PREDICCIÓN DINÁMICA DEL BUS BUNCHING EN EL COMPONENTE TRONCAL DEL SISTEMA TRANSMILENIO

Cristian Yesid Ríos Hernández
Código 9026669

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Directora
María Constanza Pabón Burbano

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, JUNIO DE 2026

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN

El sistema de transporte masivo TransMilenio, implementado en el año 2000, es la columna vertebral del Sistema Integrado de Transporte Público (SITP) de Bogotá. El sistema opera en cuatro componentes operativos: troncal (corredores para buses articulados y biarticulados sobre carriles totalmente segregados), zonal (buses padrones y estándar que ingresan a la malla vial urbana), alimentador (buses duales que conectan los barrios con las estaciones troncales) y Cable Aéreo. En conjunto, el sistema supera los 4 millones de validaciones (registros de ingreso de pasajeros) en un día hábil; de las cuales, el componente troncal aporta en promedio 2 millones de validaciones gracias a 114 km de corredores exclusivos, 9 portales y 142 estaciones troncales [1].

En esta operación de alta frecuencia se manifiesta el *bus bunching*, esto es, el agrupamiento de buses sucesivos de la misma ruta. Su origen está en una asimetría que se establece entre vehículos contiguos cuando uno de ellos sufre un retraso por una causa cualquiera (congestión vial, un abordaje más lento de lo previsto o un incidente puntual). El bus rezagado llega a la siguiente estación con una cola de pasajeros mayor a la programada, prolonga su detención para atender la demanda acumulada y se retrasa todavía más; el bus que lo sigue encuentra entonces una estación menos cargada, se detiene durante menos tiempo y reduce la distancia que lo separa del primero. Lejos de corregirse, la perturbación inicial se amplifica a lo largo del corredor, porque las fuerzas que actúan sobre cada bus (la demanda acumulada en estación y el tiempo de servicio asociado) refuerzan la separación entre el primero y el resto. El resultado visible es una alternancia de grupos de buses contiguos seguidos por intervalos prolongados sin servicio, y una desviación sistemática de la regularidad observada respecto de la programada. Las consecuencias se reparten entre el operador y el usuario: en el plano operativo, una reducción de la velocidad comercial efectiva del corredor y un uso menos eficiente de la flota desplegada; en el plano del usuario, tiempos de espera más largos en las estaciones, mayor variabilidad en la duración del viaje y un deterioro progresivo de la confiabilidad percibida del servicio.

Este trabajo de grado aborda la anterior situación operativa, es decir, cómo anticipar la formación de bus bunching con minutos de antelación para que el centro de control disponga de margen para aplicar acciones correctivas, desde la ciencia de datos. Su objetivo general es desarrollar un modelo predictivo que anticipe la formación de *bus bunching* en el componente troncal de TransMilenio a partir de los registros operativos del sistema, con el fin de apoyar la toma de decisiones del centro de control y la mejora de la regularidad del servicio. El alcance se concentra en la red troncal, que aporta el grueso de la demanda y dispone de la mejor cobertura de datos disponible. El planteamiento se apoya en la integración de las fuentes operativas centrales en un *dataset* analítico unificado, en un proceso de *feature engineering* guiado por el análisis exploratorio del fenómeno y en la comparación de varias familias de modelos predictivos, con métricas apropiadas al desbalance del problema y bajo un esquema de validación temporal que evita la fuga de información del futuro al pasado. El modelo finalmente seleccionado se somete a un análisis posterior de interpretabilidad de las variables que dominan sus decisiones y a pruebas de robustez ante perturbaciones de los datos de entrada.

Las implicaciones de un modelo capaz de anticipar el *bus bunching* van más allá del ejercicio técnico. Una predicción confiable con minutos de antelación abre la puerta a esquemas de control proactivo en el corredor, a tableros de riesgo agregados para la toma de decisiones operativas y, en una etapa posterior, a la integración con sistemas de control en lazo cerrado que ajusten la operación en tiempo real.

El presente documento presenta el proceso de entrenamiento y evaluación de dicho modelo, así como las limitaciones que persisten y las líneas que la investigación deja abiertas. Está organizado en una secuencia de capítulos que cubren, en orden, la definición del problema y los objetivos del proyecto, el marco teórico del fenómeno y los antecedentes en la literatura, la integración y preparación de los datos operativos, el análisis exploratorio y el preprocesamiento, el desarrollo y la evaluación comparativa de los modelos predictivos, la interpretabilidad y la robustez del modelo seleccionado, y las conclusiones del trabajo junto con sus limitaciones y las líneas de investigación futura.

Esta organización sigue las fases del estándar *CRISP-DM (Cross-Industry Standard Process for Data Mining)* [2], marco de referencia para proyectos de minería de datos. La comprensión del negocio se aborda en la definición del problema y los objetivos; la comprensión de los datos, en la integración de las fuentes y el análisis exploratorio; la preparación de los datos, en el preprocesamiento y el *feature engineering*; el modelado y la evaluación, en el desarrollo comparativo de los modelos y en su análisis de interpretabilidad y robustez; y la fase de despliegue se aborda de forma prospectiva en las implicaciones operativas y los trabajos futuros, al quedar una puesta en producción fuera del alcance de este trabajo.

1. DEFINICIÓN DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

El *bus bunching* o agrupamiento de buses se presenta cuando vehículos programados para circular con *headways* (intervalos regulares) terminan agrupándose. Un retraso inicial hace que más pasajeros se acumulen en la siguiente estación, lo que prolonga *dwell time* (tiempo de apertura de puertas) del bus retrasado y lo atrasa aún más, mientras el bus siguiente encuentra menos demanda y gana tiempo hasta alcanzarlo.

Dentro del componente troncal de TransMilenio, constituido por la red de corredores BRT (Bus Rapid Transit) con carriles exclusivos para buses articulados, la operación se basa en la programación de intervalos por ruta para distribuir equitativamente la demanda y asegurar tiempos de espera predecibles. Sin embargo, factores como la variabilidad de abordajes, la congestión causada por siniestros viales o vehículos varados, bloqueos derivados de manifestaciones y condiciones climáticas adversas alteran la operación planificada, rompen la cadencia y disparan el *bunching*. Las consecuencias son vacíos prolongados seguidos de llegadas múltiples, sobre-ocupación en el primer bus, baja utilización del segundo, pérdida de velocidad comercial y una percepción de servicio poco confiable para los usuarios.

La percepción del usuario es consistente con esta inestabilidad operacional. En la Encuesta de Satisfacción del Usuario de TransMilenio de diciembre de 2025 [3], el nivel de satisfacción global del componente troncal se ubicó en 78 %. Dentro de los atributos evaluados de operación de rutas, los relacionados con la regularidad y la espera presentan los siguientes resultados: el tiempo de espera para abordar el bus registra una satisfacción del 77,4 % y la cantidad de personas en el bus alcanza una satisfacción del 58,7 %. Estos resultados indican que los atributos más asociados al *bus bunchin* (intervalos irregulares, hacinamiento en unos buses y vacíos en otros), son hoy los de menor satisfacción relativa de operación de rutas en el componente troncal.

A pesar de contar con series históricas y casi en tiempo real de datos operativos, el control de la flota sigue siendo mayoritariamente reactivo, el personal de control detecta el agrupamiento cuando ya se ha materializado y, sobre la marcha, decide detener unos segundos el bus siguiente en la estación, adelantar un despacho desde patio, recortar un ciclo o autorizar que un vehículo se salte paradas. Estas acciones dependen en gran medida de la pericia del controlador en vía, de modo que la regularidad continúa deteriorándose y la percepción de confiabilidad del usuario se ve afectada. Sin un mecanismo predictivo que anticipe el *bunching* con suficiente antelación, resulta imposible aplicar las mismas acciones de forma oportuna, con un criterio uniforme y respaldadas por datos [4], [5].

1.2 FORMULACIÓN DEL PROBLEMA

La investigación se centra en responder ¿Cómo predecir la formación de *bus bunching* en el componente troncal de TransMilenio para apoyar la toma de decisiones operativas que mantengan intervalos regulares?

La predicción se aborda en este proyecto a nivel del par (servicio, estación) ordenado cronológicamente, no a nivel de bus individual. Esta elección se deriva de las fuentes operativas disponibles, que registran los eventos a nivel de servicio y estación, pero no incluyen un identificador de viaje ni de vehículo que permita seguir a un bus específico. Esta diferencia con la formulación clásica del problema en la literatura, que predice *n* paradas adelante del mismo bus a partir de información por vehículo, se asume desde la formulación del problema como una restricción estructural del trabajo y reaparece a lo largo del documento en las decisiones de modelado.

1.2.1 Sistematización

Para hacerlo, será necesario responder a interrogantes clave tales como:

- ¿Qué datos operativos, de demanda, oferta y contexto deben integrarse, y con qué criterios de calidad, para caracterizar el *bus bunching* en el componente troncal de TransMilenio?
- ¿Cuáles transformaciones estadísticas y de *feature engineering* revelan los patrones y variables que mejor caracterizan la aparición y evolución del *bus bunching*?
- Entre los algoritmos considerados (árboles de gradiente, *LSTM*, híbridos, etc.), ¿cuál ofrece la mayor precisión y *lead-time* útil para anticipar el fenómeno con fiabilidad operacional?
- ¿Cómo afectan los cambios moderados en los rangos de entrada a la estabilidad del modelo, y qué umbrales mínimos de perturbación desestabilizan la predicción?

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Desarrollar un modelo predictivo que permita predecir la formación de *bus bunching* en la red troncal de TransMilenio, utilizando técnicas de análisis de datos y aprendizaje automático, con el fin de proporcionar una herramienta que facilite la toma de decisiones operativas orientadas a mejorar la regularidad del servicio.

2.2 OBJETIVOS ESPECÍFICOS

- Recolectar, depurar y documentar un conjunto de datos operativos e históricos de TransMilenio, incluyendo variables internas y externas relevantes para el análisis del *bus bunching*.
- Realizar un análisis exploratorio de los datos para identificar patrones y relaciones, y aplicar técnicas de *feature engineering* para construir variables explicativas significativas del fenómeno.
- Desarrollar, entrenar y comparar modelos predictivos (como árboles de decisión, redes neuronales *LSTM* y modelos híbridos), evaluando su desempeño mediante métricas específicas de clasificación.
- Analizar interpretabilidad y robustez del modelo seleccionado, mediante técnicas como importancia de variables (*gain, SHAP*) y pruebas de sensibilidad frente a ruido en las entradas.

3. MARCO TEÓRICO Y ANTECEDENTES

3.1 MARCO TEÓRICO

3.1.1 Fundamentos del fenómeno de Bus Bunching

Definiciones y notación

El intervalo o *headway* $h_{i,j,t}$ se define como el tiempo transcurrido entre el arribo del bus i a la estación j y el arribo del bus inmediatamente anterior de la misma ruta, registrado en el instante t. El *headway* programado $h^0_{i,j,t}$ es el intervalo nominal derivado de la tabla de programación de despachos para ese servicio y ese tramo horario. El *dwell time* $d_{i,j,t}$ es el tiempo de puertas abiertas del bus i en la estación j en el instante *t* de arribo. La demanda o *Passenger arrival rate* $p_{j,}$corresponde al flujo de pasajeros que arriban a la plataforma o estación *j* durante el intervalo entre dos buses consecutivos.

En lo que sigue, el subíndice i se refiere al bus individual (i-ésimo arribo de la ruta), el subíndice j a la estación o parada, y el subíndice t al instante en que el bus i arriba a la estación j. La definición de *bus bunching* aplica por tanto a un par (servicio, estación) en un instante de tiempo determinado, evaluando la separación efectiva respecto del bus inmediatamente anterior del mismo servicio. Esta limitación de las fuentes tiene una consecuencia directa sobre la formulación del problema en este proyecto: aunque la literatura y las definiciones de los párrafos anteriores se refieren al bus i, la unidad de predicción adoptada en los capítulos siguientes es el par (servicio, estación) ordenado cronológicamente. El *bunching* se evalúa, por tanto, entre arribos consecutivos del mismo servicio a la misma estación, no entre buses individuales rastreados a lo largo de su recorrido.

El criterio operativo más extendido en la literatura define *bus bunching* en la estación j cuando el *headway* real cae por debajo de una fracción α del *headway* programado:

$$h_{i,j,t} \leq \alpha \cdot h^0_{i,j,t}, \quad \alpha \in (0,1) \tag{1}$$

El valor α = 0,25 es el más empleado en estudios predictivos basados en tarjeta inteligente y GPS [5], [6], mientras que α = 0,5 se usa como umbral más laxo en evaluaciones operativas. Sin embargo, no existe un umbral único: algunos estudios emplean valores fijos que oscilan entre los 20 segundos y los 3 minutos [7], por lo que la elección del umbral dependerá del objetivo del análisis y del nivel de tolerancia operativa adoptado.

El fenómeno opuesto, el *gap*, se define análogamente como $h_{i,j,t} \geq \beta \cdot h^0_{i,j,t}$ con β típicamente igual a 2 [7].

Mecanismo de bus bunching

El *bus bunching* surge de un bucle de retroalimentación positiva entre tres variables operativas interdependientes: *headway*, demanda y *dwell time*. Un retraso inicial (provocado por tráfico o semáforos) amplía el *headway* respecto al bus precedente, lo que incrementa la demanda acumulada en plataforma/estación. Esta mayor afluencia de pasajeros eleva el *dwell time* en la siguiente parada, retrasando aún más al vehículo. Mientras tanto, el bus siguiente al encontrar paradas con menor ocupación y fricción, acelera su marcha hasta alcanzar al primero, colapsando la frecuencia del servicio y destruyendo la regularidad operativa (Ver Figura 1).

Daganzo [8] formaliza este fenómeno como una inestabilidad estructural del sistema de transporte sin control, no como una falla operativa puntual. Su argumento parte de una ley de movimiento que describe cómo se propaga la hora de llegada de cada bus a lo largo del corredor: una perturbación leve en un vehículo se amplifica progresivamente en lugar de disiparse, de modo que el agrupamiento emerge incluso bajo condiciones idealizadas como buses idénticos, conductores homogéneos y demanda uniforme. Daganzo expresa esta dinámica mediante la ecuación:

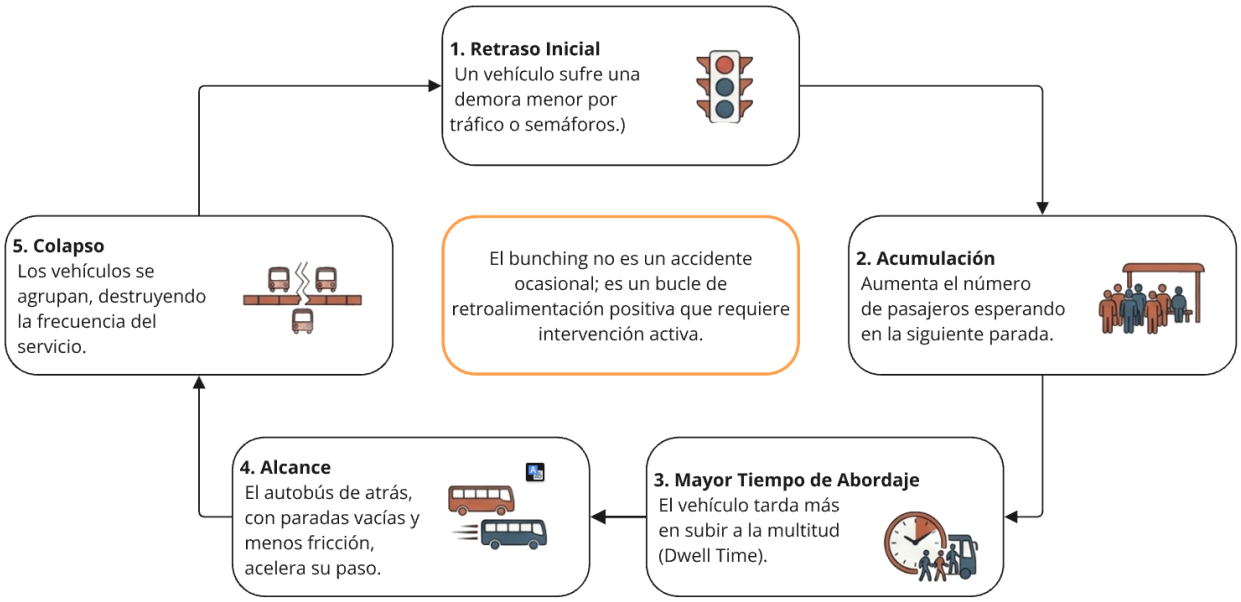
$$a_{i,j+1,t} = a_{i,j,t} + c_j + b_j\,(h_{i,j,t} - h^0_{i,j,t}) + \nu_{i,j+1,t} \qquad (2)$$

Donde:

- $a_{i,j,t}$: hora real de llegada del bus *i* a la estación *j* en el instante *t*.
- $a_{i,j+1,t}$: hora de llegada del mismo bus a la siguiente estación *j+1*.
- c_j: tiempo nominal de recorrido entre las estaciones *j* y *j+1* en condiciones normales.
- b_j: coeficiente de sensibilidad ($b_j > 0$) que captura el efecto de la demanda acumulada sobre el tiempo de servicio en la estación *j*.
- $h_{i,j,t}$: *headway* real entre el bus *i* y el bus anterior en la estación *j*, definido como la diferencia $a_{i,j,t} - a_{i-1,j,t}$.
- $h^0_{i,j,t}$: *headway* programado para ese servicio y tramo horario, derivado de la tabla de despachos.
- $\nu_{i,j+1,t}$: ruido aleatorio asociado al segmento entre las estaciones *j* y *j+1*. Agrupa todas perturbaciones no modeladas que afectan el tiempo de viaje.

