La figura 23 muestra un par de puntos GPS consecutivos del viaje 1 de la unidad 113, junto con las estaciones teóricas inferidas para cada punto.



Figura 23: Puntos GPS consecutivos del viaje 1 de la unidad 113 con las estaciones teóricas inferidas.

La figura 24 muestra la matriz de correlación entre las características numéricas generadas y la variable objetivo ETA\_proxima\_est\_s.

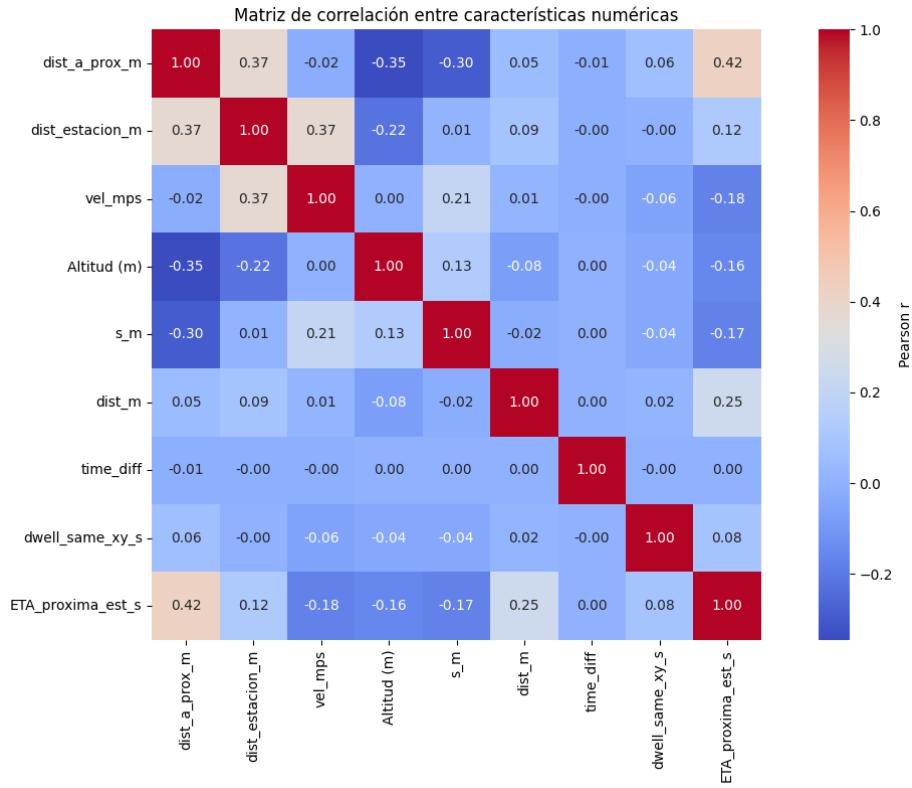


Figura 24: Matriz de correlación entre las características numéricas generadas y la variable objetivo ETA\_proxima\_est\_s.

El cuadro 10 resume la limpieza de datos realizada en base a la poca longitud de las trayectorias útiles y la presencia de valores atípicos en el tiempo de permanencia y la variable objetivo.

Cuadro 10: Resumen del conjunto de datos antes y después de la limpieza con las características generadas.

Indicador	Antes	Después	Reducción (%)
Registros	51,261,106	43,030,926	8,230,180 (16.06 %)
Unidades	156	156	0 (0.00 %)

La figura 25 muestra la distribución de la variable objetivo ETA\_proxima\_est\_s en el conjunto de datos final después de la limpieza.

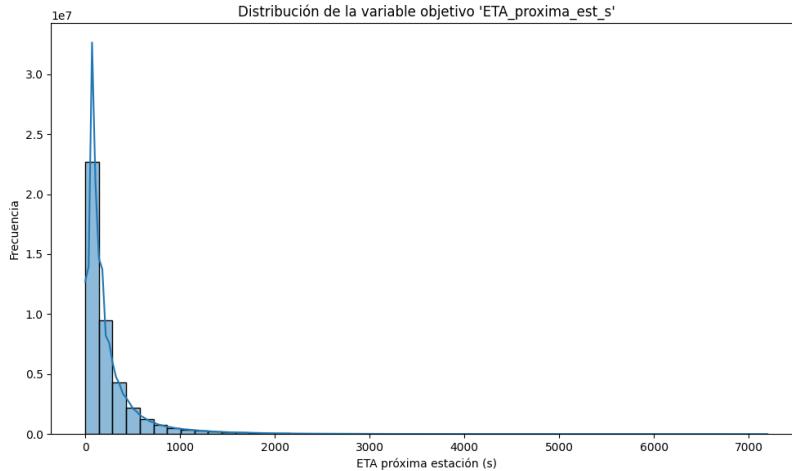


Figura 25: Distribución de la variable objetivo `ETA_proxima_est_s` en el conjunto de datos final.

### 7.3. Resultados del entrenamiento de modelos

Los resultados del entrenamiento de los modelos LightGBM y XGBoost consisten en el desempeño de cada modelo en términos de las métricas de error absoluto medio (MAE, por sus siglas en inglés), error cuadrático medio (RMSE, por sus siglas en inglés) y los umbrales de error absoluto, tanto en la validación cruzada como en el conjunto de prueba final. También se incluyen las curvas de pérdida durante el entrenamiento, el desempeño desglosado por línea y dirección, y la importancia de características según cada modelo.

#### 7.3.1. LightGBM

El cuadro 11 presenta el desempeño del modelo LightGBM en términos de las métricas MAE, RMSE, mientras que el cuadro 12 muestra el porcentaje de registros en donde el error absoluto es menor o igual a los umbrales de 60, 120 y 180 segundos, tanto en la validación cruzada como en el conjunto de prueba final.

Cuadro 11: Desempeño del modelo LightGBM.

Métrica	MAE (s)	RMSE (s)
Promedio validación cruzada	$110.14 \pm 15.64$	$302.04 \pm 50.82$
Conjunto de prueba final	122.41	344.11

Cuadro 12: Umbrales de error absoluto del modelo LightGBM.

Umbraal	$\epsilon \leq 60\text{s}$	$\epsilon \leq 120\text{s}$	$\epsilon \leq 180\text{s}$
Promedio validación cruzada	$63.45\% \pm 2.87$	$79.51\% \pm 2.50$	$86.41\% \pm 2.05$
Conjunto de prueba final	60.63 %	77.09 %	84.66 %

La mediana de las mejores iteraciones por *fold* durante la validación cruzada fue de 896. La figura 26 muestra la curva de pérdida del modelo LightGBM durante el entrenamiento

del modelo final al utilizar este número de iteraciones como referencia.

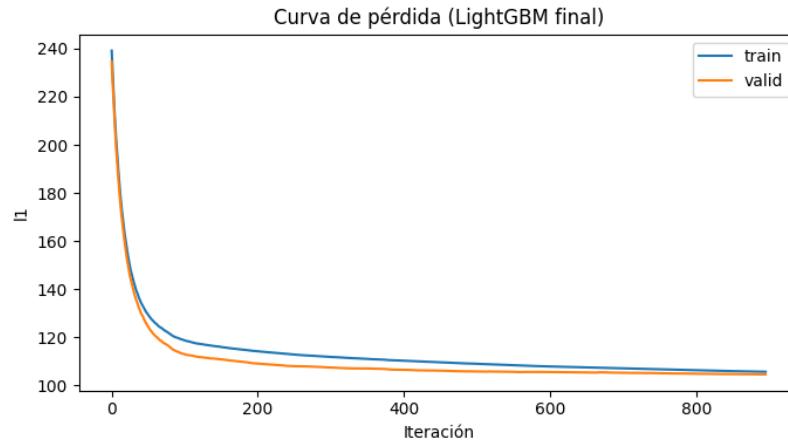


Figura 26: Curva de pérdida del modelo LightGBM durante el entrenamiento del modelo final.

El cuadro 13 muestra la función de pérdida del modelo final LightGBM en la última iteración del entrenamiento.

Cuadro 13: Desempeño del modelo LightGBM en la última iteración del entrenamiento.

Métrica	MAE (s)	RMSE (s)
Entrenamiento	106.094	276.99
Validación	109.591	302.619

La figura 27 detalla el desempeño del modelo LightGBM desglosado por línea y dirección, mostrando el MAE para cada combinación, mientras que la figura 28 muestra el RMSE para cada combinación.

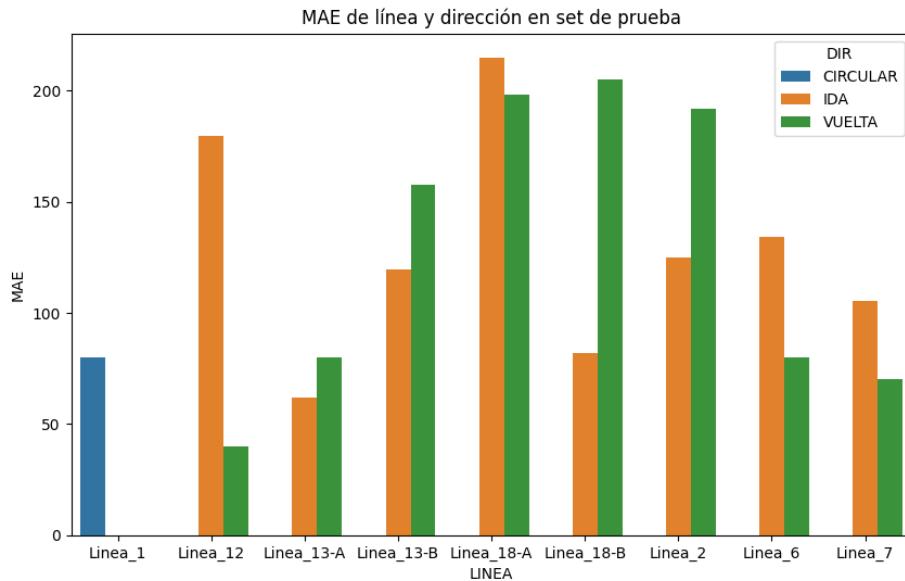


Figura 27: Desempeño del modelo LightGBM por línea y dirección (MAE).