El término $b_j\,(h_{i,j,t} - h^0_{i,j,t})$ es el responsable directo del *bunching*: al ser positivo cuando el *headway* real supera al programado, retrasa aún más al bus rezagado, evidenciando que sin un mecanismo de control el sistema es matemáticamente inestable.

Figura 1. Bucle de realimentación positiva



Fuente: Elaboración propia.

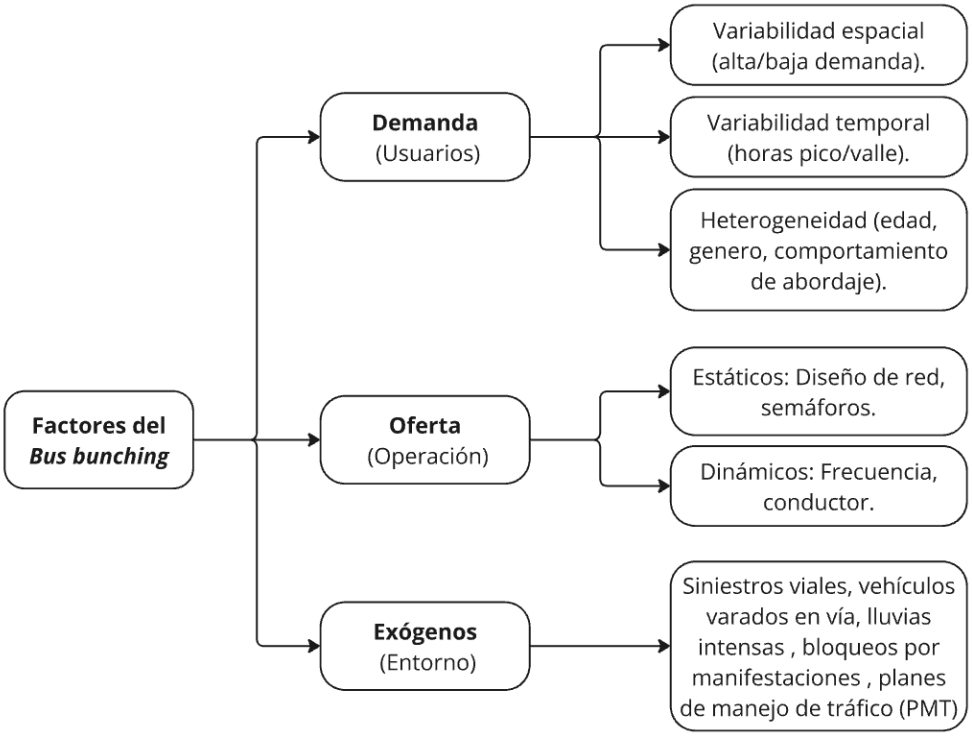
Factores que amplifican la inestabilidad

Los factores que precipitan el *bus bunching* en corredores BRT se han caracterizado en múltiples estudios. Rezazada *et al.* [7], en una revisión sistemática, agrupan dichos factores en tres grandes categorías: causas relacionadas con la demanda (variabilidad espacial y temporal, heterogeneidad de los usuarios), causas relacionadas con la oferta (diseño de red, configuración de la ruta, frecuencia, tipología de flota, *dwell time* y derechos de vía compartidos) y factores exógenos (clima y eventos externos no recurrentes como siniestros viales). Para el caso particular de TransMilenio, Naranjo [9] complementa esta visión al ubicar geográficamente los desfases en cuatro puntos del ciclo operativo (zonas de despacho, intersecciones, vías y puntos de parada) y concentra su análisis en las estaciones intermedias por ser el eslabón con capacidad más restringida.

A partir de ambos marcos, en la Figura 2 se sintetizan los factores observados en el componente troncal de TransMilenio organizándolos en tres categorías: causas de demanda, causas de oferta y operación, y eventos exógenos. El desfase que estos factores inducen puede monitorearse con las fuentes disponibles (AFC, GPS y programación GTFS) a lo largo de tres etapas del ciclo operativo del vehículo: cabeceras, recorrido entre estaciones y estaciones intermedias. En las *cabeceras o portales* predominan causas de oferta y operación (indisponibilidad del conductor o del vehículo, vehículos varados antes de salir del portal y ajustes de despacho por regulación de flota), y la única variable observable en este punto es la hora de despacho real frente a la programada. Durante el *recorrido entre estaciones* se superponen factores de oferta (tipología mixta de flota y convivencia de servicios en una misma plataforma [9], [10]) con eventos exógenos (siniestros viales, vehículos varados en vía, lluvias intensas y, en el caso particular de Bogotá, planes de

manejo de tráfico (PMT) y bloqueos puntuales de usuarios), que alteran la velocidad instantánea y el tiempo entre estaciones. En las *estaciones intermedias*, la variabilidad de abordajes (fenómeno conducido por la demanda [7], [10]) y la congestión de usuarios en plataforma derivada de la interacción demanda-capacidad introducen desviaciones adicionales que se reflejan en el *dwell time*, la ocupación y el *headway* con respecto al bus precedente. El ciclo culmina en la decisión del controlador: si el desfase acumulado se corrige oportunamente mediante acciones correctivas en tiempo real (*holding*, *skip-stop*, reasignación o ajuste de velocidad) el servicio preserva su regularidad; en caso contrario se configura el agrupamiento.

Figura 2. Factores que ocasionan el *bus bunching*



Fuente: Elaboración propia.

El bucle canónico de retroalimentación positiva, descrito por Daganzo [8] y formalizado por Moreira-Matias *et al.* [4], opera con dos brazos simultáneos representados explícitamente en la Figura 1. Por un lado, el bus retrasado encuentra cada vez más pasajeros aguas abajo: un retraso en la estación j incrementa los pasajeros en plataforma en $j+1$, alarga el *dwell time* en $j+1$ y amplifica el retraso en $j+2$, $j+3$… Por el otro, el bus siguiente de la misma ruta encuentra menos pasajeros y *dwell times* más cortos, por lo que alcanza gradualmente al bus retrasado y se configura el agrupamiento. A este bucle canónico, en sistemas BRT con cabeceras controladas como TransMilenio, se suma un efecto operativo de mayor escala temporal: un bus que regresa retrasado al portal desplaza el siguiente despacho de su servicio. Este efecto no forma parte del

bucle canónico de *bunching* y se mitiga con instrumentos de regulación de flota en cabecera, mientras que las acciones correctivas en vía (*holding*, *skip-stop*, ajuste de velocidad, reasignación) actúan sobre los dos brazos del bucle propiamente dicho.

Particularidades en sistemas BRT

La mayor parte de la literatura sobre *bus bunching* se ha desarrollado sobre rutas de bus convencional que operan en tráfico mixto. Los sistemas BRT como TransMilenio introducen diferencias estructurales que condicionan tanto la dinámica del fenómeno como su medición. Las estaciones son cerradas y la validación de pasaje se realiza al ingreso de la estación, no a bordo; por tanto, la demanda observable por *AFC (Automated Fare Collection*) se asocia a la estación, pero para el componente troncal no al bus específico en que embarcó el usuario. Los carriles son segregados, lo que atenúa la influencia del tráfico externo, pero concentra la variabilidad en las estaciones y en los puntos de cruce con vías mixtas. La flota es heterogénea (articulados, biarticulados, duales) y distintos servicios comparten las mismas plataformas, de modo que un desfase en una ruta puede bloquear físicamente el vagón y afectar al resto. Naranjo [9] y Peña y Moreno [10] documentan cómo esta convivencia de servicios y tipologías en estaciones intermedias incrementa la variabilidad del tiempo de servicio y es un factor diferenciador frente a rutas de bus convencional.

3.1.2 Datos, indicadores y estrategias operativas

Fuentes de datos en sistemas BRT

La predicción de *bus bunching* requiere información operativa de alta granularidad temporal y espacial, que en TransMilenio y en la mayoría de los sistemas BRT modernos se obtiene de tres fuentes principales. El sistema GPS/AVL (*Automatic Vehicle Location*) registra la posición del vehículo a frecuencia típica de pocos segundos y permite derivar *headway*, velocidad instantánea, tiempo de viaje entre estaciones y *dwell time* aproximado. El sistema AFC (*Automatic Fare Collection*) almacena cada validación de tarjeta con marca de tiempo y estación, lo que permite estimar demanda de abordaje por estación y franja horaria [5], [11]. El sistema GTFS (*General Transit Feed Specification*) proporciona la programación nominal: rutas, paradas, intervalos programados y horarios, y es la referencia contra la cual se comparan las observaciones [12]. A estas fuentes se suman, cuando están disponibles, mediciones de ocupación por contadores automáticos de pasajeros (APC), datos meteorológicos y registros de eventos de tráfico.

Indicadores de regularidad del servicio

La literatura compara modelos y estrategias de regulación utilizando un conjunto relativamente estable de indicadores, orientados a medir la regularidad de los *headways* y el impacto operativo de las perturbaciones [7], [13], [14]. La Tabla 1 resume las métricas más empleadas.

Tabla 1. Indicadores de regularidad del servicio

Métrica	**Descripción**	**Uso típico**
Coeficiente de variación del *headway*	Razón entre desviación estándar y media de los intervalos observados	Variabilidad global del servicio
Bunching share	Porcentaje de despachos con *headway* real ≤ α·*headway* programado	Incidencia del fenómeno
Pérdida de velocidad comercial	Diferencia porcentual entre la velocidad real y la de programación	Efecto económico-operativo

Fuente: Elaboración propia con base en [7], [13], [14].

Las tres métricas recogidas son las de uso más extendido en la literatura revisada, pero no agotan el catálogo disponible. Los operadores de transporte masivo mantienen tableros internos con indicadores adicionales (entre otros, el *Excess Waiting Time* (EWT), el *Wait Assessment*, medidas de adherencia al *headway* o al itinerario, índices de puntualidad y ventanas de tolerancia propias del sistema), cuyo diseño responde a las necesidades específicas de cada red (patrón de demanda, densidad de la infraestructura, política tarifaria, compromisos contractuales con los concesionarios, niveles de servicio acordados con el regulador) y a las particularidades del dato disponible. Muchos de esos indicadores no aparecen documentados en publicaciones revisadas por pares (circulan como parte de manuales técnicos internos, informes de gestión o cláusulas contractuales) y sus definiciones operativas pueden diferir entre sistemas. En consecuencia, al comparar desempeño entre ciudades es necesario verificar con precisión cómo se calcula cada indicador: dos tableros pueden reportar regularidad con métricas construidas de forma distinta, sin que la etiqueta permita inferirlo.

Estrategias clásicas operativas de control

Las estrategias operativas para mitigar el agrupamiento actúan sobre la posición relativa de los vehículos o sobre su programa de despacho, y pueden ejecutarse de forma reactiva por el controlador o de forma proactiva cuando existen modelos predictivos [7], [8],[14].

- *Headway-holding*: retener unos segundos al bus adelantado en un punto de control para restablecer el intervalo objetivo. Daganzo [8] formaliza esta estrategia como un esquema adaptativo basado en el *headway* observado y muestra que requiere menos *slack* programado (el tiempo adicional incorporado al recorrido programado como margen para absorber retrasos sin propagarlos al servicio) que los esquemas convencionales.
- Control dinámico de velocidad: ajustar consignas de velocidad a vehículos adelantados o retrasados dentro de límites operativos.
- *Skip-stop*: autorizar a un bus retrasado a omitir una o varias paradas de baja demanda mientras el siguiente absorbe la carga.
- Reasignación de despacho: adelantar o retrasar salidas desde patios o terminales para rellenar vacíos o evitar sobreoferta.

La efectividad de estas acciones depende de su oportunidad: aplicadas tarde, no evitan el agrupamiento; aplicadas sin información suficiente, pueden inducir perturbaciones adicionales. Esta es la razón por la que la literatura reciente combina estrategias de control con módulos de predicción que anticipan el evento antes de que ocurra [4], [14].

3.1.3 Modelado predictivo del *bus bunching*

Familias de modelos predictivos

El agrupamiento de buses se modela de forma natural como un problema de series de tiempo multivariadas: la variable objetivo (estado de operación del corredor (fluido, irregular, *bunched*) o el *headway* aguas abajo) depende de la secuencia histórica de *headways*, tiempos de detención, ocupación y variables exógenas [4], [13], [14]. En la literatura revisada se distinguen cinco grandes familias de modelos supervisados, que la Tabla 2 resume conceptualmente, así como la descripción de cómo cada familia incorpora la dependencia temporal de los datos en su formulación.

Tabla 2. Familias de modelos para predicción de *bus bunching*

Familia	**Formulación**	**Manejo de la dependencia temporal**
Máquinas de soporte vectorial	Clasificación o regresión de margen máximo con *kernels* no lineales (RBF); versiones de mínimos cuadrados (*LS-SVM*) y ε-insensible (ε-SVR) para regresión del *headway*	Rezagos manuales (*lag features*)
Ensambles de árboles	*Random Forest, XGBoost, LightGBM, CatBoost*; ensambles de votación ponderada entre varios algoritmos, con aprendizaje incremental por lotes	Rezagos manuales
Redes recurrentes profundas	*LSTM* y GRU con o sin mecanismo de atención; arquitecturas *sequence-to-sequence* (encoder-decoder)	Nativo (secuencia temporal)
Modelos de estados y transición	Cadenas de Markov sobre estados operativos obtenidos por *clustering* (*K-means*); modelos logísticos multinomiales como predictor complementario	Nativo (matriz de transición)
Marcos de aprendizaje en línea	Regresores incrementales (*Perceptron Delta Rule*, *Exponential Delta Rule*, ANN actualizada en línea, regresión lineal o kernel regression) embebidos en esquemas de predicción-control que se actualizan con cada nuevo dato	Nativo (actualización continua)

Fuente: Elaboración propia.

La tabla presenta las cinco familias en torno a las que se organiza la literatura sobre predicción de *bus bunching*. En los párrafos siguientes se describen sus bases metodológicas, las decisiones estructurales y las variantes más usadas, con referencia a los trabajos en aprendizaje automático y mención puntual a las aplicaciones representativas dentro del dominio.

Máquinas de soporte vectorial. Las *SVM* son un método de aprendizaje supervisado que separa dos clases trazando una frontera de decisión que mantenga la mayor distancia posible respecto a los ejemplos de cada clase. Esa idea de margen máximo se formaliza en el siguiente problema de optimización [15]:

$$\min_{w,\,b,\,\xi} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i \tag{3}$$

El primer término controla el ancho del margen y el segundo penaliza los ejemplos mal clasificados. El parámetro *C* regula el compromiso entre ambos: valores altos hacen el modelo más estricto frente a errores de clasificación, valores bajos toleran más violaciones a cambio de un margen más amplio.