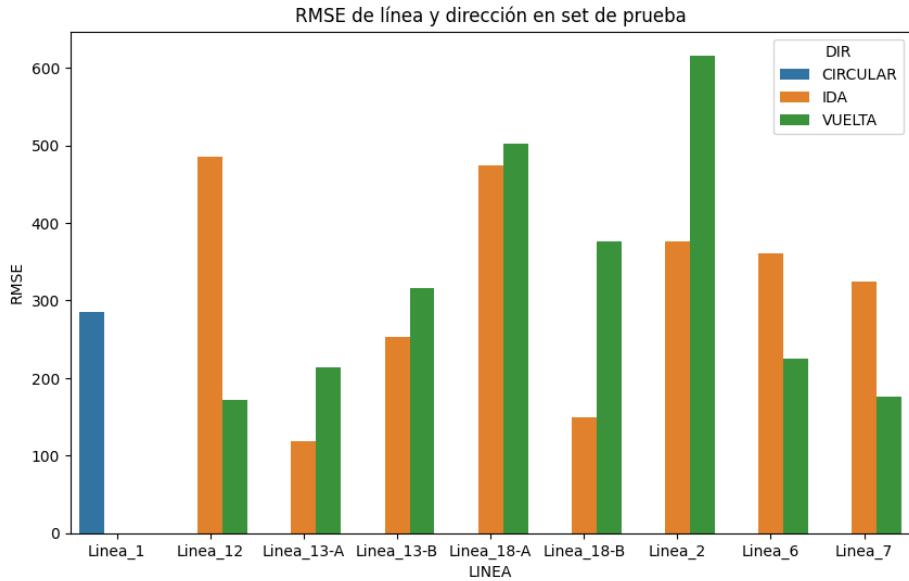


Figura 28: Desempeño del modelo LightGBM por línea y dirección (RMSE).

La figura 29 muestra, en orden descendente, las características más importantes según la ganancia promedio en el modelo LightGBM final.

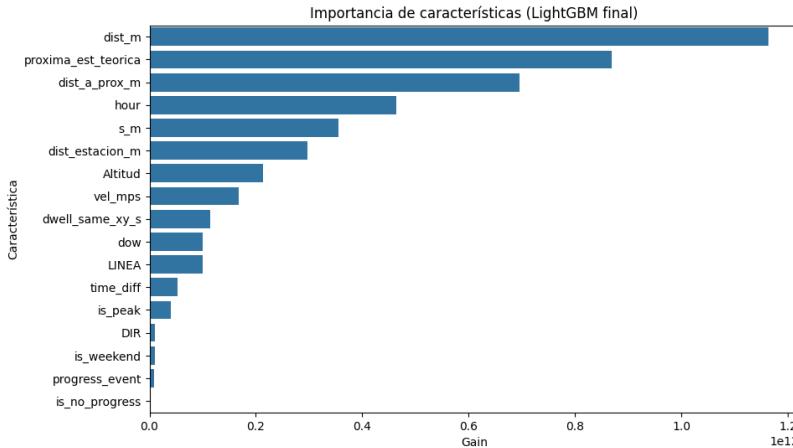


Figura 29: Importancia de características en el modelo LightGBM.

### 7.3.2. XGBoost

El cuadro 14 presenta el desempeño del modelo XGBoost en términos de las métricas MAE, RMSE, mientras que el cuadro 15 muestra el porcentaje de registros en donde el error absoluto es menor o igual a los umbrales de 60, 120 y 180 segundos, tanto en la validación cruzada como en el conjunto de prueba final.

Cuadro 14: Desempeño del modelo XGBoost.

Métrica	MAE (s)	RMSE (s)
Promedio validación cruzada	$121.82 \pm 13.03$	$323.22 \pm 41.19$
Conjunto de prueba final	125.87	352.02

Cuadro 15: Umbrales de error absoluto del modelo XGBoost.

Umbra	$\epsilon \leq 60\text{s}$	$\epsilon \leq 120\text{s}$	$\epsilon \leq 180\text{s}$
Promedio validación cruzada	$60.35\% \pm 2.41$	$77.17\% \pm 2.21$	$84.60\% \pm 1.77$
Conjunto de prueba final	59.78 %	76.28 %	84.09 %

La mediana de las mejores iteraciones por *fold* durante la validación cruzada fue de 1604. La figura 30 muestra la curva de pérdida del modelo XGBoost durante el entrenamiento del modelo final.

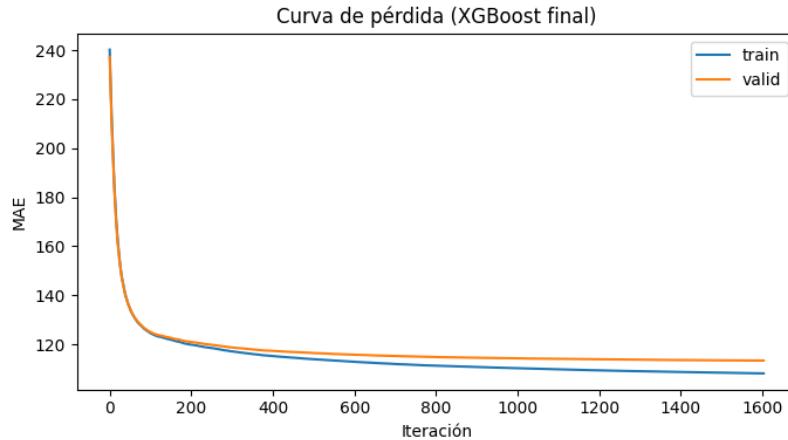


Figura 30: Curva de pérdida del modelo XGBoost durante el entrenamiento del modelo final.

El cuadro 16 muestra la función de pérdida del modelo final XGBoost en la última iteración del entrenamiento.

Cuadro 16: Desempeño del modelo XGBoost en la última iteración del entrenamiento.

Métrica	MAE (s)	RMSE (s)
Entrenamiento	108.13	281.48
Validación	113.37	310.87

La figura 31 detalla el desempeño del modelo XGBoost desglosado por línea y dirección, mostrando el MAE para cada combinación, mientras que la figura 32 muestra el RMSE para cada combinación.

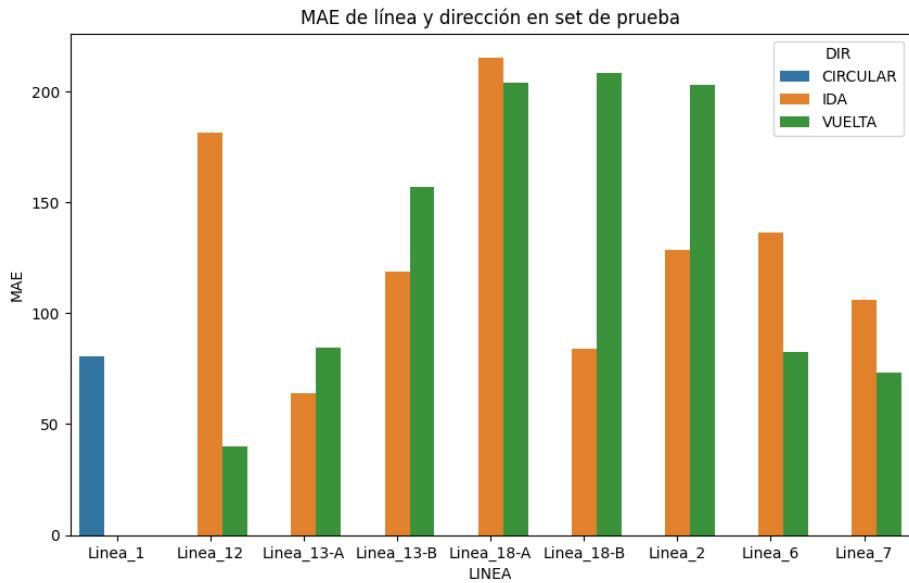


Figura 31: Desempeño del modelo XGBoost por línea y dirección (MAE).

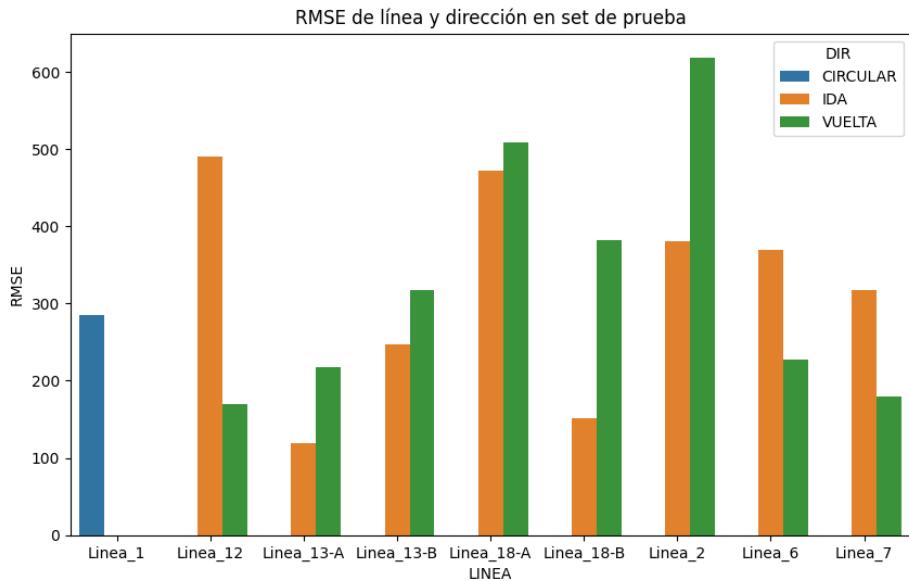


Figura 32: Desempeño del modelo XGBoost por línea y dirección (RMSE).

La figura 33 muestra las diez características más importantes según la ganancia promedio en el modelo XGBoost entrenado.