Cuando los datos no son linealmente separables se utiliza el llamado *kernel trick*: en lugar de transformar explícitamente los datos a un espacio de mayor dimensión donde sí lo sean, se reemplaza el producto interno entre puntos por una función núcleo que produce el mismo efecto. La elección más habitual es el *kernel* gaussiano (RBF), que mide la similitud entre dos ejemplos en función de su distancia:

$$K(x, x') = \exp(-\gamma \| x - x' \|^2) \tag{4}$$

La formulación *LS-SVM* reemplaza las restricciones por igualdades, con lo que la solución se obtiene resolviendo un sistema lineal en vez de un problema de programación cuadrática y por eso entrena más rápido. La variante ε-SVR adapta el método a regresión: predice directamente el *headway* ignorando errores menores que una tolerancia ε. Para series temporales, la dimensión temporal se incorpora construyendo manualmente variables rezagadas.

Ensambles de árboles. Un árbol de decisión particiona el espacio de entrada en regiones mediante una serie de preguntas del tipo "¿la variable j es menor o igual que un umbral *s*?" y asigna a cada región terminal una predicción, un valor promedio para regresión, una clase mayoritaria para clasificación. Un solo árbol tiende a sobreajustar los datos; los ensambles combinan muchos árboles para reducir esa inestabilidad.

Existen dos estrategias de combinación. En *Random Forest* [16], cada árbol se entrena con una muestra aleatoria de los datos y de las variables, y la predicción final es el promedio (regresión) o el voto mayoritario (clasificación) de todos los árboles, lo que reduce la varianza del modelo. En *gradient boosting* [17], los árboles se entrenan secuencialmente: cada nuevo árbol corrige los errores acumulados del ensamble anterior. La actualización del modelo en la iteración m tiene la forma:

$$F_m(x) = F_{m-1}(x) + \eta\, h_m(x), \tag{5}$$

donde h_m es el árbol nuevo y η la tasa de aprendizaje, que controla cuánto pesa cada nuevo árbol respecto al ensamble previo. Las implementaciones modernas (*XGBoost [18], LightGBM, CatBoost*) añaden regularización sobre las hojas, expansión de Taylor de segundo orden de la función de pérdida y aproximaciones por histogramas para acelerar el entrenamiento sobre grandes volúmenes. La importancia relativa de cada variable se obtiene de forma intrínseca por *gain* (reducción media del error que aportan los *splits* sobre esa variable) o por *permutation importance*. Esta familia tolera datos heterogéneos, requiere poca normalización y entrena rápido.

Redes recurrentes profundas. Las redes recurrentes están diseñadas para procesar secuencias: en cada paso temporal t reciben una entrada x_t y mantienen un estado interno h_t que resume lo que han visto hasta ese momento. El esquema básico de procesamiento es secuencial, desde el primer paso hasta el último, y la predicción se construye a partir del estado oculto final:

$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow x_t \rightarrow h_t \rightarrow clasificador \tag{6}$$

La red recurrente básica se entrena con dificultad sobre secuencias largas porque el gradiente, al propagarse hacia atrás muchos pasos, tiende a desvanecerse o a explotar. La red *LSTM* [19] resuelve esa limitación introduciendo una celda de memoria y tres compuertas que deciden, en cada paso, qué información del pasado conservar, qué información nueva incorporar y qué información dejar pasar como salida. Estas compuertas son funciones sigmoides que producen valores entre 0 y 1, actuando como interruptores suaves; combinadas, regulan el flujo de información de modo que la red puede mantener dependencias temporales largas sin que el gradiente se degrade. La GRU es una variante más ligera con dos compuertas en lugar de tres.

Sobre estas arquitecturas se monta el mecanismo de atención [20], que permite a la red ponderar dinámicamente cada paso pasado de la secuencia según su relevancia para la predicción. En lugar de comprimir toda la información en el último estado oculto h_t, la atención asigna un peso normalizado α_{tj} a cada paso *j*, obtenido como *softmax* de una puntuación de afinidad e_{tj}:

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_k \exp(e_{tk})} \tag{7}$$

Los pesos suman uno y forman una distribución sobre la secuencia. El vector de contexto que alimenta la predicción se construye como suma ponderada de los estados ocultos:

$$c_t = \Sigma\_j\, \alpha_{tj} \cdot h_t \tag{8}$$

de modo que los pasos más informativos reciben pesos altos y dominan la salida. Esta

ponderación es además interpretable, ya que los α_{tj} indican explícitamente en qué parte de la secuencia se concentra el modelo para cada predicción.

Las arquitecturas *sequence-to-sequence* [21] articulan un codificador, que comprime la secuencia de entrada, y un decodificador, que genera la secuencia de salida, opcionalmente con atención sobre la salida del codificador. A diferencia de *SVM* y árboles, estas redes manejan la dimensión temporal de forma nativa y no requieren ingeniería explícita de rezagos, a cambio de mayores volúmenes de datos y menor transparencia.

Modelos de estados y transición. Esta familia describe la operación del corredor como una secuencia de estados discretos, por ejemplo, fluido, irregular o *bunched,* y se concentra en modelar cómo se pasa de un estado a otro con el paso del tiempo. La hipótesis simplificadora central es la propiedad de Markov: el estado siguiente depende solo del estado actual, no de la historia completa. Es decir:

$$P(S_{t+1} = j \mid S_t = i,\ S_{t-1}, \dots, S_1) = P(S_{t+1} = j \mid S_t = i) = p_{ij} \tag{9}$$

de modo que el sistema queda determinado por la matriz de transición *P = [* p_{ij}*]*, que recoge para cada par de estados la probabilidad de pasar de uno al otro. Una extensión útil incorpora variables explicativas *x* (*hora del día, ocupación, velocidad)* al predecir el estado siguiente mediante un modelo logístico multinomial:

$$P(S_{t+1} = k \mid x) = \frac{\exp(\beta_k^\top x)}{\sum_{j=1}^{K} \exp\left(\beta_j^\top x\right)} \tag{10}$$

Cada estado k tiene su propio vector de coeficientes β_k, y la fórmula transforma esos productos en probabilidades que suman uno. La calidad del modelo depende fuertemente de la discretización inicial (habitualmente obtenida por *clustering* sobre atributos operativos como velocidad, ocupación y *headway*) y del número de estados elegido. Su ventaja principal es la interpretabilidad: las probabilidades de transición y los coeficientes β_k ofrecen una lectura directa del comportamiento del sistema [22].

Marcos de aprendizaje en línea. A diferencia de las familias anteriores, que asumen un conjunto de entrenamiento fijo y un período de despliegue posterior, los marcos de aprendizaje en línea actualizan los parámetros del modelo con cada nueva observación. La regla genérica es el descenso por gradiente estocástico:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta L(f_\theta(x_t), y_t) \tag{11}$$

donde θ son los parámetros, n_t la tasa de aprendizaje y *L* la función de pérdida que cuantifica el error en la observación actual. Una implementación común para modelos lineales es el *Recursive*

Least Squares (RLS) con factor de olvido, que pondera las observaciones de modo que las recientes pesen más que las antiguas, lo que permite al estimador seguir derivas lentas de la distribución. Variantes equivalentes existen sobre redes neuronales (reglas delta, retropropagación en línea), árboles incrementales (*Hoeffding Trees*) y ensambles con reemplazo gradual.

Lo distintivo de esta familia no es el modelo base sino la política de actualización y su integración con un circuito de decisión que produce alertas y acciones. Es apropiada para flujos de datos en tiempo real y para entornos con *concept drift*, donde la distribución de los datos varía a lo largo del tiempo.

La elección entre familias no es excluyente: los modelos secuenciales nativos absorben la secuencia sin ingeniería de características pero requieren mayor volumen de datos y son menos interpretables; los ensambles de árboles necesitan construir rezagos explícitos pero exponen directamente la importancia de cada variable y entrenan más rápido [12], [23]; los modelos de estados ofrecen diagnóstico estructural al costo de menor exactitud; los marcos en línea sacrifican poder estadístico por adaptabilidad. Un criterio pragmático frecuente es comparar al menos un representante de cada familia sobre el mismo corpus con particiones temporales equivalentes.

Métricas de evaluación del modelo

Las métricas estándar para evaluar un modelo de predicción de *bus bunching* se agrupan en dos familias. Las métricas estadísticas, calculadas sobre datos de prueba fuera de muestra, evalúan la calidad de la predicción: *accuracy, precision, recall* y *F1* cuando se clasifica el evento, y MAE o RMSE cuando se predice el *headway* como variable continua [5], [13], [6]. Las métricas operativas, orientadas al uso del modelo como alerta, evalúan la anticipación y la tasa de aciertos por horizonte. La Tabla 3 sintetiza las métricas más empleadas.

Tabla 3. Métricas de evaluación del modelo

Categoría	**Métrica**	**Interpretación**	**Etapa de uso**
Clasificación	*Accuracy*, *Precision*, *Recall*, *F1*, ROC-AUC	Calidad global de la alerta (aciertos frente a falsas alarmas)	Desarrollo y comparación de modelos
Error continuo	MAE, RMSE del *headway* pronosticado	Exactitud al predecir la magnitud del intervalo	Desarrollo y comparación de modelos
Anticipación	Sensibilidad a *n* paradas o a *t* minutos	¿Cuántos eventos se detectan con antelación suficiente?	Desarrollo y comparación de modelos
Interpretabilidad	*SHAP*, *gain*, *permutation importance*	Importancia relativa de cada variable	Análisis del modelo seleccionado

Fuente: Elaboración propia con base en [5], [12], [13], [6] , [24].

Lead-time y horizonte de predicción

El *lead-time* es la anticipación con que el modelo emite la alerta antes de que el evento de agrupamiento se materialice. Se expresa en dos unidades según cómo se ancle la predicción: en número de paradas adelante (*n-step ahead*) cuando se ancla en el recorrido del bus, y en minutos cuando se ancla en el reloj operativo. La primera unidad es la más habitual en la literatura porque corresponde con la estructura natural del dato, cada parada genera una observación de *headway* y *dwell time*, y porque el horizonte espacial define qué acciones correctivas son viables: una alerta con tres paradas de antelación habilita retención o ajuste de velocidad, mientras que una alerta con diez paradas permite reasignación de despachos o *skip-stop* planificado.

Operativamente, el *lead-time* es la métrica que determina la utilidad práctica del sistema. Una alerta emitida menos de un minuto antes del evento no deja margen para ejecutar la acción correctiva; una alerta emitida demasiado pronto se basa en información poco informativa y eleva la tasa de falsa alarma. Existe, por tanto, un horizonte útil que debe calibrarse empíricamente para cada corredor y que compromete exactitud a cambio de anticipación.

Para TransMilenio, la calibración del horizonte debe considerar al menos tres factores específicos: el tiempo típico entre estaciones, que en el componente troncal varía entre uno y tres minutos en hora pico, lo que traduce un horizonte de cinco paradas a aproximadamente cinco a quince minutos de anticipación, el catálogo de acciones disponibles para el controlador (retención, *skip-stop*, reasignación, ajuste de velocidad), cada una con requerimientos de anticipación distintos, y la tolerancia operativa a la falsa alarma, que limita el umbral de decisión y, por vía indirecta, el horizonte en que la predicción aún aporta valor. La práctica habitual es presentar el desempeño del modelo como una curva de sensibilidad frente a horizonte, lo que permite al operador fijar el punto de corte en función de la acción correctiva y de su costo operativo.

Desbalance de clases

El *bus bunching* es un evento de baja prevalencia individual a nivel de cada arribo. En los estudios revisados, la clase positiva representa entre el 5 % y el 15 % de las observaciones por arribo [6]; esta baja prevalencia por evento no es incompatible con que el fenómeno tenga impacto sostenido sobre la calidad del servicio percibida por los usuarios, dado que afecta de manera concentrada a corredores y franjas horarias con alta demanda. Un clasificador que prediga siempre "no *bunching*" alcanzaría exactitudes altas, pero sería inútil como alerta, de ahí que la exactitud global no sea adecuada. En este contexto los dos tipos de error tienen costo operativo: omitir un evento real deja al centro de control sin oportunidad de corregirlo, y emitir una falsa alarma activa intervenciones (retención, salto de parada, regulación de velocidad) que afectan al servicio sin necesidad. *F1*, al ser la media armónica de precisión y *recall*, penaliza simultáneamente ambos errores y por eso se adopta como métrica preferida sobre la clase minoritaria.

Las estrategias documentadas para atender el desbalance incluyen el remuestreo sintético (SMOTE y variantes), las funciones de pérdida ponderadas (*Focal Loss*, *class-weighted cross-entropy*) y la calibración del umbral de decisión sobre la curva ROC o *precision-recall* [13], [6]. Sun *et al.* [13] subrayan que, en presencia de desbalance, la elección del umbral es un parámetro operativo más que estadístico: debe fijarse en función del costo relativo entre atender una alerta falsa y perder un evento real.

Interpretabilidad

En un sistema de apoyo a decisiones operativas, la alerta debe ir acompañada de una explicación que permita al controlador evaluar su pertinencia y elegir la acción correctiva apropiada. La literatura distingue dos enfoques complementarios: interpretabilidad *intrínseca*, cuando el modelo expone su estructura de decisión por construcción, e interpretabilidad *post-hoc*, cuando las explicaciones se generan después de entrenar un modelo de caja negra. Una presentación sistemática de los métodos que se describen a continuación (tanto intrínsecos como post-hoc) se encuentra en el libro abierto de Molnar [25].

Los modelos intrínsecamente interpretables (árboles de decisión individuales, modelos logísticos, cadenas de Markov sobre estados discretizados [22]) muestran directamente la regla de decisión y la contribución de cada variable. En ensambles de árboles, la interpretabilidad intrínseca se extiende a través de dos medidas estándar: el *gain* (reducción promedio de impureza aportada por cada variable en los *splits*) y la *permutation importance* (degradación del desempeño al permutar aleatoriamente una variable de entrada). Ambas entregan un ordenamiento global de variables, útil para diagnóstico y para selección de atributos.

Para modelos de caja negra (redes recurrentes, ensambles profundos, *SVM* con kernels no lineales) la literatura emplea métodos post-hoc. Los valores *SHAP* (*SHAPley Additive exPlanations*), fundamentados en teoría de juegos cooperativos, atribuyen a cada variable su contribución marginal a la predicción de una instancia específica y permiten construir explicaciones tanto locales (¿por qué este bus recibió una alerta?) como globales (¿qué factores pesan sistemáticamente en las alertas del corredor?). Los gráficos de dependencia parcial (*Partial Dependence Plots*) complementan el análisis mostrando el efecto marginal promedio de cada variable sobre la predicción.

La interpretabilidad cumple tres funciones transversales en este tipo de sistemas: *operativa*, pues justifica la alerta ante el controlador y guía la elección de la acción correctiva; *de diagnóstico*, pues permite identificar correlaciones espurias y sesgos del modelo antes del despliegue; y *de auditoría*, pues facilita documentar el comportamiento del modelo ante entidades de regulación y ante cambios de programación. Para TransMilenio, esta dimensión es especialmente relevante porque la acción correctiva se ejerce sobre un servicio público de alta visibilidad y porque los insumos de datos (AFC, GPS, GTFS) son gestionados por distintas áreas que deben comprender las señales que el modelo utiliza.

Validación temporal y aprendizaje incremental

La validación por partición aleatoria no es apropiada para series de tiempo, porque introduce fuga de información del futuro hacia el entrenamiento. La literatura revisada emplea partición temporal estricta; primer tramo del periodo para entrenamiento, último tramo para validación y, cuando el periodo es largo, validación por ventana deslizante (*rolling window*) que simula el despliegue del modelo en producción [4], [12]. Los patrones de demanda y operación cambian entre semanas (*concept drift*), de modo que un modelo entrenado con datos de hace seis meses puede degradarse si no se actualiza. El aprendizaje incremental (reentrenamiento por lotes o en línea sobre los datos más recientes) aparece en varios trabajos como estrategia para mantener la calidad de la predicción a lo largo del tiempo [4], [12]. Este aspecto es particularmente relevante para TransMilenio, donde eventos coyunturales (cierres de troncales, paros, cambios de programación) alteran la operación con frecuencia.