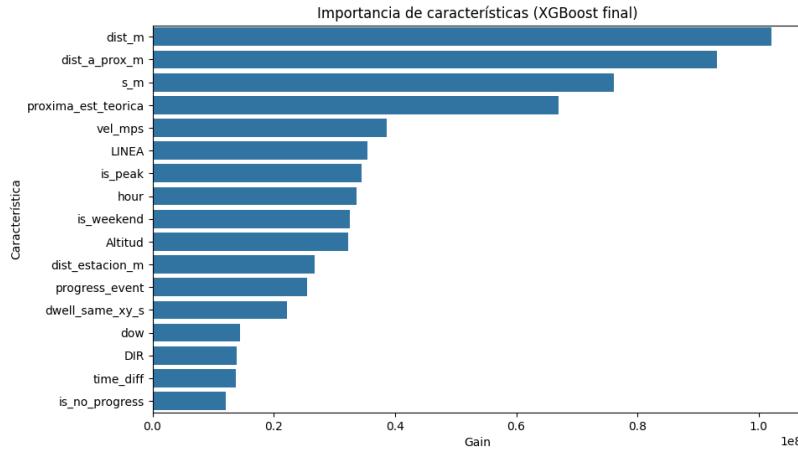


Figura 33: Importancia de características en el modelo XGBoost.

### 7.3.3. Resultados del cálculo de tiempos de viaje históricos entre estaciones

El cuadro 17 ejemplifica el formato de las estadísticas calculadas de los tiempos de viaje A2A históricos entre estaciones de la misma línea.

Cuadro 17: Formato de las estadísticas A2A históricas.

LINEA	estacion_actual	siguiente_estacion	p10	p25	p50	p75	p90
Línea_12	CENMA	MONTE MARÍA	407.0	606.0	882.0	1245.0	1710.0
Línea_6	SANTA TERESA	CAPUCHINAS	120.0	156.0	180.0	240.0	252.0

El cuadro 18 ejemplifica el formato de las estadísticas calculadas de los tiempos de viaje D2A históricos entre estaciones de la misma línea.

Cuadro 18: Formato de las estadísticas D2A históricas.

LINEA	estacion_actual	destino	p10	p25	p50	p75	p90
Línea_12	CENMA	MONTE MARÍA	159.6	300.0	798.0	1020.0	1200.4
Línea_6	SANTA TERESA	CAPUCHINAS	60.0	60.0	120.0	178.0	223.0

## 7.4. Resultados de la implementación de los modelos

La figura 34 muestra la obtención de los candidatos para la estación “MONTÚFAR”, de la línea 13, dirección “VUELTA” en el *backend* implementado, para el 13 de mayo de 2024 a las 14:38:00.

```

    === Unidades disponibles para Línea_13-A dirección VUELTA ===
    Unidad: 154 Último registro: 2024-05-13 14:37:46 Próxima estación: 4 GRADOS SUR
    Unidad: 221 Último registro: 2024-05-13 14:37:18 Próxima estación: ACUEDUCTO
    Unidad: 161 Último registro: 2024-05-13 14:37:54 Próxima estación: EL CALVARIO
    Unidad: 157 Último registro: 2024-05-13 14:37:59 Próxima estación: EXPOSICIÓN
    Unidad: 215 Último registro: 2024-05-13 14:37:40 Próxima estación: HANGARES
    Unidad: 205 Último registro: 2024-05-13 14:37:41 Próxima estación: INDUSTRIA
    Unidad: 206 Último registro: 2024-05-13 14:37:58 Próxima estación: PLAZA ARGENTINA
    Unidad: 153 Último registro: 2024-05-13 14:37:52 Próxima estación: TERMINAL

    === Unidades disponibles para Línea_13-A dirección opuesta ===
    Unidad: 242 Último registro: 2024-05-13 14:37:48 Próxima estación: CANTÓN EXPOSICIÓN
    Unidad: 158 Último registro: 2024-05-13 14:37:58 Próxima estación: IGSS ZONA 9
    Unidad: 240 Último registro: 2024-05-13 14:37:56 Próxima estación: LOS ARCOS
    Unidad: 201 Último registro: 2024-05-13 14:37:54 Próxima estación: SEIS 26
    Unidad: 152 Último registro: 2024-05-13 14:37:55 Próxima estación: TIPOGRAFÍA

```

Figura 34: Obtención de candidatos para la estación “MONTÚFAR”, línea 13, dirección “VUELTA” en el *backend* implementado.

La figura 35 muestra la predicción del tiempo estimado de llegada a la próxima estación teórica para uno de los candidatos obtenidos en la figura 34 al utilizar el modelo LightGBM entrenado, así como la consulta de los tiempos históricos de viaje A2A de los tramos entre estaciones hasta la estación “MONTÚFAR”.

```

--- Unidad: 157
Datos recibidos para predicción a estación más cercana: {'Placa': '157', 'Fecha': Timestamp('2024-05-13 14:37:59'), 'LINEA': 'Línea_13-A', 'DIR': 'VUELTA', 'proxima_est_teorica': 'EXPOSICIÓN', 'dist_a_prox_m': 75.81926727294922, 's': 9.166666984558105, 'Altitud (m)': 1511.0, 's_m': 1344.40869140625, 'dist_m': 13.307112693786621, 'time_diff': 0, 'progress_event': 1, 'hour': 14, 'dow': 6, 'is_weekend': 1, 'is_peak': 0}
Predicción a próxima estación teórica (EXPOSICIÓN): [53.20403727]

ETA histórico desde EXPOSICIÓN hasta TERMINAL : 180.0
ETA histórico desde TERMINAL hasta INDUSTRIA : 120.0
ETA histórico desde INDUSTRIA hasta TIVOLI : 120.0
ETA histórico desde TIVOLI hasta MONTÚFAR : 180.0
Para la unidad 157 el ETA acumulado es: 653.2040372691856

```

Figura 35: Predicción del tiempo estimado de llegada a la próxima estación teórica y consulta de tiempos históricos A2A en el *backend* implementado.

La figura 36 muestra la respuesta final del *backend* implementado para la consulta del tiempo estimado de llegada a la estación “MONTÚFAR” de la línea 13, dirección “VUELTA”. En esta, se observa la información del candidato seleccionado para la estación consultada, incluyendo el tiempo estimado de llegada calculado por el modelo LightGBM y las estaciones intermedias que debe recorrer la unidad hasta llegar a dicha estación. También se muestra el despliegue de esta información en el *frontend* implementado.

Por otro lado, la figura 37 muestra la respuesta del *backend* implementado para la consulta del tiempo estimado de llegada a la estación “MONTÚFAR” de la línea 13, dirección “VUELTA”, para el 13 de mayo de 2024 a las 23:00:00, en donde no se encontraron candidatos disponibles para dicha estación en ese momento.

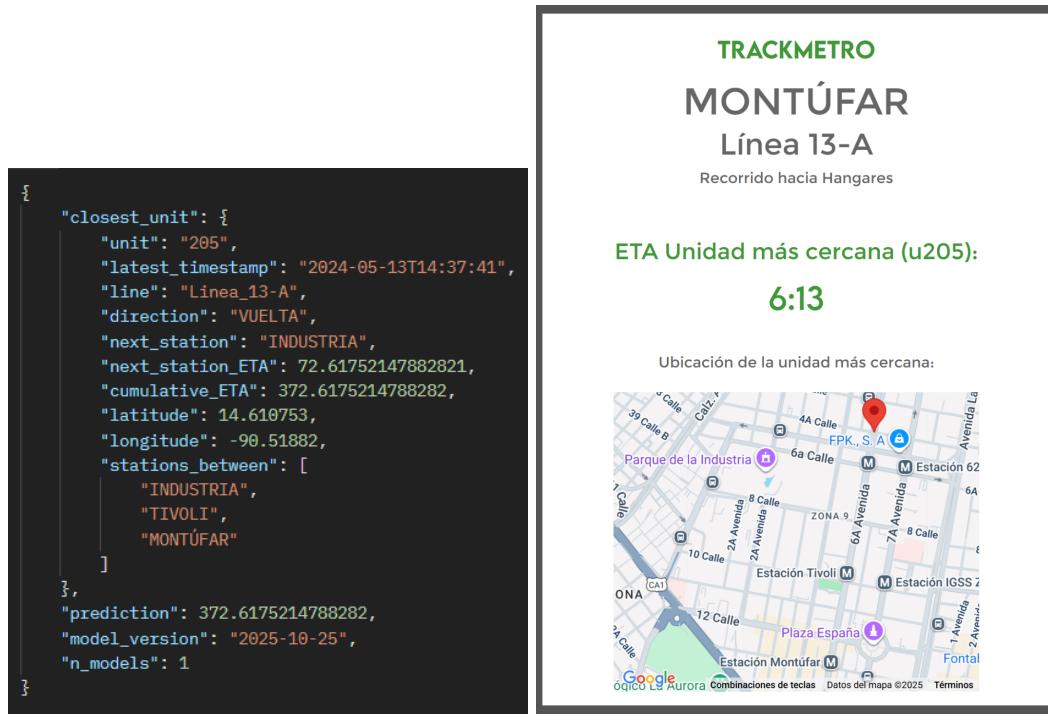


Figura 36: Respuesta final del *backend* implementado para la consulta del tiempo estimado de llegada a la estación “MONTÚFAR”, línea 13, dirección “VUELTA” el 13 de mayo de 2024 a las 14:38:00 y su despliegue en el *frontend*.

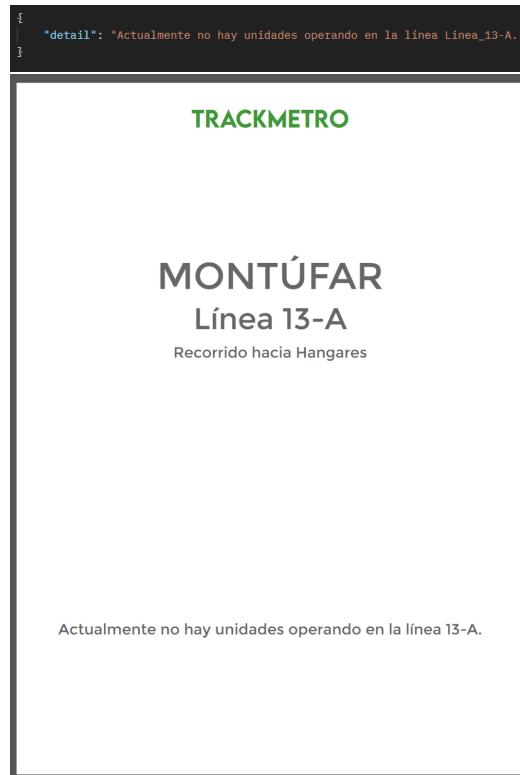


Figura 37: Respuesta final del *backend* implementado para la consulta del tiempo estimado de llegada a la estación “MONTÚFAR”, línea 13, dirección “VUELTA” el 13 de mayo de 2024 a las 23:00:00 y su despliegue en el *frontend*.

Cuando se solicita la predicción del tiempo estimado de duración de un viaje entre estaciones específicas, el *backend* implementado primero realiza el mismo proceso para obtener el mejor candidato para la estación de origen. Luego, utiliza los tiempos históricos A2A para calcular el tiempo estimado de duración del viaje entre las estaciones de origen y destino solicitadas. La figura 38 muestra la predicción del tiempo estimado de duración del viaje entre las estaciones “MONTÚFAR” y “FUERZA AÉREA” de la línea 13, dirección “VUELTA”,

```
El mejor candidato es la unidad: 205 con un ETA total de: 362.8535915510969
INFO: 127.0.0.1:9316 - "POST /trip HTTP/1.1" 200 OK

==== Calculando duración del viaje entre estaciones ====
ETA histórico desde MONTÚFAR hasta ACUEDUCTO : 150.0
ETA histórico desde ACUEDUCTO hasta FUERZA AÉREA : 177.0
Duración estimada del viaje entre MONTÚFAR y FUERZA AÉREA : 689.8535915510969
INFO: 127.0.0.1:9315 - "POST /trip HTTP/1.1" 200 OK
□
```

Figura 38: Predicción del tiempo estimado de duración del viaje entre las estaciones “MONTÚFAR” y “FUERZA AÉREA” el 13 de mayo de 2024 a las 14:38:00.

La figura 39 muestra la respuesta final del *backend* implementado para la consulta del tiempo estimado de duración del viaje entre las estaciones “MONTÚFAR” y “FUERZA AÉREA” de la línea 13, dirección “VUELTA”, así como su despliegue en el *frontend* implementado.

```
{
  "closest_unit": {
    "unit": "205",
    "latest_timestamp": "2024-05-13T14:37:41",
    "line": "Línea 13-A",
    "direction": "VUELTA",
    "next_station": "INDUSTRIA",
    "next_station_ETA": 72.61752147882821,
    "cumulative_ETA": 372.6175214788282,
    "latitude": 14.610753,
    "longitude": -90.51882,
    "stations_between": [
      "INDUSTRIA",
      "TIVOLI",
      "MONTÚFAR"
    ]
  },
  "prediction": 699.6175214788282,
  "model_version": "2025-10-25",
  "n_models": 1
}
```

### TRACKMETRO

**MONTÚFAR**  
 Línea 13-A  
 Recorrido hacia  
 Hangares

**FUERZA AÉREA**  
 Línea 13-A  
 Recorrido hacia  
 Hangares

**Duración total de tu viaje:**

**11:40**

desde ahora

Ubicación de la unidad más cercana (205):

Figura 39: Respuesta final del *backend* implementado para la consulta del tiempo estimado de duración del viaje entre las estaciones “MONTÚFAR” y “FUERZA AÉREA” el 13 de mayo de 2024 a las 14:38:00 y su despliegue en el *frontend*.

## CAPÍTULO 8

---

### Discusión

---

El proceso de limpieza de datos realizado mediante el conteo de estaciones diarias visitadas tuvo un impacto considerable en la reducción del volumen de estos. Sin embargo, en la figura 20, que presenta los puntos GPS de las trayectorias identificadas como no útiles para las unidades 049, 401 216, se observa que estos encuentran concentrados, respectivamente, en la Playa 12, ubicada en la Central de Mayoreo (CENMA) de Villa Nueva; en la Playa 2, al final de la calzada Atanasio Tzul; y en la Playa 4, sobre el Anillo Periférico. Esto indica que la reducción del 18.96 % de los datos iniciales indicada en el cuadro 6 verdaderamente corresponde a la eliminación de registros no útiles para el análisis.

Además, en la comparación ilustrada en la figura 21 se observa que los datos conservados tras la limpieza corresponden a trayectorias coherentes con las rutas oficiales del sistema Transmetro, por lo que el proceso de limpieza fue efectivo para garantizar que los datos utilizados en el análisis posterior fueran representativos del comportamiento operativo de las unidades del sistema.

La comparación entre el viaje 1 de la unidad 113 con el mapa de la línea 18 de Transmetro que se presenta en la figura 22 evidencia que la inferencia de línea y dirección para este mismo viaje que se presenta en el cuadro 8 es el correcto. Además, la figura 23 muestra que las estaciones teóricas inferidas para puntos GPS consecutivos de este viaje, coinciden con la trayectoria que está siguiendo la unidad sobre la línea 18.

De manera similar, la inferencia de línea y dirección para el viaje 146 de la unidad 204, que se presenta en el cuadro 9, también es correcta, ya que la unidad efectivamente inició su recorrido sobre la línea 1 en dirección CIRCULAR y luego continuó su viaje sobre la línea 6 en dirección IDA.

Esto indica que la preselección de líneas candidatas, la proyección de puntos GPS sobre las polilíneas y la aplicación del algoritmo de Viterbi fueron efectivas para inferir la línea y dirección inicial de cada viaje, y la inferencia de dirección actual y próxima estación teórica que se construyen sobre estas bases es coherente, incluso en casos donde las unidades cambiaron de línea durante su recorrido.

La matriz de correlación entre las características numéricas que se presenta en la figura 24 muestra relaciones coherentes con la variable objetivo `ETA_proxima_est_s`. En particular, la distancia a la próxima estación (`dist_a_prox_m`) presenta una correlación positiva moderada ( $r = 0.42$ ), mientras que la velocidad instantánea (`vel_mps`) y la posición curvilínea (`s_m`) muestran correlaciones negativas ( $r = -0.18$  y  $r = -0.17$ , respectivamente). Estas relaciones son consistentes con la lógica del fenómeno: las unidades más alejadas y con menor velocidad presentan mayores tiempos de llegada estimados. Las correlaciones moderadas y bajas entre las demás variables sugieren que las características capturan distintos aspectos del comportamiento de las unidades, reduciendo la multicolinealidad y mejorando la capacidad generalizadora de los modelos.

La limpieza realizada de acuerdo a las variables de tiempo de permanencia y tiempo estimada de llegada representaron una reducción de aproximadamente 16 % respecto al conjunto de datos, tal como se observa en el cuadro 10. Sin embargo, la distribución de la variable objetivo `ETA_proxima_est_s` que se presenta en la figura 25 muestra que el conjunto de datos final presenta una representación adecuada de los tiempos de llegada de las unidades cuando se encuentran en operación, con la mayoría de los datos concentrados alrededor de los 300 segundos (5 minutos) y una cola que se extiende hasta valores mayores a 1000 segundos (17 minutos aproximadamente). Esto indica que, a pesar de la reducción en el volumen de datos, la limpieza permitió conservar únicamente aquellos registros en los que las unidades se encontraban en operación, sin introducir sesgos significativos en la distribución de la variable objetivo, preservando la diversidad de situaciones operativas presentes en el sistema Transmetro.

El modelo LightGBM mostró un desempeño consistente en la predicción de tiempos de llegada. En el cuadro 11, se observa que alcanzó un error absoluto medio (MAE, por sus siglas en inglés) promedio de  $110.14 \pm 15.64$  segundos y un error cuadrático medio (RMSE, por sus siglas en inglés) promedio de  $302.04 \pm 50.82$  segundos durante la validación cruzada, mientras que en el conjunto de prueba final, el error aumentó moderadamente a 122.41 segundos de MAE y 344.11 segundos de RMSE, indicando una ligera pérdida de generalización, pero es de esperarse al evaluar sobre registros no vistos durante el entrenamiento. Estos resultados sugieren que el modelo es capaz de capturar las relaciones entre las variables espaciotemporales generadas durante la ingeniería de características y el tiempo estimado de llegada.

En términos de precisión operacional del modelo LightGBM, el cuadro 12 muestra que aproximadamente el 60.63 % de las predicciones del modelo en el conjunto de prueba final presentan un error absoluto menor o igual a 1 minuto, mientras que el 77.09 % y 84.66 % se encuentran dentro de los umbrales de 2 y 3 minutos, respectivamente. Dependiendo de la precisión requerida para la aplicación del modelo, estos resultados son un indicador de la utilidad práctica del este para estimar tiempos de llegada a la próxima estación teórica en el sistema Transmetro. Además, una desviación estándar promedio de tan solo 15.64 segundos en el MAE durante la validación cruzada sugiere una estabilidad significativa del modelo frente a la segmentación temporal empleada.

La curva de pérdida del modelo LightGBM que se presenta en la figura 26 muestra una rápida convergencia durante las primeras iteraciones, estabilizándose alrededor de la iteración 900, valor que coincide con la mediana de las mejores iteraciones observadas en los distintos *folds* durante la validación cruzada. A pesar de que la pérdida del conjunto de

validación es ligeramente superior a la del conjunto de entrenamiento, el MAE de la última iteración del entrenamiento final que se indica en el cuadro 13 evidencia una diferencia mínima entre ambos conjuntos (aproximadamente 3 segundos). Aún más importante, la pérdida del conjunto de entrenamiento y validación siempre disminuyen de manera paralela. Esto sugiere que el modelo no presenta un sobreajuste significativo.

El análisis desglosado por línea y dirección que se presenta en las figuras 27 y 28 revela heterogeneidad en el error según el contexto operacional. Líneas con trayectorias más predecibles, como la línea 12 (VUELTA) y la línea 13 (IDA), que cuentan con carriles exclusivos, alcanzaron MAE inferiores a 70 segundos y tasas de precisión operacional superiores al 85 % dentro del umbral de 3 minutos. En contraste, líneas con mayor exposición al tráfico vehicular, como las líneas 18 A/B y la línea 2, mostraron MAE superiores a 180 segundos y una menor proporción de aciertos dentro del umbral más estricto (1 minuto). Esta variabilidad refleja el impacto de las condiciones de tráfico, longitud de la ruta y número de estaciones intermedias sobre la dificultad del problema de predicción, más que una deficiencia intrínseca del modelo.