3.1.4 Estrategia de almacenamiento y procesamiento de datos

El análisis del fenómeno de *bus bunching* exige procesar volúmenes considerables de datos operativos provenientes de los sistemas AVL/GPS y de validación de pasajeros, los cuales generan millones de eventos diarios con alta granularidad temporal. Bajo estas condiciones, la elección de la estrategia de almacenamiento y de las herramientas de procesamiento condiciona de manera directa la viabilidad del estudio, su trazabilidad y la capacidad de reproducir resultados. La presente sección describe el enfoque adoptado, articulado en dos componentes complementarios: un patrón arquitectónico para organizar los datos a lo largo de su ciclo de vida, y un motor de procesamiento analítico que permite ejecutar consultas eficientes sobre grandes volúmenes desde un entorno de análisis científico en *Python*.

Organización del flujo de datos: arquitectura medallón

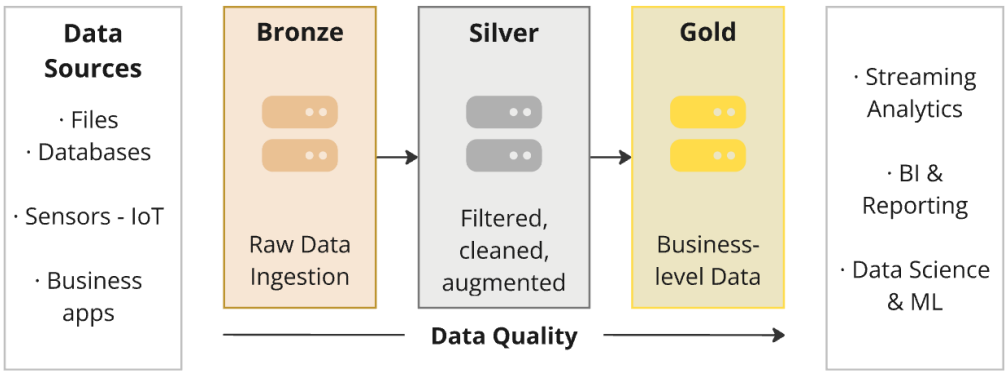
Para estructurar el ciclo de vida de los datos, desde su ingesta hasta su consumo por los modelos analíticos, se adoptó la denominada arquitectura medallón (*medallion architecture*), un patrón de diseño popularizado por *Databricks* en el contexto del paradigma *data lakehouse* [26], [27]. Este enfoque organiza la información en capas progresivas de refinamiento, cada una con un propósito definido y un nivel creciente de calidad y valor analítico, lo que resulta particularmente adecuado para proyectos de análisis de datos y *machine learning* que requieren preservar la trazabilidad desde la fuente cruda hasta los activos finales utilizados en el modelado.

La arquitectura define tres capas claramente diferenciadas. La capa *Bronze* corresponde a la ingesta y persistencia de los datos en su estado original, sin transformaciones semánticas, preservando la integridad y la procedencia de la información como fuente única de verdad; en este nivel se prioriza la inmutabilidad y el uso de esquemas de particionamiento que faciliten la persistencia y la consulta básica [27]. La capa Silver realiza la depuración, normalización e integración de los datos: aquí se corrigen inconsistencias estructurales y semánticas, se gestionan valores atípicos o inválidos y se aplican reglas de calidad y de negocio que permiten disponer de

información confiable para análisis exploratorios y procesos analíticos intermedios. Finalmente, la capa Gold concentra los datos preparados para el consumo final, donde se realizan agregaciones y transformaciones orientadas a casos de uso específicos (indicadores operativos, conjuntos de entrenamiento o tableros de seguimiento), optimizando la estructura para su explotación eficiente y garantizando alineación con los objetivos del análisis.

La Figura 3 ilustra el flujo unidireccional entre las tres capas y la progresión de calidad de los datos a medida que avanzan por el *pipeline*.

Figura 3. *Pipeline* arquitectura medallón



Fuente: Adaptado de [27].

La adopción de este patrón aporta varios beneficios al estudio. En primer lugar, el flujo unidireccional entre capas facilita la auditoría de las transformaciones aplicadas y permite regenerar capas derivadas a partir de los datos originales, lo que favorece la detección y corrección de errores sin pérdida de información. En segundo lugar, la separación de responsabilidades reduce el acoplamiento entre procesos y permite que cada capa incorpore controles y validaciones acordes con su nivel de madurez. Por último, el procesamiento progresivamente más depurado mejora el desempeño de almacenamiento y consulta a medida que los datos avanzan por el *pipeline*, y permite incorporar nuevos volúmenes o ampliar el horizonte temporal sin alterar la lógica de consumo final [27].

Motor de procesamiento analítico: *DuckDB*

Para materializar las transformaciones definidas por la arquitectura medallón y ejecutar las consultas analíticas requeridas por el estudio se adoptó *DuckDB*, un motor de base de datos analítica embebido y orientado a columnas, diseñado específicamente para cargas de trabajo OLAP (*Online Analytical Processing*) sobre conjuntos de datos de mediana y gran escala [28]. A diferencia de los sistemas cliente-servidor tradicionales, *DuckDB* se ejecuta dentro del mismo proceso del lenguaje anfitrión (en este caso Python), lo que elimina la sobrecarga de comunicación por red y permite ejecutar consultas SQL directamente sobre archivos en formatos como Parquet, *CSV* o *Arrow*, sin necesidad de cargar previamente toda la información en memoria [28].

La elección de *DuckDB* responde a las limitaciones que presentan herramientas como Pandas y *Polars* cuando el volumen de datos se aproxima o supera la capacidad de memoria RAM disponible. Pandas, ampliamente difundida en el ecosistema científico de *Python* [29], opera exclusivamente en memoria y bajo un modelo de ejecución *eager*; al manipular tablas con decenas de millones de registros, presenta degradaciones de rendimiento y, con frecuencia, fallos por desbordamiento de memoria. *Polars* mitiga esta situación gracias a su motor implementado en Rust, su modelo de ejecución *lazy* y su paralelización nativa, alcanzando aceleraciones considerables respecto a Pandas en cargas analíticas; no obstante, sigue acotado por los recursos físicos de la máquina y su soporte SQL, aunque en expansión, no cubre la totalidad de las operaciones analíticas complejas que demanda el estudio (joins múltiples, funciones de ventana y agregaciones jerárquicas).

DuckDB resuelve estas limitaciones a partir de tres características de su diseño. La primera es la ejecución *out-of-core*, que permite procesar volúmenes de datos superiores a la memoria disponible mediante particionamiento automático y *spilling* a disco. La segunda es un optimizador de consultas vectorizado que aprovecha el procesamiento por lotes y la paralelización multi-hilo, alcanzando velocidades comparables o superiores a las de otros motores analíticos en operaciones típicas sobre datos columnares [28]. La tercera es la interoperabilidad directa con *DataFrames* de Pandas y Polars, así como con archivos *Parquet* y *Arrow,* lo que permite integrar *DuckDB* en flujos de trabajo de Python sin requerir migraciones complejas ni copias adicionales de los datos.

La Tabla 4 sintetiza las diferencias principales entre las tres herramientas consideradas, en relación con los aspectos relevantes para el procesamiento de los datos del estudio.

Tabla 4. Comparación entre *Pandas, Polars y DuckDB*

Característica	***Pandas***	***Polars***	***DuckDB***
Modelo de ejecución	*Eager*, en memoria	*Lazy,* vectorizado	Vectorizado, *out-of-core*
Paralelización	Limitada (un hilo)	Multi-hilo nativa	Multi-hilo nativa
Soporte SQL	No	Parcial	Completo (ANSI SQL)
Procesamiento de datos mayores que la memoria disponible	No soportado (falla por memoria)	API de *streaming* por lotes	*Spilling* automático a disco
Lectura directa de Parquet	Sí (vía *PyArrow*)	Sí, nativa	Sí, nativa con *pushdown*

Fuente: Elaboración propia.

De la comparación se observa que *DuckDB* combina, en una misma herramienta, la capacidad de procesamiento *out-of-core*, la paralelización nativa y el soporte completo de SQL analítico, características que resultan determinantes para el cálculo de *headways, dwell times* e indicadores de *bunching* directamente sobre registros operacionales de alta granularidad. En este contexto, la herramienta ofrece un balance adecuado entre potencia analítica, eficiencia computacional y simplicidad de integración con el entorno de análisis en *Python*, evitando etapas intermedias de transformación costosas y simplificando la implementación del pipeline asociado a la arquitectura medallón descrita previamente.

3.2 ANTECEDENTES

El fenómeno de *bus bunching* se ha estudiado desde perspectivas complementarias: modelado estadístico y caracterización descriptiva en contextos locales, y predicción supervisada con aprendizaje automático en contextos internacionales. A continuación, se destacan los antecedentes más relevantes.

3.2.1 TransMilenio, Bogotá

Peña y Moreno [10]. Midieron en campo los tiempos de permanencia en parada (*dwell time*) a lo largo de un tramo de aproximadamente 9,7 km de la Troncal Norte, con trece estaciones sencillas operadas por buses articulados y biarticulados, y ajustaron modelos de dwell time basados en el número de abordajes y descensos siguiendo la metodología del *Transit Capacity and Quality of Service Manual*. Encuentran que la variabilidad de abordajes y descensos es el principal predictor del *dwell time* y, por extensión, de la capacidad efectiva del corredor en hora pico. Aporte y diferencia*:* el estudio identifica variables locales clave, pero no genera alertas predictivas; el presente trabajo toma esas variables como insumo y avanza hacia modelos de predicción en línea.

Naranjo [9]. Analiza con datos masivos (75.541 registros de paso de buses por estaciones de la Troncal Calle 80) la saturación del sistema y la variabilidad del tiempo de servicio, identificando estaciones críticas como Avenida 68, con coeficientes de variación del tiempo de servicio superiores al 100 %. Muestra que las metodologías tradicionales de capacidad, basadas en frecuencias programadas, subestiman el riesgo real de apelotonamiento en estaciones con alta variabilidad. Aporte y diferencia*:* caracteriza descriptivamente el fenómeno y sus correlatos locales; el presente trabajo retoma su diagnóstico y avanza hacia la predicción anticipada con modelos de aprendizaje automático.

3.2.2 Estudios internacionales con redes recurrentes

Jiao et al. [6] - *LSTM con atención (Xiangyang, China).* Proponen una arquitectura *LSTM* de dos capas con mecanismo de atención, combinada con *Focal Loss* y SMOTE para atender el desbalance severo de la clase *bunching* (≈ 6,1 % de las observaciones). Entrenado con datos AFC y GPS de la Ruta 9 de Xiangyang (26 de octubre a 23 de diciembre de 2020), el modelo alcanza *Precision* y *Recall* de 0,89 en la clase *bunching*, superando al *LSTM* estándar, *SVM*, ANN y *Random Forest*. El

estudio de ablación identifica la velocidad del bus como atributo más crítico. *Aplicabilidad:* constituye un referente cuantitativo para comparar desempeño de modelos recurrentes en el presente trabajo y valida la combinación *Focal Loss* + SMOTE para clases minoritarias, situación equivalente a la de TransMilenio.

Gong et al. [11] - *SD-seq2seq (múltiples rutas, China).* Proponen un modelo *sequence-to-sequence* que integra explícitamente atributos de oferta (tipo de estación, *dwell time*) y de demanda (abordajes y descensos estimados desde AFC) para predecir el *headway* aguas abajo y, con él, la ocurrencia de *bus bunching*. La evaluación sobre múltiples rutas reales muestra mejoras frente a modelos base. *Aplicabilidad:* su diseño de atributos es directamente trasladable a TransMilenio, que cuenta con información equivalente de estaciones, *dwell time* y demanda por AFC.

3.2.3 Estudios internacionales con ensambles y modelos de soporte

Yu et al. [5] - *LS-SVM con datos de tarjeta inteligente (Beijing).* Demuestran que a partir de seis variables derivadas exclusivamente de AFC (*headway* del bus precedente, demanda de abordaje y descenso del bus objetivo y del siguiente, tiempo de viaje interestación) es posible predecir el *headway* y clasificar *bunching* una parada adelante. Sobre dos rutas de Beijing (julio–noviembre de 2012), *LS-SVM* identifica más del 95 % de los eventos de *bus bunching* y mantiene buena precisión al aumentar la anticipación. *Aplicabilidad:* es el estudio fundacional de la predicción de *bus bunching* basada en AFC y define un *benchmark* directamente replicable en TransMilenio, que dispone de fuentes de datos equivalentes.

Santos et al. [12] - *Ensamble de árboles (Curitiba y ciudad A, Brasil).* Proponen un ensamble por votación de Random Forest, XGBoost y CatBoost, con aprendizaje incremental por lotes, para predicción *multistep* hasta diez paradas adelante. Integran datos GPS, GTFS, clima y tráfico sobre dos ciudades brasileñas (Curitiba, 31 días, 219 rutas, 5,6 % de *bunching*; y una segunda ciudad, 12 días, 106 rutas, 13,5 % de *bunching*). El modelo alcanza una eficacia global entre 74 % y 80 % y sostiene un *F-measure* del orden de 74 % hasta diez paradas adelante, superando a líneas base lineales (regresión lineal, logística, *SVM* y RVM). *Aplicabilidad:* Curitiba es, entre los casos de estudio revisados, el más cercano a TransMilenio en arquitectura BRT y densidad de demanda; el trabajo valida la viabilidad del *lead-time* extendido con ensambles y el aprendizaje incremental es especialmente pertinente para TransMilenio, donde los patrones varían entre semanas.

Yang et al. [30] - *SVM (Changzhou, China).* Aplican distintas variantes de *SVM* sobre datos AFC para detectar *bus bunching* a partir de la irregularidad de *headway* a nivel de parada y comparan su desempeño en una aplicación real. *Aplicabilidad:* aporta evidencia sobre el comportamiento de *SVM* como línea base frente a arquitecturas más complejas.

3.2.4 Estudios sobre modelos de estados y control predictivo

Deng et al. [22] - *Cadenas de Markov de estados (Xi'an, China).* Clasifican las trayectorias GPS en cinco estados de operación mediante *K-means*, modelan su evolución dinámica con una cadena de Markov y ajustan un modelo logístico multinomial para predecir el estado siguiente. La

validación sobre cuatro rutas urbanas de Xi'an reporta exactitudes de predicción por estación entre 80 % y 86 %, lo que evidencia que el enfoque captura adecuadamente las transiciones entre estados y permite anticipar condiciones cercanas al agrupamiento. *Aplicabilidad:* provee una alternativa interpretable y basada en estados operativos que puede complementar modelos de caja negra en la fase de análisis del corredor.

Andres y Nair [14] - *Marco de control predictivo (Dublín, Irlanda).* Combinan predicción de *headway* con estrategias dinámicas de retención y validan el esquema sobre datos de una ruta ocupada en Dublín. Muestran que la combinación de predicción y retención reduce la desviación de *headways* frente a estrategias reactivas. *Aplicabilidad:* ofrece un esquema de referencia para integrar el modelo predictivo con acciones operativas, que en una etapa posterior puede evaluarse en el centro de control de TransMilenio.

Moreira-Matias et al. [4] - *Aprendizaje en línea (Oporto, Portugal).* Presentan un marco de aprendizaje en línea que combina clasificadores incrementales y reglas operativas para emitir alertas y activar acciones correctivas en tiempo real. Validan el enfoque sobre datos reales de Oporto y reportan reducciones en la incidencia de *bus bunching*. *Aplicabilidad:* constituye el antecedente metodológico más cercano al objetivo final del presente trabajo, pues articula predicción, alerta y decisión operativa.

3.2.5 Estudios sobre diseño de la evaluación y predicción de tiempos de viaje

Sun et al. [13] - *Compromiso sensibilidad/especificidad.* Analizan cómo la elección de umbrales de decisión afecta el balance entre alertas verdaderas y falsas en predicción de *bus bunching* y proponen criterios para calibrar el clasificador según el costo operativo de cada tipo de error. *Aplicabilidad:* su marco de evaluación orienta la selección del umbral operativo del modelo que se desarrollará, más allá de métricas agregadas como la exactitud.

Reddy et al. [31] - *Predicción de tiempos de viaje bajo alta variabilidad (Chennai, India).* Aplican ε-SVR con kernel lineal sobre datos GPS de buses en tráfico heterogéneo y muestran el valor de incorporar la varianza en la formulación del predictor. *Aplicabilidad:* relevante como referencia para predictores continuos en corredores con alta variabilidad, condición parcialmente compartida por TransMilenio.