El análisis de importancia de características del modelo LightGBM que se presenta en la figura 29 revela que las variables espaciales y temporales constituyen el núcleo predictivo del modelo. Entre las más relevantes se encuentran la distancia del punto GPS al segmento de línea más cercano (`dist_m`), la próxima estación teórica (`prox_estacion_teorica`), la distancia a esta estación (`dist_a_prox_m`) y el avance en la posición curvilínea (`s_m`). Esto sugiere que el modelo prioriza información posicional continua y dinámica sobre atributos estáticos, como la línea o dirección, lo cual es coherente con la naturaleza secuencial del movimiento de las unidades. La variable de hora del día (`hour`) también mostró una importancia significativa, lo que sugiere que el modelo captura patrones temporales de demanda y congestión, modulando las predicciones según el contexto horario.

Como se observa en el cuadro 14, el modelo XGBoost presentó un desempeño ligeramente inferior al modelo LightGBM. En validación cruzada, alcanzó un MAE promedio de  $121.82 \pm 13.03$  segundos y un RMSE promedio de  $323.22 \pm 41.19$  segundos. En el conjunto de prueba final, el MAE obtenido fue de 125.87 segundos y el RMSE de 352.02 segundos, lo que indica una ligera pérdida de generalización similar a la observada en LightGBM. En términos de precisión operacional, el cuadro 15 muestra que 59.78 %, 76.28 % y 84.09 % de las predicciones del modelo quedaron dentro de los umbrales de 1, 2 y 3 minutos, respectivamente, lo que es ligeramente inferior a los resultados de LightGBM, pero aún así indica una utilidad práctica para la estimación de tiempos de llegada.

La curva de pérdida del modelo XGBoost que se presenta en la figura 30 muestra una rápida reducción del MAE en las primeras iteraciones, seguida de mejoras graduales hasta las 1604 iteraciones, evidenciando una convergencia más lenta en comparación con el modelo LightGBM. Esto es consistente con la dinámica de aprendizaje de XGBoost, que utiliza una estrategia de crecimiento enfocada en la profundidad. Sin embargo, al igual que la curva de LightGBM, la pérdida del conjunto de validación se mantiene cercana a la del conjunto de entrenamiento. Esto, junto a la pequeña diferencia en el MAE final entre ambos conjuntos que se muestra en el cuadro 16 (aproximadamente 5 segundos), sugiere que el modelo XGBoost tampoco presenta un sobreajuste significativo.

El desempeño del modelo XGBoost desglosado por línea y dirección, que se presenta en

las figuras 31 y 32, muestra patrones similares a los observados en LightGBM. Nuevamente, líneas con rutas más predecibles, como la línea 12 (VUELTA) y la línea 13 (IDA), presentaron un MAE inferior a 75 segundos y tasas de precisión operacional superiores al 80 % dentro del umbral de 3 minutos, mientras que líneas expuestas al tráfico vehicular, como las líneas 18 A/B y la línea 2, mostraron MAE superiores a 180 segundos y menores tasas de aciertos dentro del umbral más estricto (1 minuto).

La importancia de características del modelo XGBoost, que se presenta en la figura 33, confirma el papel escencial de las variables espaciales y de progreso: la distancia al segmento de línea más cercano (`dist_m`), la próxima estación teórica (`prox_estacion_teorica`), la distancia a esta estación (`dist_a_prox_m`) y la posición curvilínea (`s_m`) son las variables más relevantes. No obstante, a diferencia del modelo LightGBM, el modelo XGBoost asigna una mayor importancia a variables categóricas como la línea (`LINEA`) y la dirección (`DIR_init`), lo que sugiere que XGBoost puede beneficiarse más de información estática en su proceso de predicción, captando patrones específicos asociados a ciertas rutas y direcciones. Variables temporales como la hora del día (`hour`) también mostraron relevancia, indicando que ambos modelos reconocen la influencia de patrones temporales en los tiempos de llegada.

Con el conjunto de datos y las características generadas, el modelo LightGBM supera al modelo XGBoost: MAE de 122.41 segundos frente a 125.87 segundos, RMSE de 344.11 segundos frente a 352.02 segundos, y una mayor proporción de predicciones dentro de los umbrales de 1, 2 y 3 minutos. LightGBM parece priorizar con más fuerza las variables continuas de progreso espacial, mientras que XGBoost potencia el efecto de las categóricas (línea y dirección).

Los ejemplos de las estadísticas de tiempos históricos por tramos entre estaciones que se presentan en los cuadros 17 y 18 demuestran que los tiempos estimados de viaje entre llegada a estación A y llegada a estación B (A2A) son consistentemente mayores que los tiempos entre salida de estación A y llegada a estación B (D2A). Esto es coherente, ya que los tiempos A2A incluyen tanto el tiempo de viaje entre estaciones como el tiempo de permanencia de las unidades en la estación A, mientras que los tiempos D2A excluyen este último componente. Con estas dos nociones, el tiempo estimado de llegada de una estación a otra puede seleccionarse según la perspectiva del usuario: A2A es útil para los pasajeros de Transmetro que desean conocer el tiempo total hasta la llegada a su destino, ya que perciben tanto el tiempo de viaje como el tiempo de permanencia en estaciones intermedias. D2A, por otro lado, es más relevante para los operadores del sistema, quienes pueden estar interesados en optimizar los tiempos de espera y transferencia entre líneas. Además, se puede hacer uso de las diferentes estadísticas generadas para ambas nociones dependiendo del contexto (e.g., utilizar el percentil 90 para considerar situaciones con tráfico denso).

En las figuras 34 y 35 puede observarse como, al solicitarle al *backend* implementado el tiempo estimado de llegada a un estación específica mediante el *endpoint* POST /ETA, este primero obtiene las unidades candidatas que se encuentran operando en la línea sobre la cual se encuentra la estación solicitada y luego utiliza el modelo cargado en memoria (en este caso, el modelo LightGBM entrenado) para predecir el tiempo estimado de llegada de cada unidad candidata a su próxima estación teórica según su registro más reciente. Posteriormente, al tiempo estimado de llegada a la próxima estación teórica de cada unidad candidata se le suma el tiempo estimado histórico entre la próxima estación teórica y la estación solicitada, obtenido del almacenamiento de tiempos históricos entre estaciones (en

este caso, utilizando la perspectiva A2A y la media como nivel de servicio). Finalmente, el *backend* retorna la unidad candidata con el menor tiempo estimado de llegada a la estación solicitada, tal como se muestra en la figura 36.

La estrategia de selección de candidatos basada en la línea de la estación solicitada permite no solo mantener la coherencia operativa del sistema (ya que, lógicamente, una unidad solo puede llegar a una estación si está operando en la línea correspondiente), sino también evitar la necesidad de predecir tiempos de llegada para todas las unidades del sistema que se encuentren en operación, lo que optimiza el rendimiento del sistema y reduce la carga computacional en el *backend*.

También se observa en la figura 36 que el tiempo estimado de llegada a la estación solicitada se presenta en un formato claro y conciso, facilitando su interpretación por parte de los usuarios. Además, la complementación de la predicción del modelo con los tiempos históricos entre estaciones proporciona una estimación coherente respecto a las condiciones actuales de las unidades candidatas.

En el caso en el que no existan unidades candidatas operando en la línea de la estación solicitada debido a que su último registro tenga una vigencia mayor al umbral definido (en este caso, 1 hora), la figura 37 muestra que el *backend* retorna un mensaje informando esta situación, evitando de manera efectiva la generación de predicciones inválidas o incoherentes respecto al estado operativo actual del sistema Transmetro.

Finalmente, la figura 38 demuestra como, mediante el *endpoint* POST /trip, el *backend* es capaz de predecir la duración estimada de un viaje específico entre dos estaciones de la misma línea, realizando el mismo proceso de selección de unidades candidatas y predicción de tiempos de llegada a la estación de inicio del viaje, y luego sumando el tiempo estimado histórico entre las estaciones de inicio y fin del viaje al tiempo estimado de llegada a la estación de inicio de la unidad candidata con el menor tiempo estimado de llegada.

Como se muestra en la figura 39, el *backend* retorna la unidad candidata seleccionada con su duración estimada de viaje a la estación final del viaje indicado. Puede notarse que la unidad candidata seleccionada es la misma que la indicada en la respuesta de la predicción del tiempo estimado de llegada a la estación solicitada en la figura 36, lo que evidencia la coherencia de la estrategia implementada en el *backend* para ambos *endpoints*.



# CAPÍTULO 9

---

## Conclusiones

---

La limpieza de datos y la ingeniería de características permitieron transformar los registros crudos de posición GPS de las unidades de Transmetro proporcionados por la Empresa Municipal de Transporte (EMT) en un conjunto de datos estructurado y adecuado para la implementación de modelos predictivos; es decir, la información de la que dispone actualmente la EMT sobre la posición de las unidades de Transmetro, junto con la información disponible públicamente acerca de las líneas y estaciones del sistema, es suficiente para inferir, mediante la preselección de líneas candidatas, la proyección de puntos GPS sobre las polilíneas de dichas líneas y la aplicación del algoritmo de Viterbi, la línea y dirección de cada viaje realizado por las unidades del sistema, así como la dirección a la que se dirige cada unidad en cada registro temporal, su próxima estación teórica, la distancia restante hasta dicha estación y el tiempo estimado de llegada a la misma.

Las características generadas a partir de dicha inferencia espacial de las unidades de Transmetro y las características temporales derivadas de la fecha y hora de los registros capturan distintos aspectos del comportamiento operativo del sistema, presentando relaciones coherentes con la variable objetivo; i.e., el tiempo estimado de llegada a la próxima estación teórica.

La implementación de modelos de aprendizaje automático basados en árboles de decisión, particularmente LightGBM y XGBoost, permitió aprovechar las características generadas para predecir los tiempos estimados de llegada a la próxima estación teórica, logrando un desempeño estable y adecuado a diferentes umbrales de error absoluto en segundos. El modelo de LightGBM alcanzó un error absoluto medio (MAE, por sus siglas en inglés) de 122.41 segundos y un error cuadrático medio (RMSE, por sus siglas en inglés) de 344.11 segundos en el conjunto de prueba final, superando al modelo de XGBoost en todas las métricas de evaluación. Aproximadamente el 85 % de las predicciones de ambos modelos se encontraron dentro del umbral de tres minutos de error, demostrando una utilidad práctica para aplicaciones operativas y de información al usuario. Además, el análisis de importancia de características de ambos modelos confirmó que las variables espaciales y temporales, como la distancia a la próxima estación, la posición curvilínea y la hora del día, son los principales

determinantes del tiempo estimado de llegada, en coherencia con la naturaleza dinámica del transporte público.

La complementación de los modelos predictivos con tiempos históricos de viaje entre estaciones de una misma línea, calculados a partir de los datos preprocesados, permitió estimar la duración total de viajes entre estaciones, ofreciendo dos nociones distintas: desde llegada a llegada (A2A) y desde salida a llegada (D2A). Añadido a esto, cada una de estas nociones puede ser representada mediante diferentes estadísticas (media, mediana, percentiles), permitiendo adaptar las estimaciones a distintos contextos operativos y necesidades de los usuarios.

El diseño e implementación de un *backend* basado en una API RESTful utilizando FastAPI, que expone *endpoints* específicos para obtener tiempos estimados de llegada a estaciones y duraciones de viajes entre estaciones mediante la selección de la mejor unidad candidata de los registros de posición más recientes y la combinación de las predicciones de los modelos con estadísticas históricas entre estaciones garantiza coherencia en un contexto operativo. La integración de este *backend* con un *frontend* web desarrollado en React proporciona una interfaz fácil de interpretar para que potenciales usuarios del sistema Transmetro interactúen con los modelos predictivos y accedan a la información relevante de manera dinámica.

# CAPÍTULO 10

---

## Recomendaciones

---

Si se desean utilizar la interfaz o los modelos desarrollados en este trabajo en un ambiente de producción, es necesario contar con suficientes registros del día en curso para generar las características requeridas por los modelos predictivos. Por lo tanto, se recomienda:

- Implementar un sistema de almacenamiento en caché que retenga los registros recientes de cada unidad en memoria, permitiendo un acceso rápido y eficiente a los datos necesarios para la generación de características en tiempo real.
- Establecer un proceso de actualización continua que obtenga los registros más recientes de cada unidad de Transmetro desde la plataforma comercial utilizada por la EMT, los procese y luego los almacene en intervalos regulares, asegurando que las características derivadas estén siempre basadas en la información más reciente disponible.
- Monitorear el rendimiento del sistema de generación de características en producción para identificar y resolver posibles cuellos de botella o retrasos en la disponibilidad de datos.
- Considerar la implementación de mecanismos de tolerancia a fallos que permitan manejar situaciones en las que los datos del día en curso no estén disponibles, como el uso de datos históricos o promedios móviles para estimar las características faltantes.

Alternativamente, si se contara con una API u otra interfaz que evite la necesidad de obtener los registros de las unidades de Transmetro directamente desde la plataforma comercial utilizada por la EMT, se podría simplificar el proceso de generación de características y reducir la complejidad del sistema en producción.

Además, se recomienda evaluar periódicamente el desempeño de los modelos en producción y ajustar las estrategias de generación de características según sea necesario para mantener la precisión y relevancia de las predicciones.



# CAPÍTULO 11

---

## Bibliografía

---

- [1] A. T. Murray, R. Davis, R. J. Stimson y L. Ferreira, «Public Transportation Access,» *Transportation Research Part D: Transport and Environment*, vol. 3, págs. 319-328, 5 sep. de 1998, ISSN: 13619209. DOI: 10.1016/S1361-9209(98)00010-8
- [2] D. de Tránsito de la Dirección General de la Policía Nacional Civil, «Boletín Estadístico No. 12 2024,» Policía Nacional Civil, inf. téc., 2024.
- [3] D. de Tránsito de la Dirección General de la Policía Nacional Civil, «Boletín Estadístico No. 1 2025,» Policía Nacional Civil, inf. téc., 2025.
- [4] E. Quiñónez, *Más de 400 mil usuarios beneficiados con el Transmetro*, 2021.
- [5] G. Beirão y J. S. Cabral, «Understanding attitudes towards public transport and private car: A qualitative study,» *Transport Policy*, vol. 14, págs. 478-489, 6 nov. de 2007, ISSN: 0967070X. DOI: 10.1016/j.tranpol.2007.04.009
- [6] C. Paliwal y P. Biyani, «To each route its own ETA: A generative modeling framework for ETA prediction,» en *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, oct. de 2019, págs. 3076-3081, ISBN: 978-1-5386-7024-8. DOI: 10.1109/ITSC.2019.8917465
- [7] K. P. Murphy, *Machine learning : a probabilistic perspective*. MIT Press, 2012, pág. 1067, ISBN: 9780262018029.
- [8] Z. Zhang, Y. Yuan y X. ( Yang, «A Hybrid Machine Learning Approach for Freeway Traffic Speed Estimation,» *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2674, págs. 68-78, 10 oct. de 2020, ISSN: 0361-1981. DOI: 10.1177/0361198120935875
- [9] J. Wang, «Vehicular Traffic Flow Prediction Model Using Machine Learning-Based Model,» Unpublished manuscript, inf. téc., 2025.
- [10] S. V. Kumar, L. Vanajakshi y S. C. Subramanian, «A model based approach to predict stream travel time using public transit as probes,» en *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, jun. de 2011, págs. 101-106, ISBN: 978-1-4577-0890-9. DOI: 10.1109/IVS.2011.5940413

- [11] D. Hernández, «Activos y estructuras de oportunidades de movilidad: Una propuesta analítica para el estudio de la accesibilidad por transporte público, el bienestar y la equidad,» *EURE (Santiago)*, vol. 38, págs. 117-135, 115 sep. de 2012, ISSN: 0250-7161. DOI: 10.4067/S0250-71612012000300006
- [12] C. Brakewood y K. Watkins, «A literature review of the passenger benefits of real-time transit information,» *Transport Reviews*, vol. 39, págs. 327-356, 3 mayo de 2019, ISSN: 14645327. DOI: 10.1080/01441647.2018.1472147
- [13] A. M. Roshandeh y O. C. Puan, «Assessment of impact of variable message signs on traffic surveillance in Kuala Lumpur,» en *2009 IEEE International Conference on Intelligence and Security Informatics*, IEEE, 2009, págs. 223-225, ISBN: 978-1-4244-4172-3. DOI: 10.1109/ISI.2009.5137309
- [14] C. Mataix González, «Argumentos para la cultura,» *Fundación de la Energía*, 2010.
- [15] UN-Habitat, *Mobility and Transport*, <https://unhabitat.org/topic/mobility-and-transport>, 2025.
- [16] A. Amine y B. Arimah, *Planning and design for sustainable urban mobility : global report on human settlements 2013*. UN HABITAT ; Earthscan from Routledge, 2014, pág. 317, ISBN: 9789211319293.
- [17] H. Levinson, S. Zimmerman, J. Clinger y G. Rutherford, «Bus Rapid Transit: An Overview,» *Journal of Public Transportation*, vol. 5, págs. 1-30, 2 abr. de 2002, ISSN: 1077-291X. DOI: 10.5038/2375-0901.5.2.1 dirección: <https://www.sciencedirect.com/science/article/pii/S1077291X2200426X>
- [18] ANADIE, «Desarrollo e Infraestructura,» ANADIE, inf. téc., dic. de 2024. dirección: <https://anadie.gob.gt/wp-content/uploads/2025/01/RANADIE1.pdf>
- [19] D. de Tránsito de la Dirección General de la Policía Nacional Civil, «Boletín Estadístico No. 5 2025,» Policía Nacional Civil, inf. téc., 2025.
- [20] M. de Guatemala, *EMPRESA MUNICIPAL DE TRANSPORTE DE LA CIUDAD DE GUATEMALA Y SUS ÁREAS DE INFLUENCIA URBANA EMT*, [https://do.cs.muniguate.com/2022/memoria/arch/MEMORIA\\_DE\\_LABORES\\_2021\\_EMPRESA\\_MUNICIPAL\\_DE\\_TRANSPORTE\\_EMT.pdf](https://do.cs.muniguate.com/2022/memoria/arch/MEMORIA_DE_LABORES_2021_EMPRESA_MUNICIPAL_DE_TRANSPORTE_EMT.pdf), 2022.
- [21] M. de Guatemala, *Movilidad Urbana*, 2025. dirección: <https://www.muniguate.com/movilidadurbana/transmetro/>
- [22] M. M. Rahman, S. Wirasinghe y L. Kattan, «Users' views on current and future real-time bus information systems,» *Journal of Advanced Transportation*, vol. 47, págs. 336-354, 3 abr. de 2013, ISSN: 0197-6729. DOI: 10.1002/atr.1206
- [23] M. M. Rahman, *Real Time Bus Information Provision : users' Views and Some Aspects of Bus Travel Time in Calgary*. Library y Archives Canada = Bibliothèque et Archives Canada, 2012, ISBN: 9780494818978.
- [24] K. Dziekan y A. Vermeulen, «Psychological Effects of and Design Preferences for Real-Time Information Displays,» *Journal of Public Transportation*, vol. 9, págs. 1-19, 1 feb. de 2006, ISSN: 1077-291X. DOI: 10.5038/2375-0901.9.1.1
- [25] A. Pagani, F. Bruschi y V. Rana, «Time of arrival cumulative probability in public transportation travel assistance,» en *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, oct. de 2017, págs. 1-6, ISBN: 978-1-5386-1526-3. DOI: 10.1109/ITSC.2017.8317863