Kakarla et al. [23] - *Ensambles sobre GPS disperso en BRT (India).* Evalúan Random Forest, *LightGBM* y *XGBoost* en un BRT con datos GPS dispersos y encuentran que *XGBoost* reduce sustancialmente el error respecto a métodos convencionales. *Aplicabilidad:* aporta evidencia sobre la robustez de ensambles en condiciones de muestreo irregular, similar a la que puede presentarse en tramos del sistema TransMilenio.

Matseliukh et al. [24] - *Aprendizaje automático interpretable para retrasos de transporte.* Combinan modelos de aprendizaje automático con *SHAP* para analizar retrasos en transporte público e identificar los factores sistemáticos asociados. *Aplicabilidad:* metodología de referencia

para la dimensión de interpretabilidad del modelo final.

3.2.6 Síntesis transversal de la literatura

Más allá de los aportes individuales descritos antes, la literatura revisada permite identificar tres conjuntos de hallazgos transversales que orientan las decisiones metodológicas del presente trabajo: las variables comúnmente reportadas como predictores, la evidencia disponible sobre el compromiso entre anticipación y precisión, y los aprendizajes sobre interpretabilidad de modelos en el dominio.

Variables comunes reportadas

Los modelos supervisados alcanzan sus mejores desempeños cuando, además de los headways, incorporan información operativa, de demanda y de contexto. El conjunto mínimo de variables explicativas que reporta la literatura revisada incluye:

- *Headways* históricos y diferencia frente al intervalo programado [5], [6].
- *Dwell time* y conteo de abordajes y descensos por estación [10], [11].
- Factor de ocupación estimado a partir de AFC o telemetría [5], [13].
- Velocidad instantánea y tiempo de viaje entre estaciones [23].
- Factores exógenos: hora del día, día-tipo, condiciones meteorológicas y estado del tráfico externo [12], [24].

Compromiso *lead-time* / precisión observado

La evidencia empírica disponible permite acotar el rango operativamente útil del horizonte de predicción. Yu et al. [5] muestran que *LS-SVM* sobre datos AFC mantiene, en la Ruta A de Beijing, una sensibilidad del 100 % a una parada adelante, de 98,4 % a tres paradas y de 73,8 % a cinco paradas, lo que indica una degradación apreciable a partir del cuarto o quinto paso. Santos et al. [12] reportan, para su ensamble de árboles con predicción multistep, que el modelo sostiene un F-measure próximo al 74 % hasta diez paradas adelante y una eficacia global entre 74 % y 80 %, evidencia de que los ensambles pueden extender el horizonte operativo a costa de una pérdida acotada de desempeño. Jiao et al. [6] validan su modelo *LSTM*-Attention sobre un horizonte de aproximadamente diez intervalos entre estaciones y documentan, mediante estudio de ablación, que la velocidad del bus es el atributo más informativo para sostener la anticipación. Estos tres puntos sugieren que el rango operativamente útil se ubica entre tres y diez paradas, con un sweet spot habitual entre cinco y ocho.

Hallazgos de interpretabilidad en el dominio

Los trabajos revisados aportan evidencia sobre las variables que dominan las predicciones en distintos modelos. Santos et al. [12] reportan la importancia por gain de su ensamble y destacan el *headway* actual, el tiempo de viaje y la ocupación como variables dominantes; Kakarla et al. [23] comparan la importancia de atributos entre *XGBoost, LightGBM* y *Random Forest* sobre datos

de BRT; y Matseliukh et al. [24] articulan un flujo completo de aprendizaje automático interpretable con *SHAP* para identificar factores sistemáticos de retraso en transporte público urbano. Estos hallazgos orientan tanto la selección de atributos para los modelos a desarrollar como la metodología de análisis de interpretabilidad propuesta.

4. INTEGRACIÓN Y PREPARACIÓN DE DATOS

En este capítulo se describe el proceso de recolección, consolidación y preparación de los datos operativos de TransMilenio. Se describen las fuentes de información empleadas, así como el procedimiento de extracción, carga e integración de estos datos en un *dataset* unificado. De igual forma, se expone la arquitectura de datos adoptada, y se detallan los criterios de alineación temporal y espacial, la construcción de la variable objetivo y las principales características del conjunto de datos resultante, que constituye la base analítica para las etapas posteriores de análisis exploratorio, preprocesamiento, y modelamiento predictivo.

Antes de describir las fuentes, es importante aclarar la unidad de análisis operativa del proyecto. Como se anticipó en el planteamiento del problema y en el marco teórico, las tres fuentes operativas de TransMilenio empleadas (validaciones, intervalos de servicios y apertura de puertas) registran los eventos a nivel de servicio y estación, pero ninguna incluye un identificador de viaje (*viaje_id*) ni un identificador de vehículo que permita seguir a un bus individual a lo largo de su recorrido. En consecuencia, la unidad de predicción del modelo desarrollado en los capítulos siguientes es el par (servicio, estación) ordenado cronológicamente, no el bus individual. Esta elección difiere de parte de la literatura clásica de *bus bunching*, que predice n paradas adelante del mismo bus a partir de información GPS por vehículo, y es una limitación que se asume desde la formulación del problema para alinear el modelo con la información efectivamente disponible en las fuentes oficiales del operador.

4.1 Fuentes de datos operativos

Se trabajó con tres fuentes de datos operativos de la red troncal de TransMilenio, complementadas con un diccionario geográfico de estaciones. Los datos provienen del Sistema de Información de Recaudo y Control Integrado (SIRCI) de TransMilenio [32], fuente primaria de información operacional del sistema BRT de Bogotá. TransMilenio S.A. suministró los *datasets* en formato plano, los cuales cubren el período comprendido entre el 1 de enero de 2022 y el 31 de diciembre de 2025, lo que proporciona cuatro años completos de información.

4.1.1 Intervalos de servicio

La información de intervalos constituye la fuente principal para el cálculo de la variable objetivo y para el *feature engineering*. La información se registra con una resolución a nivel de evento, capturando cada arribo individual de los vehículos a las estaciones de parada de su ruta programada. Para cada evento, se extraen el intervalo teórico programado y el intervalo real observado, junto con atributos dimensionales como el identificador del servicio, la estación y las marcas de tiempo (teóricas y reales) de llegada. A partir de estos campos, se deriva el *headway ratio* (cociente entre el intervalo real y el teórico), una métrica técnica fundamental para cuantificar la desviación operativa y evaluar el nivel de cumplimiento respecto a la programación establecida por el sistema.

4.1.2 Apertura de puertas

Esta información detalla el tiempo que el bus permanece con las puertas abiertas en plataforma, lo que permite determinar el *dwell time* o tiempo de permanencia. Cada registro se encuentra asociado a un servicio específico, una estación y una marca de tiempo, permitiendo una resolución a nivel de evento. Esta información permite identificar cuellos de botella operativos, integrándose con la fuente de intervalos para identificar el fenómeno de agrupamiento de buses en puntos críticos de la red.

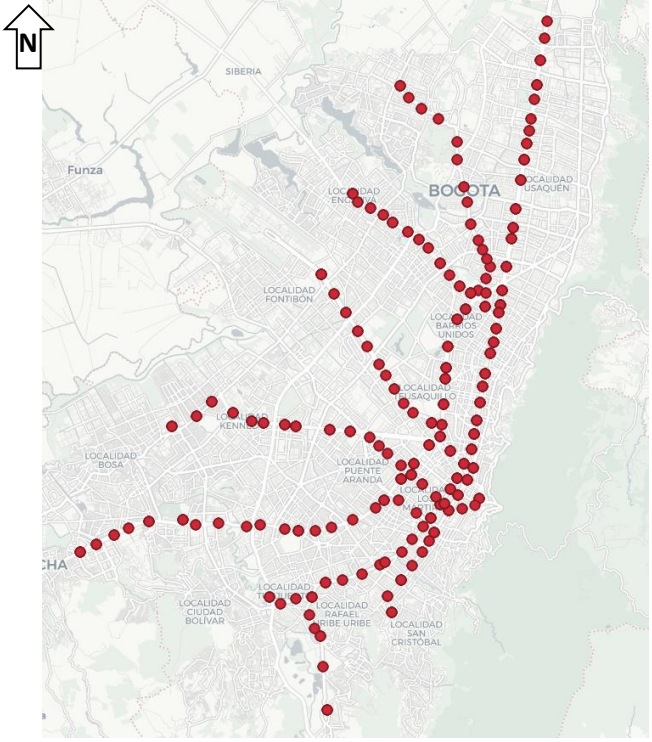
4.1.3 Validaciones

La información de validaciones presenta el número de abordajes de pasajeros por estación en intervalos de cinco minutos. Es importante precisar que la información a nivel de estación corresponde a un agregado que incluye todas las plataformas de acceso y la totalidad de rutas que operan en dicha estación. Esta fuente permite cuantificar la demanda de pasajeros con resolución temporal de cinco minutos y resolución espacial a nivel de estación.

4.1.4 Diccionario geográfico de estaciones

Se construyó un archivo de referencia con 161 estaciones de la red troncal (ver Figura 4), que incluye el identificador del polígono (*id_poligono*), el nombre de la estación, la zona geográfica a la que pertenece y los identificadores utilizados en los sistemas de validaciones y de intervalos. Este diccionario se elaboró con el propósito de vincular las diferentes fuentes de información disponibles, dado que los códigos de identificación de estaciones no son homogéneos entre los distintos sistemas. Para su construcción, se partió de capas geográficas disponibles en los datos abiertos de la ciudad de Bogotá [33], las cuales sirvieron como base para consolidar un identificador espacial unificado y establecer la correspondencia entre las distintas codificaciones operativas. De esta manera, el diccionario actúa como una tabla de mapeo que permite homologar los códigos de estación entre las tres fuentes de datos y el identificador unificado *id_estacion*, facilitando así el proceso de integración mediante cruces relacionales.

Figura 4. Estaciones componente troncal sistema TransMilenio



Fuente: Elaboración propia

La siguiente tabla resume las características principales de cada fuente de datos.

Tabla 5. Características principales de las fuentes de datos

Fuente	Resolución Temporal	Volumetría	Variables principales
Intervalos	Nivel Evento	269.238.286	intervalo_teorico_s, intervalo_real_s (tiempo de intervalos)
Tiempo apertura puertas	Nivel Evento	330.566.164	apertura_puertas_s (tiempo de apertura de puertas)
Validaciones	Agregado (5 min)	47.595.389	validaciones (conteo de abordajes)
Estaciones	Referencial	161	nombre, zona, ids SAE

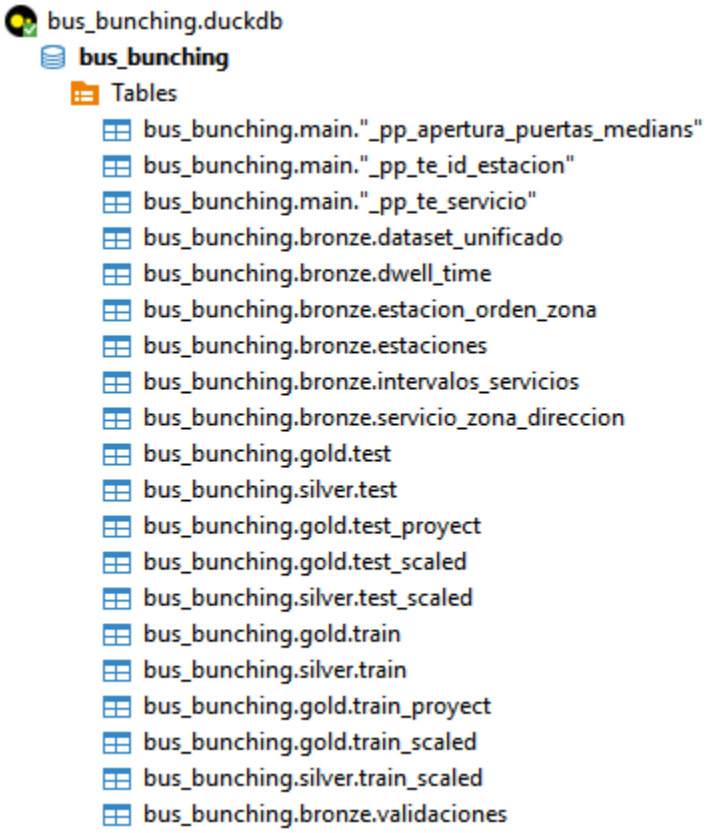
Fuente: Elaboración propia

4.2 Arquitectura de datos

El flujo de datos sigue una arquitectura por capas inspirada en el patrón medallón, utilizado en ingeniería de datos para garantizar la trazabilidad entre datos crudos y datos transformados. Los archivos CSV originales se almacenaron y se cargaron en *DuckDB*, un motor analítico columnar embebido que ejecuta consultas SQL sobre datos locales sin requerir infraestructura distribuida. La base analítica del proyecto organiza la información en tres esquemas: *bronze, silver* y *gold*.

Las tres capas implementan un flujo progresivo en el que cada nivel parte de la salida del anterior (ver Figura 5). La capa *bronze* concentra la carga limpia y la integración entre fuentes; la capa *silver* aplica el preprocesamiento sobre la partición temporal de entrenamiento y prueba; la capa *gold* añade las variables derivadas resultantes del *feature engineering*. Esta separación preserva la salida de cada etapa y permite modificarla sin alterar las anteriores. Las subsecciones siguientes describen el contenido de cada capa.

Figura 5. Tablas de base de datos analítica *DuckDB*



Fuente: Elaboración propia

4.2.1 Capa *bronze*

La capa *bronze* almacena la carga limpia desde los archivos crudos, sin aplicar transformaciones analíticas. Cada fuente operativa se carga en su propia tabla (validaciones, tiempos de apertura de puertas e intervalos de servicios) junto con el diccionario de estaciones. En esta capa se realiza la homologación de identificadores de estación mediante el diccionario geográfico y la alineación temporal de las tres fuentes a una resolución común de cinco minutos. El resultado de la integración se almacena como el *dataset* unificado de la capa *bronze*.

4.2.2 Capa *silver*

La capa *silver* contiene los datos después de la partición temporal de entrenamiento y prueba y de las transformaciones de preprocesamiento. Estas transformaciones cubren cuatro operaciones puntuales sobre los registros del *dataset* unificado: imputación de valores faltantes, codificación de variables categóricas de alta cardinalidad mediante *target encoding* suavizado, *winsorización* en el percentil 99 para acotar valores extremos en variables de cola pesada, y transformación logarítmica más escalado para los modelos sensibles a la distribución de los datos. Todas se ajustan exclusivamente sobre el conjunto de entrenamiento y se aplican sin reajustar sobre el de prueba, garantizando que ningún parámetro contenga información del periodo posterior al corte temporal. La capa incluye cuatro conjuntos: entrenamiento y prueba sin escalado para el Pipeline A (modelos de árboles), y entrenamiento y prueba con escalado para el Pipeline B (modelos lineales y redes neuronales).

La justificación de esta separación en dos *pipelines* se presenta en el Capítulo 5, en la sección de preprocesamiento, donde se detallan las transformaciones aplicadas y su relación con los distintos tipos de modelos evaluados.

4.2.3 Capa *gold*

La capa *gold* contiene los conjuntos finales listos para el modelamiento, con las variables derivadas resultantes del *feature engineering*: variables de rezago (*lags*), promedios y desviaciones móviles (*rolling*), derivadas (*deltas*), acumuladas y variables de interacción. Los conjuntos sin escalar alimentan el Pipeline A y los conjuntos escalados alimentan el Pipeline B.

La siguiente tabla resume la estructura de la base de datos.