- [26] B. Yu, Z.-Z. Yang, K. Chen y B. Yu, «Hybrid model for prediction of bus arrival times at next station,» *Journal of Advanced Transportation*, vol. 44, págs. 193-204, 3 jul. de 2010, ISSN: 0197-6729. DOI: 10.1002/atr.136
- [27] S. Wang y L. Zhang, «A Review of GIS Technology Applications in Transportation Planning and Management,» *Scientific Journal of Technology*, vol. 6, págs. 108-113, abr. de 2024. DOI: 10.54691/e1s89w46
- [28] P. Neeley y A. Narkawicz, «Map Projection Induced Variations in Locations of Polygon Geofence Edges,» inf. téc., 2017.
- [29] S. M. Omohundro, «Five balltree construction algorithms,» 1989.
- [30] F. Pedregosa et al., «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.
- [31] J. P. Snyder, *Map projections-A working manual*. US Government Printing Office, 1987, vol. 1395.
- [32] M. A. Quddus, W. Y. Ochieng y R. B. Noland, «Current map-matching algorithms for transport applications: State-of-the art and future research directions,» *Transportation research part c: Emerging technologies*, vol. 15, n.º 5, págs. 312-328, 2007.
- [33] C. E. White, D. Bernstein y A. L. Kornhauser, «Some map matching algorithms for personal navigation assistants,» *Transportation research part c: emerging technologies*, vol. 8, n.º 1-6, págs. 91-108, 2000.
- [34] L. R. Rabiner, «A tutorial on hidden Markov models and selected applications in speech recognition,» *Proceedings of the IEEE*, vol. 77, n.º 2, págs. 257-286, 2002.
- [35] G. D. Forney, «The viterbi algorithm,» *Proceedings of the IEEE*, vol. 61, n.º 3, págs. 268-278, 2005.
- [36] J. H. Friedman, «1999 reitz lecture,» *Ann. Stat.*, vol. 29, n.º 5, págs. 1189-1232, 2001.
- [37] T. Chen y C. Guestrin, «Xgboost: A scalable tree boosting system,» en *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, págs. 785-794.
- [38] G. Ke et al., «Lightgbm: A highly efficient gradient boosting decision tree,» *Advances in neural information processing systems*, vol. 30, 2017.
- [39] C. J. Willmott y K. Matsuura, «Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,» *Climate research*, vol. 30, n.º 1, págs. 79-82, 2005.
- [40] C. Bergmeir, R. J. Hyndman y B. Koo, «A note on the validity of cross-validation for evaluating autoregressive time series prediction,» *Computational Statistics & Data Analysis*, vol. 120, págs. 70-83, 2018.
- [41] B. Lubanovic, *FastAPI : modern Python web development*. O'Reilly Media, Inc, 2024, ISBN: 9781098135508.
- [42] A. B. Mukherjee y E. Porcello, *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, 2018.



# CAPÍTULO 12

---

Anexos

---

## 12.1. Polilínea generada para cada línea de Transmetro

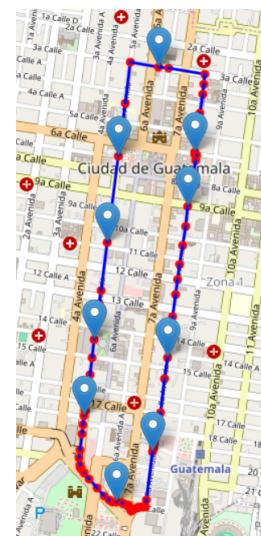


Figura 40: Polilínea generada para la línea 1 de Transmetro.

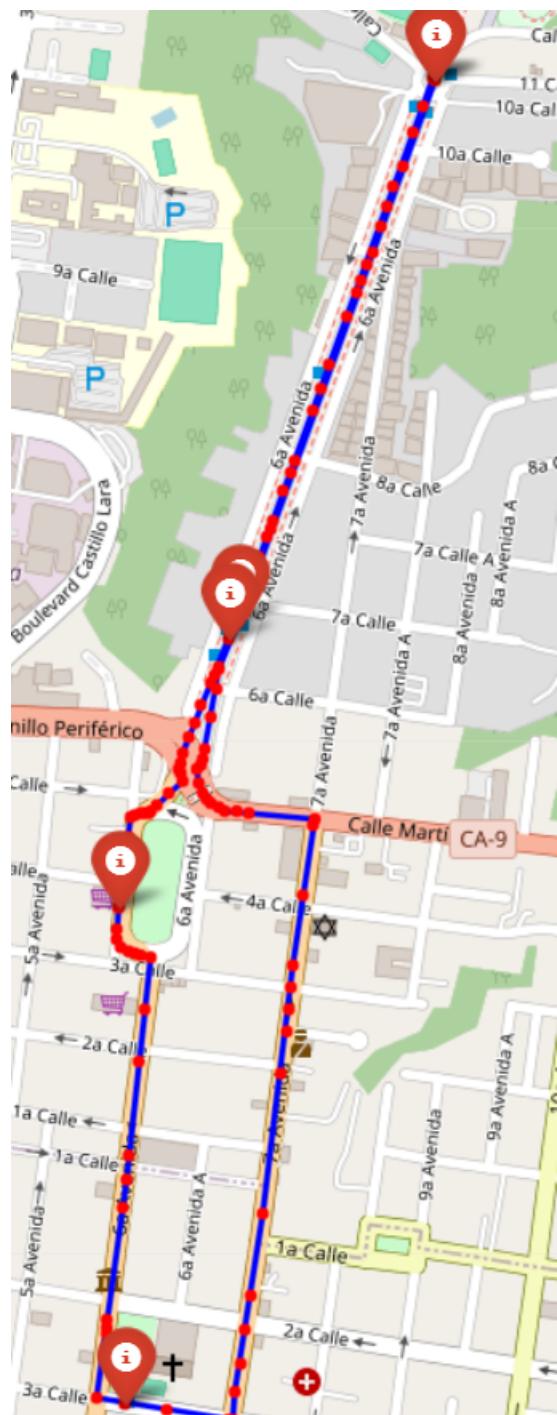


Figura 41: Polilínea generada para la línea 2 de Transmetro.

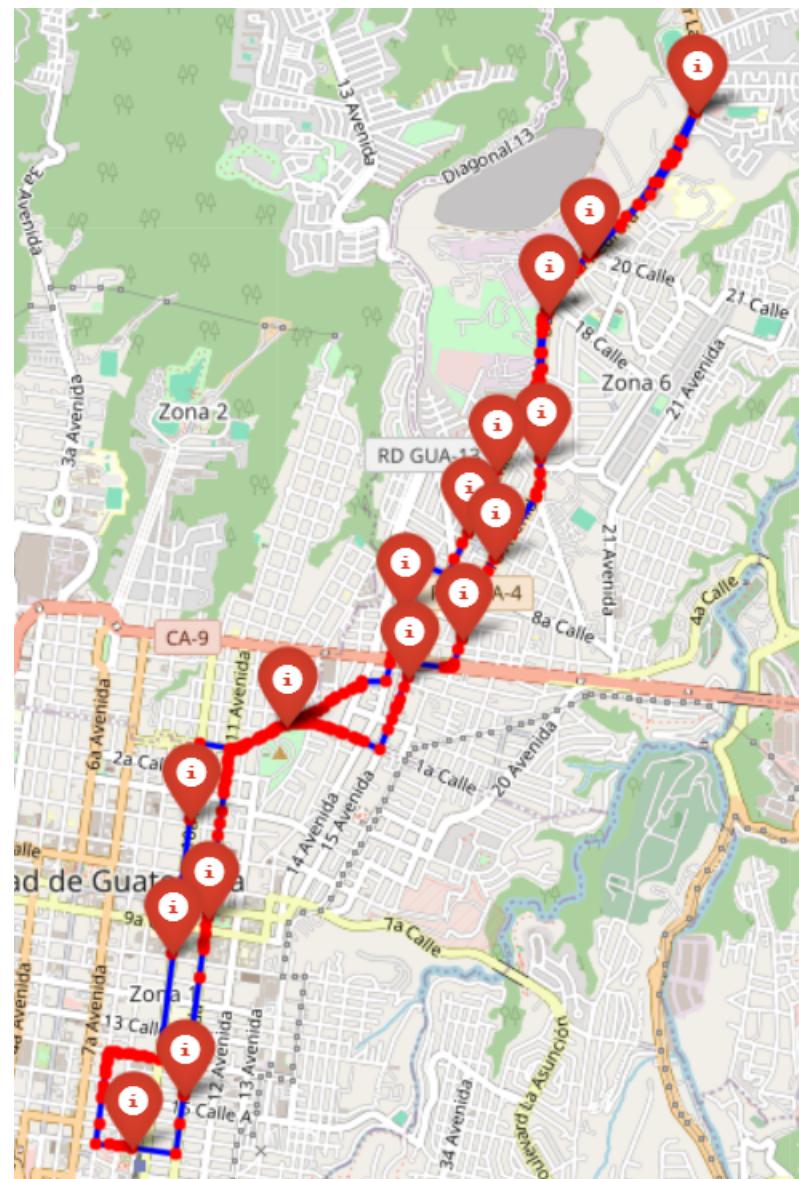


Figura 42: Polilínea generada para la línea 6 de Transmetro.

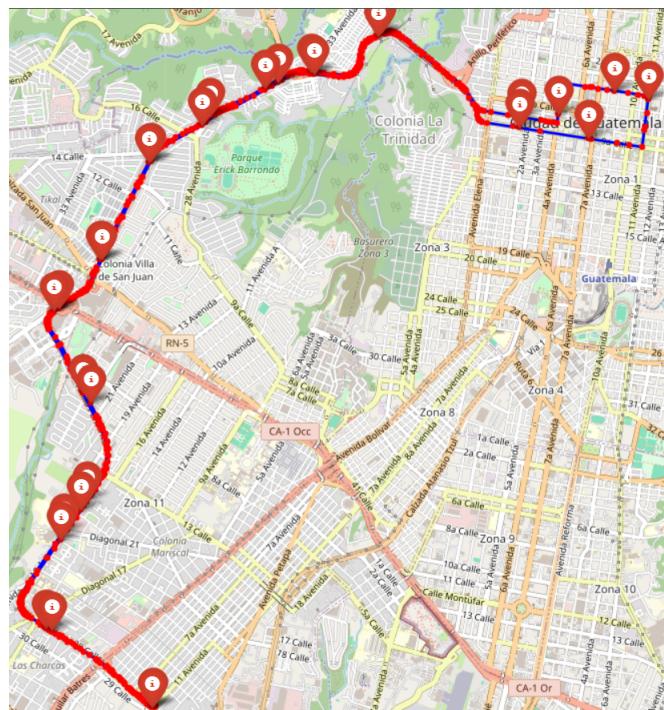


Figura 43: Polilínea generada para la línea 7 de Transmetro.