Tabla 6. Estructura de la base de datos por capas

Esquema	Tabla	Descripción
bronze	*validaciones*	Carga limpia desde archivos crudos de recaudo
bronze	*dwell_time*	Carga limpia desde archivos crudos de apertura de puertas
bronze	*intervalos_servicios*	Carga limpia desde archivos crudos de regularidad
bronze	*estaciones*	Diccionario geográfico (161 estaciones)
bronze	*dataset_unificado*	JOIN de las tres fuentes
silver	*train / test*	Pipeline A: sin escalado
silver	*train_scaled / test_scaled*	Pipeline B: con escalado
gold	*train / test*	Pipeline A con variables derivadas
gold	*train_scaled / test_scaled*	Pipeline B con variables derivadas
main	*_pp_te_servicio/_pp_te_id_estacion/_pp_apertura_puertas_median*	Artefactos persistidos del preprocesamiento

Fuente: Elaboración propia

4.2.4 Implicaciones de la arquitectura adoptada

La adopción de *DuckDB* como motor analítico y de la arquitectura medallón para organizar los datos tiene consecuencias prácticas que resulta pertinente detallar, ya que condicionaron tanto el alcance del trabajo como las decisiones de implementación posteriores.

En el lado favorable, esta combinación permite procesar volúmenes de datos del orden de cientos de millones de filas del *dataset* analítico en un equipo personal, sin recurrir a infraestructura distribuida ni a *frameworks* como Spark. La organización por capas introduce puntos de control físicos entre fases: una falla en el *feature engineering* no obliga a recalcular el cruce inicial en la capa *bronze*, y una revisión del preprocesamiento no exige volver a procesar la ingesta. Esta separación facilita la re-ejecución selectiva del flujo durante el desarrollo iterativo y consolida una única fuente de verdad para el modelado, ya que los componentes de la fase de modelado consumen los conjuntos de entrenamiento y prueba de la capa *gold* sin riesgo de inconsistencias entre ellos. El uso de SQL declarativo con funciones de ventana, además, permite calcular los *lags* y las estadísticas *rolling* de la capa *gold* sobre el conjunto completo en tiempos compatibles con un flujo de trabajo iterativo.

Estas ventajas conllevan costos que se manifestaron en la propia estructura del proyecto. El más relevante es la ausencia de un objeto único de inferencia: las transformaciones de las capas *silver* y *gold* residen como consultas SQL parametrizadas mediante archivos de configuración, en lugar de un objeto serializado de *scikit-learn* que pueda invocarse sobre un registro individual. Esta misma decisión introduce, además, un acoplamiento al motor: las transformaciones están escritas en SQL y un cambio de motor implicaría reescribirlas. El esquema de depuración también se vuelve más complejo, ya que un comportamiento inesperado puede tener origen en cualquiera de las capas y exige inspeccionar las tablas intermedias.

En conjunto, la elección se considera adecuada para los objetivos del proyecto: el caso de uso es un análisis por lotes para un trabajo de investigación, el volumen de datos excluye un enfoque enteramente en memoria y la separación por capas hace posible reformular el *feature engineering* en varias ocasiones sin alterar las capas inferiores.

4.3 Integración del *dataset* unificado

El principal desafío de la integración radica en que las tres fuentes de datos utilizan identificadores de estación diferentes y tienen resoluciones temporales distintas. Para resolver esto, se adoptó una estrategia en dos pasos: primero la alineación de cada fuente a un formato común y luego el cruce sobre las claves unificadas.

4.3.1 Alineación temporal y espacial

La alineación temporal consistió en llevar las tres fuentes a una misma rejilla temporal con granularidad de cinco minutos, etiquetando cada registro con la franja de cinco minutos a la que

pertenece según su marca de tiempo (no se agregan registros entre sí: la operación es de asignación de etiqueta, no de agregación). Las validaciones y velocidades ya se encontraban en esta granularidad de forma nativa. Los intervalos de servicio, que registran cada arribo individual, se cruzaron uno a uno con la información de apertura de puertas, y posteriormente se etiquetaron con la franja de cinco minutos en la que se ubica el arribo (la franja que contiene la marca de tiempo, no la más cercana).

La alineación espacial se realizó mediante el diccionario de estaciones. Las validaciones y los intervalos utilizan códigos específicos de cada sistema que se mapearon al identificador unificado *id_poligono*. De las 161 estaciones del diccionario, 150 presentaron datos operativos efectivos en las tres fuentes durante el período de análisis.

4.3.2 Cruce y dataset resultante

Una vez alineadas las fuentes, se ejecutó el cruce sobre la clave compuesta formada por la marca de tiempo y el identificador de estación. La tabla de intervalos de servicio se utilizó como tabla base, dado que contiene el registro de cada arribo individual de un bus a una estación, que es la unidad de análisis del problema. Las validaciones se incorporaron como información contextual del lugar del arribo.

El *dataset* unificado resultante contiene 269.238.286 registros (volumetría del *dataset* de intervalos) que cubren 150 estaciones, 143 servicios troncales y 1.428 días de operación. La tabla siguiente presenta las variables principales del *dataset*.

Tabla 7. Variables principales del *dataset* unificado

Variable	**Tipo**	**Descripción**
datetime	*TIMESTAMP*	Fecha y hora del intervalo (5 min)
id_estacion	*INT64*	Identificador unificado de estación
servicio	*VARCHAR*	Identificador del servicio troncal
nombre_estacion	*VARCHAR*	Nombre de la estación troncal
zona	*VARCHAR*	Zona de localización de la estación
latitud	*FLOAT64*	Latitud del centroide de la estación (WGS84)
longitud	*FLOAT64*	Longitud del centroide de la estación (WGS84)
hora_teorica	*TIME*	Hora programada de llegada
hora_real	*TIME*	Hora real de llegada
intervalo_teorico_s	*INT64*	Intervalo programado (s)
intervalo_real_s	*INT64*	Intervalo observado (s)
validaciones	*INT64*	Conteo de registro de pasajeros en la franja de 5 min
apertura_puertas	*INT64*	Tiempo de apertura de puertas (s)

Fuente: Elaboración propia

4.4 Definición de la variable objetivo

El *bus bunching* se operacionaliza a partir del *headway ratio*, definido como el cociente entre el intervalo real observado y el intervalo teórico programado entre el arribo actual y el del bus inmediatamente anterior del mismo servicio en la misma estación. Un valor inferior a 1 indica que el bus llegó antes de lo programado respecto al despacho nominal, es decir, está acercándose al bus precedente del mismo servicio: en valores suficientemente bajos esa proximidad configura el agrupamiento que el modelo debe anticipar. Se definió la variable binaria *bunching_025*, que toma el valor 1 cuando el *headway ratio* es inferior a 0.25, es decir, cuando el intervalo real es menor al 25% del intervalo programado.

$$headway\_ratio_i = \frac{intervalo\ real_i}{intervalo\ teórico_i} \tag{12}$$

$$bunching\_025_i = \begin{cases} 1, & si\ headway\_ratio_i < 0.25 \\ 0, & si\ headway\_ratio_i \geq 0.25 \end{cases} \tag{13}$$

El uso del umbral de 0.25 (o 1/4) del intervalo programado se justifica en la literatura académica como el criterio más riguroso y estándar para identificar eventos reales de *bus bunching* (Ver sección 3.1.1). A diferencia de umbrales más amplios (como el 0.5), este valor asegura que se están detectando casos donde la irregularidad es crítica y la formación de pelotones es inminente o ya ha ocurrido. En la siguiente tabla se presenta la descripción de las dos variables que se calculan e integran en el *dataset* unificado.

Tabla 8. Variables calculadas del dataset unificado

Variable	Tipo	Descripción
headway_ratio	*FLOAT64*	Cociente entre el intervalo real y el intervalo teórico programado.
bunching_025	*INT64*	Variable binaria que indica la presencia (1) o ausencia (0) de bus *bunching*.

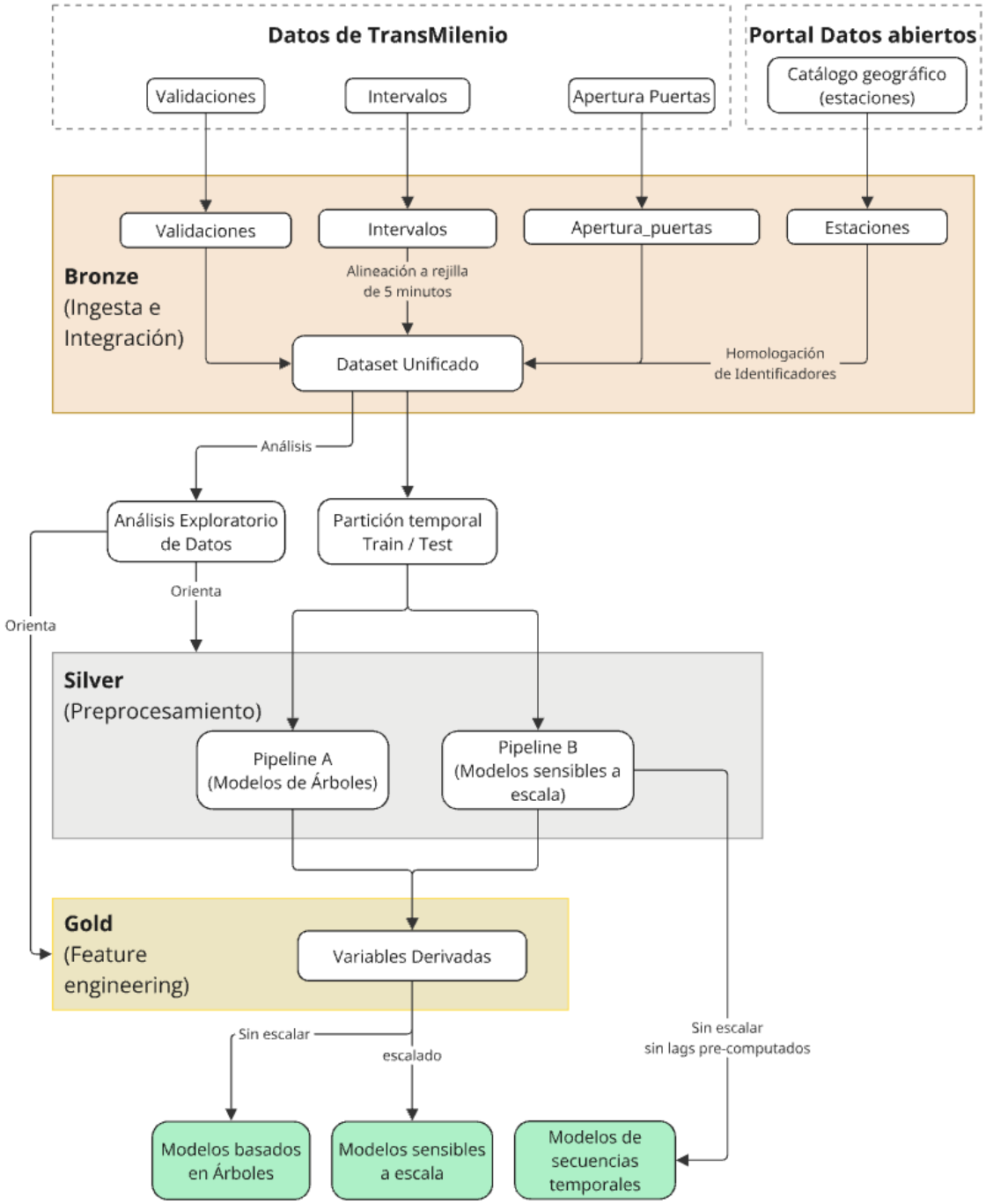
Fuente: Elaboración propia

4.5 Consideraciones finales del proceso de integración

La integración de las tres fuentes operativas en un único *dataset* unificado de aproximadamente 269 millones de registros establece la base para las etapas posteriores del proyecto. La adopción de una arquitectura por capas garantiza la trazabilidad entre los datos crudos y los conjuntos finales de modelamiento, mientras que la elección de *DuckDB* como motor analítico permite manejar el volumen de datos de forma eficiente sin recurrir a infraestructura distribuida. El capítulo siguiente presenta el análisis exploratorio realizado sobre este *dataset* y las transformaciones de preprocesamiento que preparan los datos para la fase de modelado.

Con el fin de sintetizar la arquitectura del proceso y la secuencia de transformación de los datos, el siguiente diagrama ilustra el flujo de datos desde las fuentes originales en archivos planos hasta el proceso de modelación, pasando por las capas *bronze, silver* y *gold*.

Figura 6. Flujo de datos e integración



Fuente: Elaboración propia

5. ANÁLISIS EXPLORATORIO Y PREPROCESAMIENTO

Este capítulo presenta los resultados del análisis exploratorio de datos (EDA) realizado sobre el *dataset* unificado descrito en el capítulo anterior, junto con las transformaciones de preprocesamiento y de *feature engineering* que materializan las capas *silver* (transformaciones puntuales: imputación, codificación, escalado opcional) y *gold* (variables derivadas con dependencia temporal: rezagos, móviles, derivadas, interacciones y variables agregadas del corredor) sobre la arquitectura medallón presentada en el capítulo de integración que preparan los datos para la etapa de modelamiento. Las decisiones de preprocesamiento se fundamentan en los hallazgos del EDA, de modo que cada resultado exploratorio se conecta con una acción concreta sobre los datos. Se describe además la estrategia de partición temporal, la imputación de valores faltantes, la codificación de variables categóricas, las transformaciones específicas por tipo de modelo y la construcción de variables derivadas en la capa *gold*.

5.1 Análisis Exploratorio de Datos

5.1.1 Objetivos y metodología de Análisis Exploratorio de Datos

El análisis exploratorio de datos se desarrolló con cuatro objetivos específicos:

- **Caracterizar el comportamiento de las variables principales:** describir la distribución, tendencia central, dispersión, asimetría y presencia de valores extremos en variables como intervalo teórico, intervalo real, *headway ratio*, validaciones y apertura de puertas.
- **Evaluar la calidad y completitud de los datos:** identificar valores nulos, revisar su magnitud e interpretar su origen operativo, con el fin de definir tratamientos de imputación consistentes con el problema de estudio.
- **Identificar patrones temporales y espaciales del *bunching*:** analizar la variación del fenómeno por hora del día, día de la semana, corredor, servicio, estación y zona, para reconocer contextos operativos asociados a mayor propensión al evento.
- **Orientar las decisiones de preprocesamiento y *feature engineering*:** sustentar, con base en los hallazgos del EDA, las transformaciones posteriores de imputación, exclusión de variables con riesgo de *leakage*, *target encoding*, winsorización, escalado y construcción de variables derivadas.

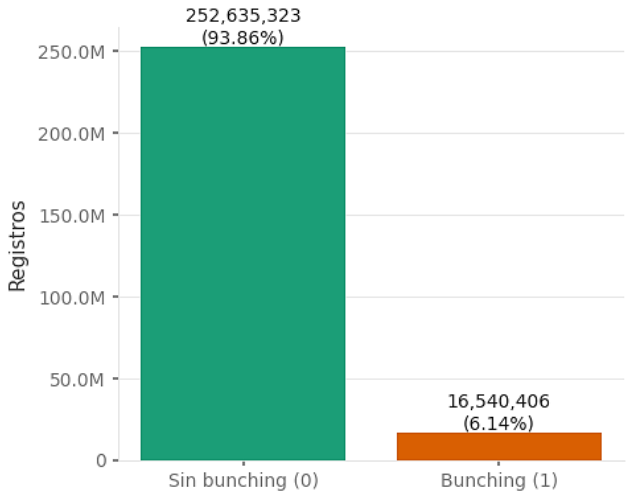
El EDA se estructuró como una secuencia de análisis aplicada sobre el *dataset* unificado, con énfasis en su utilidad para la etapa de modelamiento predictivo:

- **Caracterización inicial de la variable objetivo:** definición operativa del *bus bunching* a partir del *headway ratio* y revisión de la distribución de clases de la variable binaria *bunching_025*.
- **Análisis de calidad de datos:** cuantificación de valores nulos e interpretación de su significado operacional, junto con la definición de criterios de tratamiento para las variables afectadas.
- **Análisis univariado de variables principales:** cálculo de estadísticos descriptivos y visualización de distribuciones para identificar sesgos, colas pesadas y valores atípicos.
- **Análisis temporal:** estudio de patrones por hora del día, día de la semana y tendencia diaria, así como descomposición STL de las series agregadas para diferenciar componentes de tendencia, estacionalidad y residuo.
- **Análisis espacial y operativo:** exploración de la tasa de *bunching* por corredor, servicio y estación, con el fin de identificar heterogeneidad territorial y operacional en el fenómeno.
- **Análisis de asociación entre variables y *target*:** evaluación de relaciones entre las variables explicativas y *bunching_025* mediante métricas de asociación y análisis complementarios de correlación e interacción, con énfasis en su utilidad predictiva y en la detección de variables con riesgo de filtración de información.
- **Derivación de implicaciones para preprocesamiento:** traducción de los hallazgos del EDA en decisiones concretas para la construcción de las capas *silver* y *gold*, incluyendo imputación, codificación, *winsorización*, transformación, escalado y generación de *variables* temporales y de interacción.