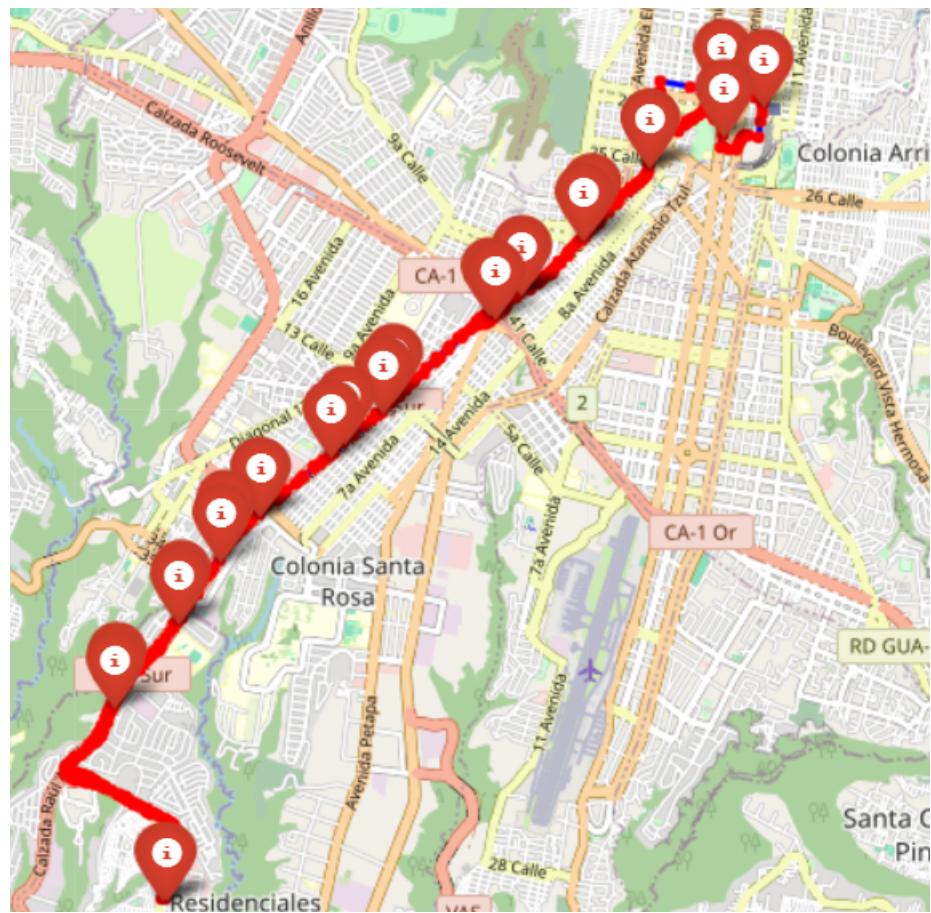


Figura 44: Polilínea generada para la línea 12 de Transmetro.

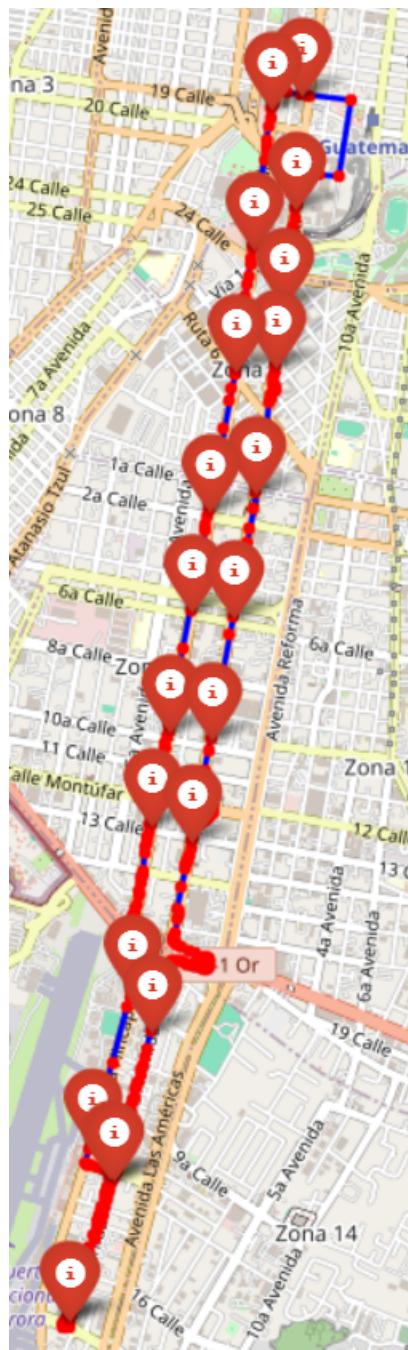


Figura 45: Polilínea generada para la línea 13-A de Transmetro.

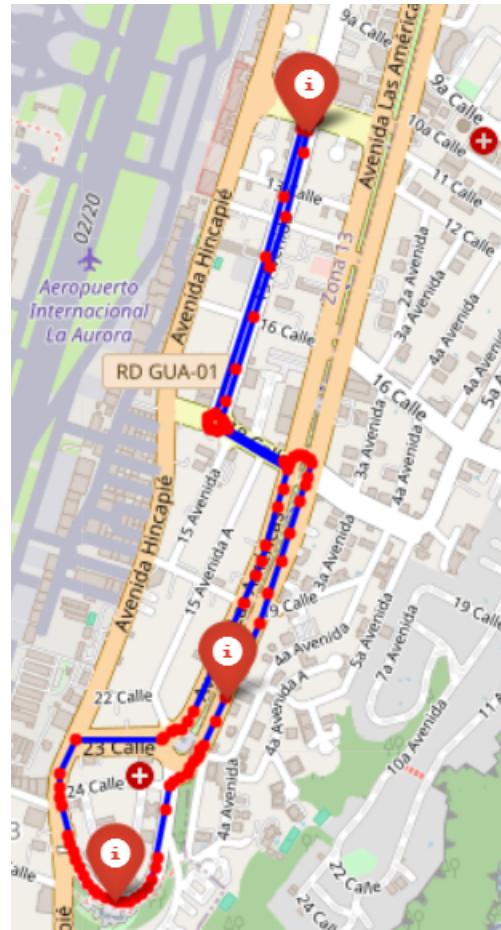


Figura 46: Polilínea generada para la línea 13-B de Transmetro.

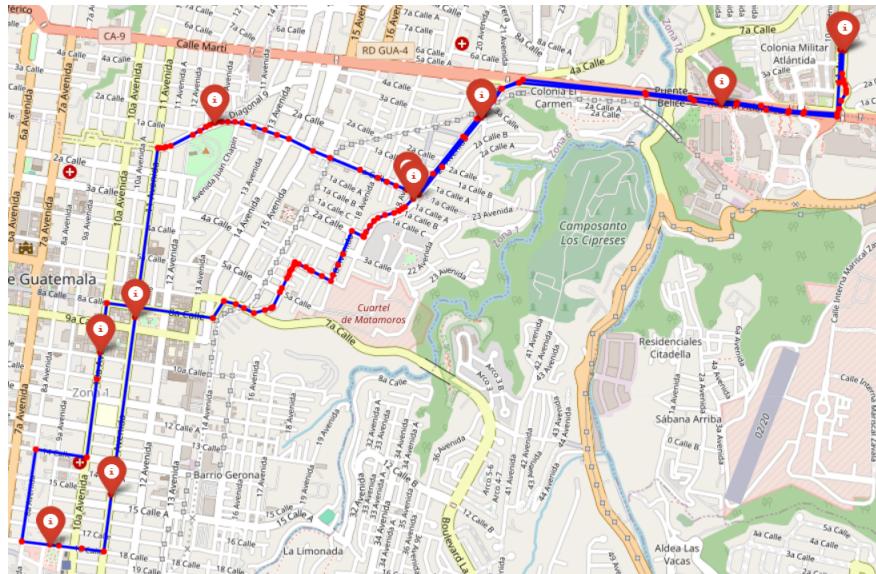


Figura 47: Polilínea generada para la línea 18-A de Transmetro.

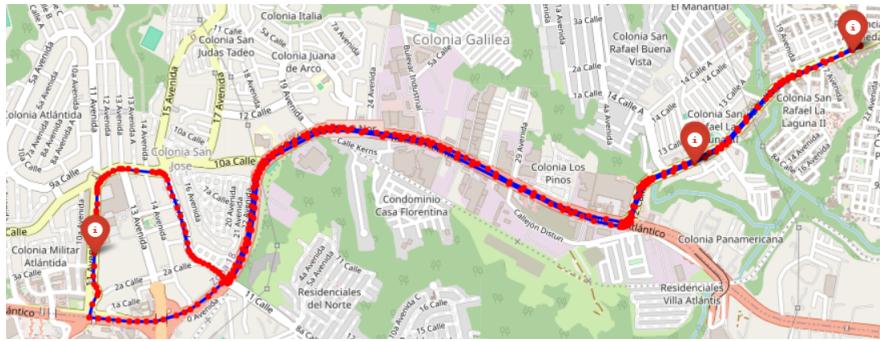


Figura 48: Polilínea generada para la línea 18-B de Transmetro.

## 12.2. Prototipos realizados en Figma

Prototipos realizados en Figma para el *frontend* de la interfaz de usuario:

- Versión *desktop*: [https://www.figma.com/proto/zVuXze7a9pkVi0Z1XrEx64/Wireframes-Transmetro?node\\_id=18-100&p=f&t=NQVNQGKf0C59ibB0-1&scaling=min-zoom&content-scaling=fixed&page\\_id=0%3A1](https://www.figma.com/proto/zVuXze7a9pkVi0Z1XrEx64/Wireframes-Transmetro?node_id=18-100&p=f&t=NQVNQGKf0C59ibB0-1&scaling=min-zoom&content-scaling=fixed&page_id=0%3A1)
- Versión *mobile*: [https://www.figma.com/proto/zVuXze7a9pkVi0Z1XrEx64/Wireframes-Transmetro?node\\_id=1-3&p=f&t=rxChw5ILyLYtuZKt-1&scaling=min-zoom&content-scaling=fixed&page\\_id=1%3A2&starting-point-node\\_id=1%3A3&show-sidebar=1](https://www.figma.com/proto/zVuXze7a9pkVi0Z1XrEx64/Wireframes-Transmetro?node_id=1-3&p=f&t=rxChw5ILyLYtuZKt-1&scaling=min-zoom&content-scaling=fixed&page_id=1%3A2&starting-point-node_id=1%3A3&show-sidebar=1)

## 12.3. Repositorios del proyecto

Los repositorios creados para el desarrollo del proyecto son los siguientes:

- Repositorio del *backend*: <https://github.com/pabloozamora/transmetro-predictor-backend.git>
- Repositorio del *frontend*: <https://github.com/pabloozamora/transmetro-predictor-frontend.git>