5.1.2 Distribución de la variable objetivo

La distribución de la variable objetivo en el *dataset* completo de 269.238.286 registros muestra un marcado desbalance de clases: 93,86% corresponden a la clase 0 (sin *bunching*) y 6,14% a la clase 1 (*bunching*), lo que representa una proporción de aproximadamente 15,3 a 1. Este desbalance condiciona la elección de métricas de evaluación y las estrategias de entrenamiento adoptadas en la etapa de modelamiento. La siguiente figura presenta la distribución de clases en el *dataset*.

Figura 7. Distribución de la variable objetivo *bunching_025*



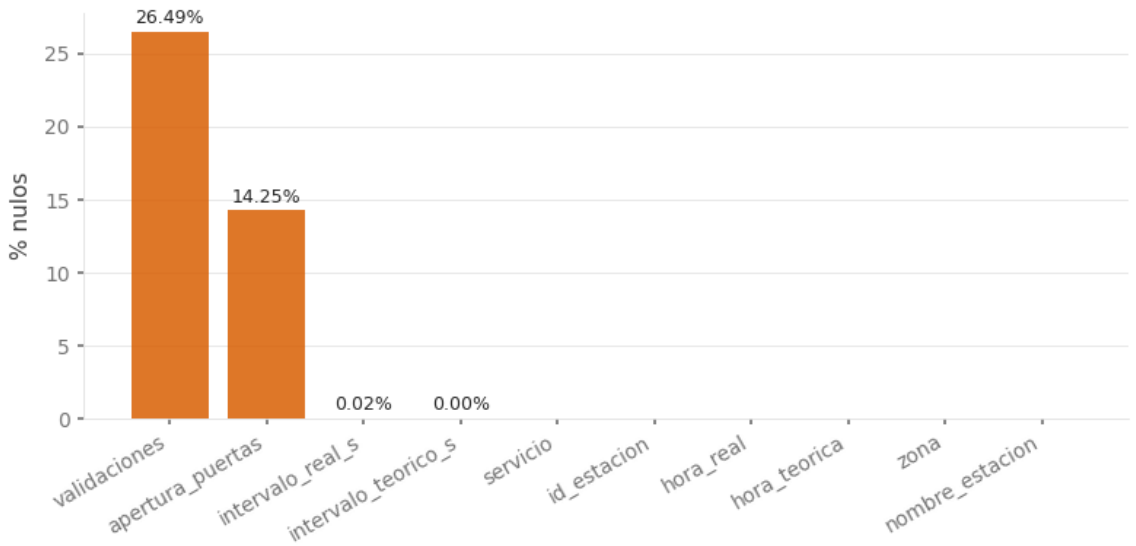
Fuente: Elaboración propia

5.1.3 Análisis de valores nulos

El *dataset* unificado presenta valores nulos en tres variables, con magnitudes muy distintas. La variable validaciones concentra la mayor proporción de datos faltantes, con un 26,49%. Esta tasa no constituye un problema de calidad de datos en sentido estricto, ya que refleja franjas horarias o estaciones en las que hubo servicio programado, pero no se registraron abordajes, lo cual puede deberse principalmente a periodos de baja demanda, interrupciones operativas o mantenimiento de estaciones.

La apertura de puertas presenta una tasa de nulos del 14,25%. El intervalo real tiene solo un 0,02% de valores ausentes. El porcentaje elevado de nulos de la apertura de puertas tiene dos orígenes posibles que conviene distinguir: por un lado, fallos del sensor embarcado que registra la duración de apertura, situación en la que el dato no llega a la base; por otro, paradas con demanda muy baja en las que no se abrieron las puertas (o se abrieron por un tiempo no registrado), lo que también se materializa como nulo. El intervalo real, en cambio, depende únicamente de las marcas de tiempo de arribo registradas por el sistema de programación, por lo que su tasa de nulos casi nula (0,02 %) corresponde a errores puntuales de captura. La siguiente figura muestra la proporción de nulos por variable.

Figura 8. Proporción de valores nulos por variable



Fuente: Elaboración propia

El tratamiento de los valores nulos se definió de la siguiente manera:

- las validaciones nulas se imputarán con el valor 0, bajo la interpretación de que la ausencia de registro equivale a la ausencia de abordajes en esa franja.
- La apertura de puertas se imputará con la mediana por combinación de servicio y estación, calculada exclusivamente sobre el conjunto de entrenamiento, con un valor de respaldo global para combinaciones sin datos históricos.
- Los registros con intervalo real nulo se eliminaron, y no se consideró su imputación dado su volumen despreciable y su importancia en el cálculo de la variable *headway ratio*.

5.1.4 Distribuciones de las variables principales

El análisis de las distribuciones de las variables continuas revela características relevantes para las decisiones de preprocesamiento. La tabla siguiente resume los estadísticos descriptivos.

Tabla 9. Estadísticos descriptivos de las variables continuas principales

Variable	Media	Desv. estándar	Q1	Mediana	Q3	Rango
Intervalo teórico (seg.)	327,9	127,2	244,0	330,0	390,0	1 – 2.081
Intervalo real (seg.)	372,8	267,2	199,0	337,0	486,0	1 – 3.599
Headway ratio	1,57	11,24	0,67	1,02	1,45	0,00 – 3.212
Validaciones (por 5 min)	49,8	65,4	11,0	27,0	60,0	1 – 1177
Apertura puertas (seg.)	16,6	29,8	9,0	13,0	18,0	0,00 – 3.522

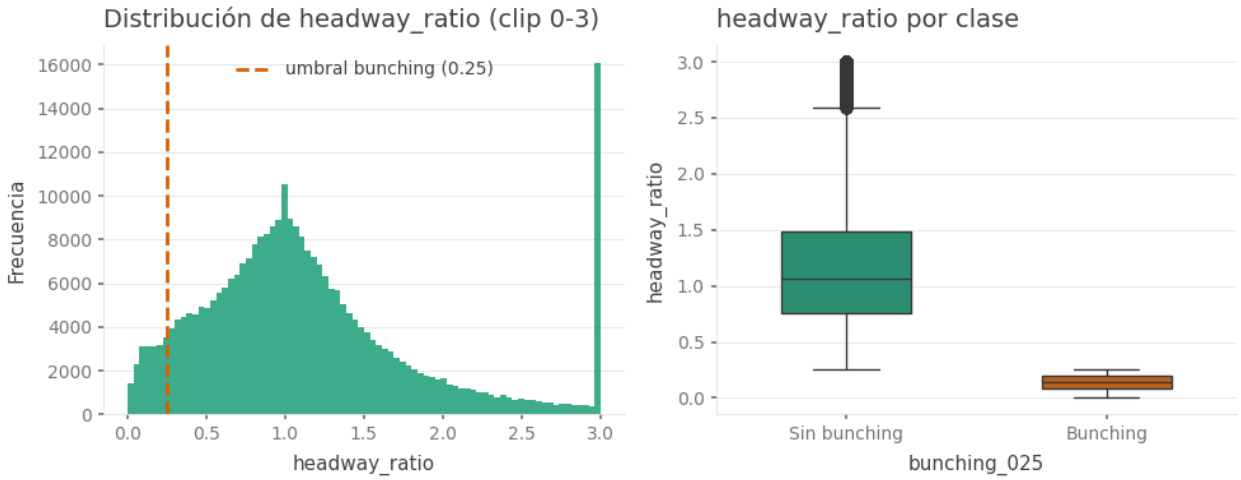
Fuente: Elaboración propia

En la tabla anterior se evidencia que el intervalo real promedio (372,77 s) supera al teórico (327,91 s) en aproximadamente un 13,7 %, lo que sugiere una tendencia operativa a registrar arribos con intervalos mayores a los programados. Este comportamiento es coherente con la dinámica de un sistema BRT con corredores compartidos, donde la congestión sobre el carril troncal, los tiempos variables de embarque y desembarque, y las acciones de regulación en cabecera tienden a ampliar los intervalos reales por encima de la programación. La diferencia se acompaña de una dispersión también mayor: la desviación estándar del intervalo real (267,18 s) supera ampliamente la del teórico (127,19 s), y el rango intercuartílico real (199 a 486 s) es más amplio que el programado (244 a 390 s), lo que confirma la variabilidad que introduce la operación frente a la planificación nominal.

La mediana de *headway_ratio* se sitúa en 1,02 y su rango intercuartílico se extiende entre 0,67 y 1,45, lo que indica que el 50 % central de los arribos opera con desviaciones moderadas respecto a la programación. La desviación estándar reportada (11,24) y el máximo de 3.212 obedecen a un grupo reducido de registros con intervalos teóricos muy pequeños, donde el cociente se vuelve numéricamente inestable. Estos casos no reflejan el comportamiento típico del sistema y deberán examinarse con mayor detalle en las secciones siguientes.

Las variables de validaciones y apertura puertas presentan medias claramente superiores a sus medianas: validaciones (49,84 frente a 27,00) y apertura puertas (16,60 s frente a 13,00 s) registran además máximos elevados (1.177 validaciones y 3.522 s respectivamente). El rango intercuartílico de validaciones (11 a 60) y de apertura puertas (9 a 18 s) muestra que la operación habitual ocurre en valores muy inferiores al máximo, patrón esperable en una red con estaciones heterogéneas donde los nodos de alta demanda en hora pico y los puntos de intercambio troncal generan registros muy por encima del comportamiento medio.

Figura 9. Distribución del *headway ratio*



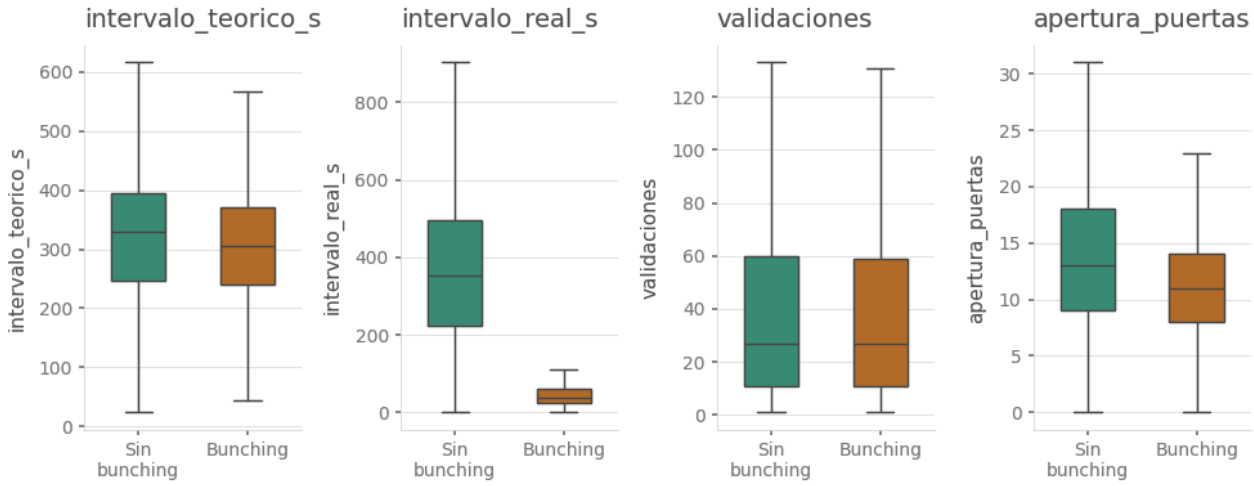
Fuente: Elaboración propia

La presencia de valores atípicos extremos en el *headway ratio* y el intervalo real motivó la aplicación de *winsorización* en el percentil 99 como parte del preprocesamiento del Pipeline B.

Con respecto a los resultados obtenidos para la variable *apertura_puertas*, los registros con valor cero se han identificado como operaciones no válidas. Estas inconsistencias, originadas por fallos en los sensores de puerta o problemas en la transmisión de datos, reciben el mismo tratamiento que los valores nulos. Por tanto, han sido excluidos del cálculo de indicadores para evitar sesgos en los resultados de operación. Los umbrales resultantes se presentan más adelante en la sección de preprocesamiento.

La siguiente figura presenta boxplots de las variables numéricas separados por clase (*bunching* vs. no *bunching*).

Figura 10. *Boxplots* por clase de *bunching*

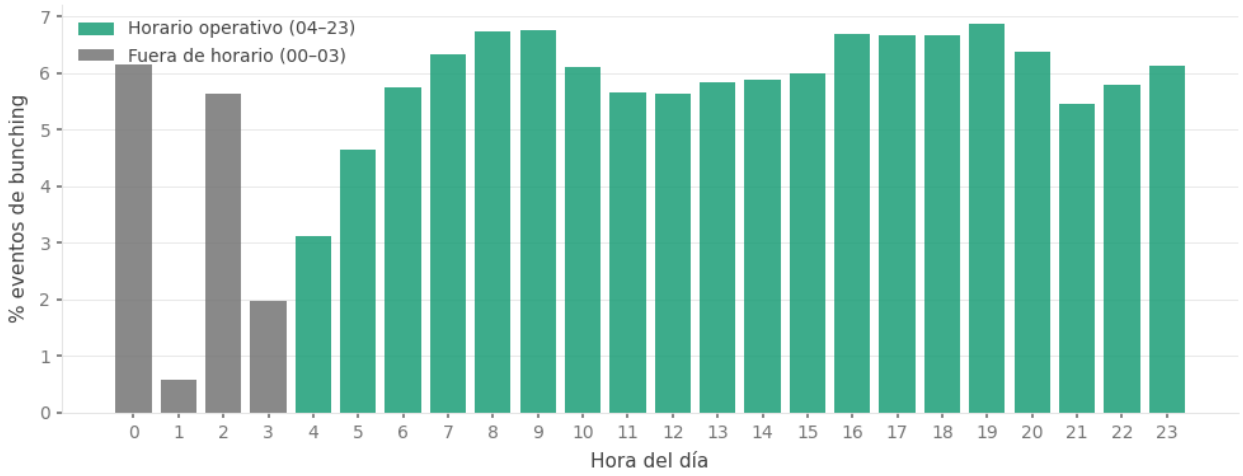


Fuente: Elaboración propia

5.1.5 Patrones temporales

El análisis de la tasa de *bunching* por hora del día revela un patrón consistente con los ciclos de demanda del sistema. Las horas de menor incidencia dentro del horario operativo son las primeras del servicio (hora 4, con 3,20%). La tasa aumenta progresivamente durante la mañana, alcanzando un primer pico en la hora 8 (6,74%), se estabiliza durante el mediodía en torno al 5,7%–6,1%, y vuelve a subir en la tarde hasta alcanzar el máximo en la hora 19 (6,85%). Este perfil temporal es coherente con la hipótesis de que el *bunching* se intensifica en los períodos de mayor demanda, cuando las perturbaciones en los tiempos de embarque se propagan con mayor facilidad entre buses consecutivos. La Figura 11 muestra la tasa de *bunching* por hora del día.

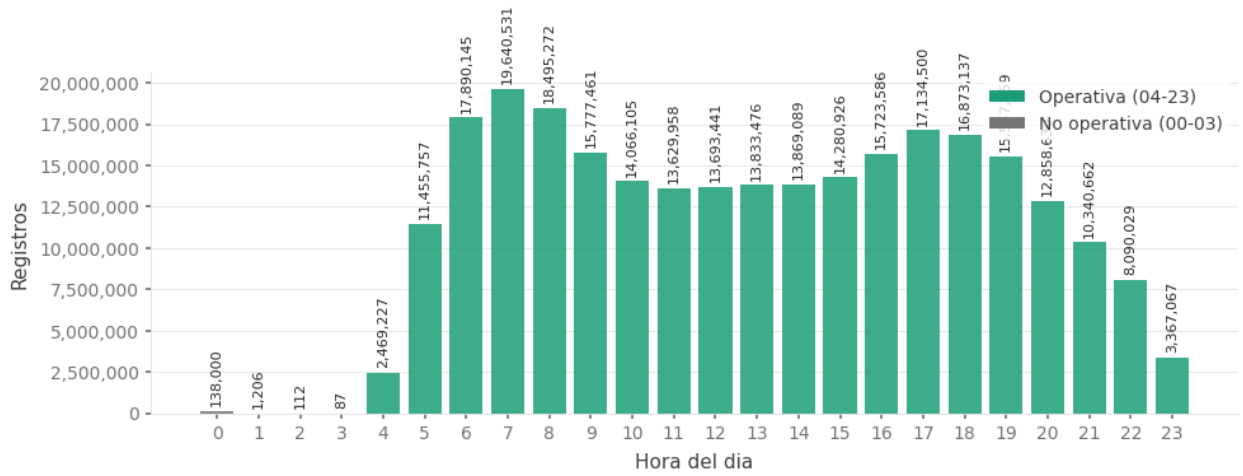
Figura 11. Tasa de *bunching* por hora del día



Fuente: Elaboración propia

El patrón horario motiva la pregunta de si la ventana completa 00–23 aporta señal operativa homogénea. El volumen de registros por hora del día, presentado en la Figura 12, revela una asimetría marcada: las horas 00–03 concentran en conjunto menos del 0,05 % de los registros totales, con volúmenes entre 10^2 y 10^5 observaciones por hora frente a los ~10^7 que exhiben las horas operativas (04–23). Estos registros residuales corresponden a la ventana no operativa del subsistema troncal (servicios especiales, operaciones de mantenimiento y artefactos de la convención GTFS de servicios nocturnos (*hora_real* ≥ 24:00), y distorsionan las agregaciones por hora, los *heatmaps* y los *rankings* presentados a continuación.

Figura 12. Volumen de registros por hora del día

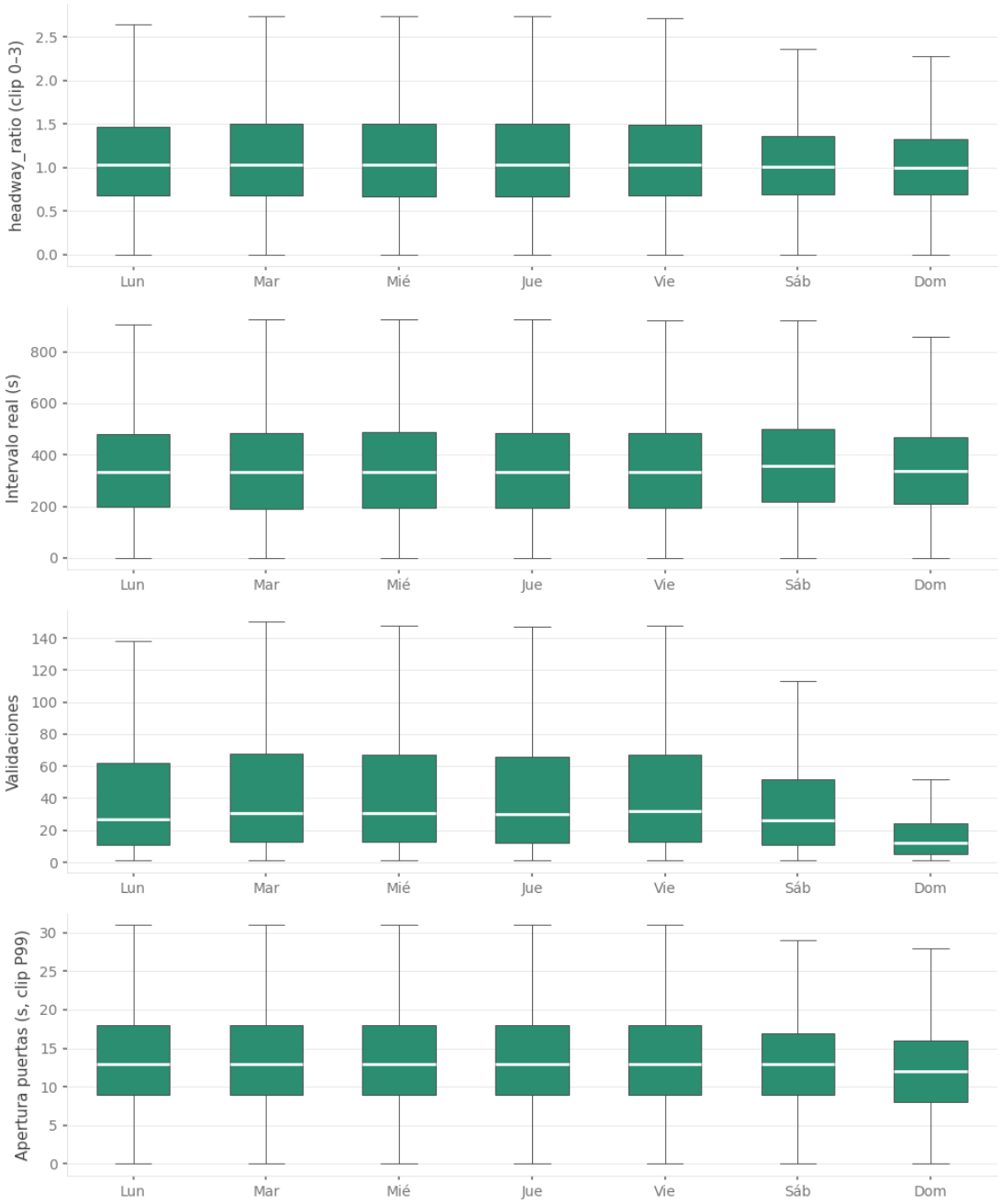


Fuente: Elaboración propia

Con base en este hallazgo, el análisis exploratorio restringe de aquí en adelante el *dataset* al horario operativo 04:00–23:00.

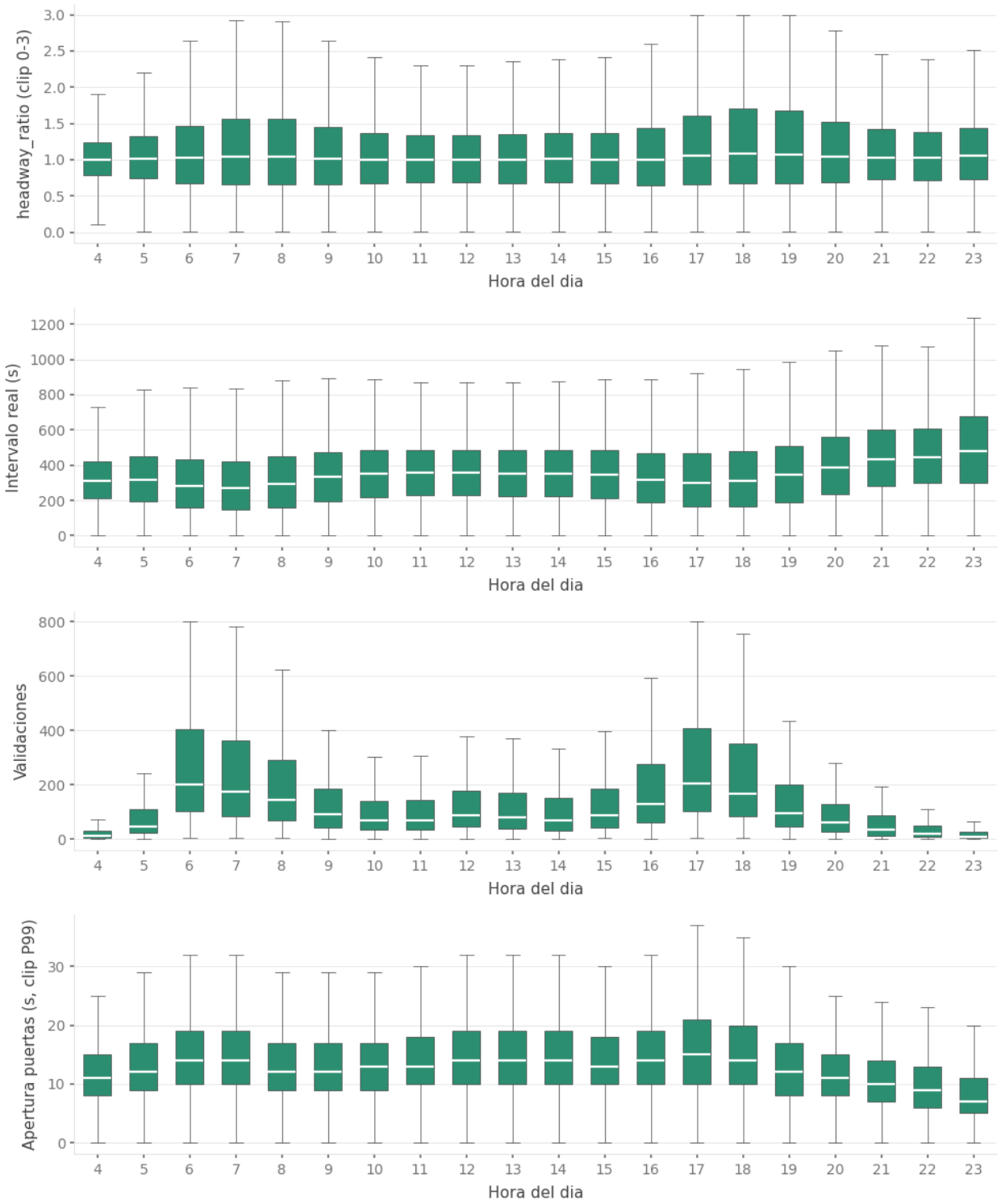
El análisis por día de la semana muestra una reducción menor los fines de semana. La siguiente figura presenta *boxplots* de las variables principales por día de la semana.

Figura 13. Distribuciones por día de la semana



Fuente: Elaboración propia

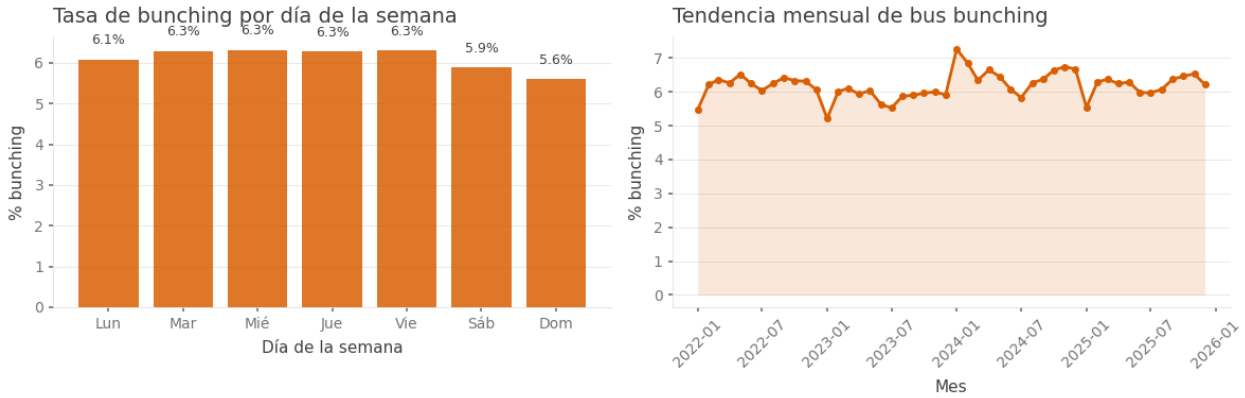
Figura 14. Distribuciones por hora del día



Fuente: Elaboración propia

El análisis de la tendencia diaria a lo largo del período completo permite identificar cambios estructurales y estacionalidad en la serie.

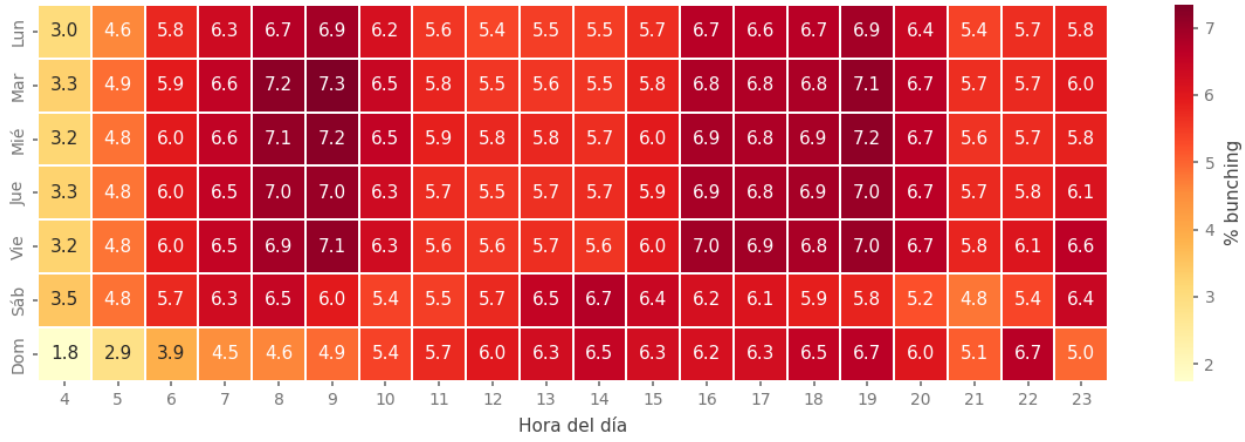
Figura 15. Tendencia diaria y mensual de la tasa de *bunching*



Fuente: Elaboración propia

El siguiente mapa de calor cruza la hora del día con el día de la semana, permitiendo identificar las franjas de mayor concentración de *bunching*.

Figura 16. *Heatmap* de *bunching*: hora del día vs. día de la semana

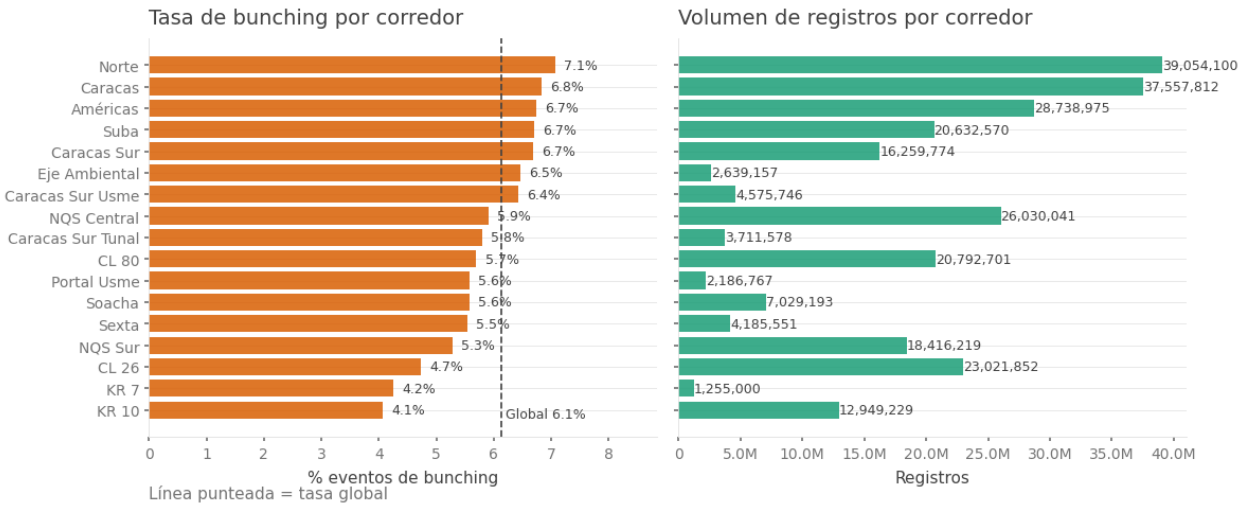


Fuente: Elaboración propia

5.1.6 Patrones espaciales

La distribución geográfica del *bunching* muestra diferencias entre los corredores troncales. El corredor *Norte* presenta la tasa más alta (7,1%), seguido por *Caracas* (6,8%) y *Américas* (6,7%). En contraste, los corredores *KR 10* (4,1%) y *KR 7* (4,2%) muestran las tasas más bajas. Esta variabilidad sugiere que las condiciones operativas de cada corredor influyen en la propensión al bunching, lo que motivó la inclusión del *target encoding* por estación como variable en el modelamiento. La siguiente figura muestra la tasa de *bunching* por zona geográfica.

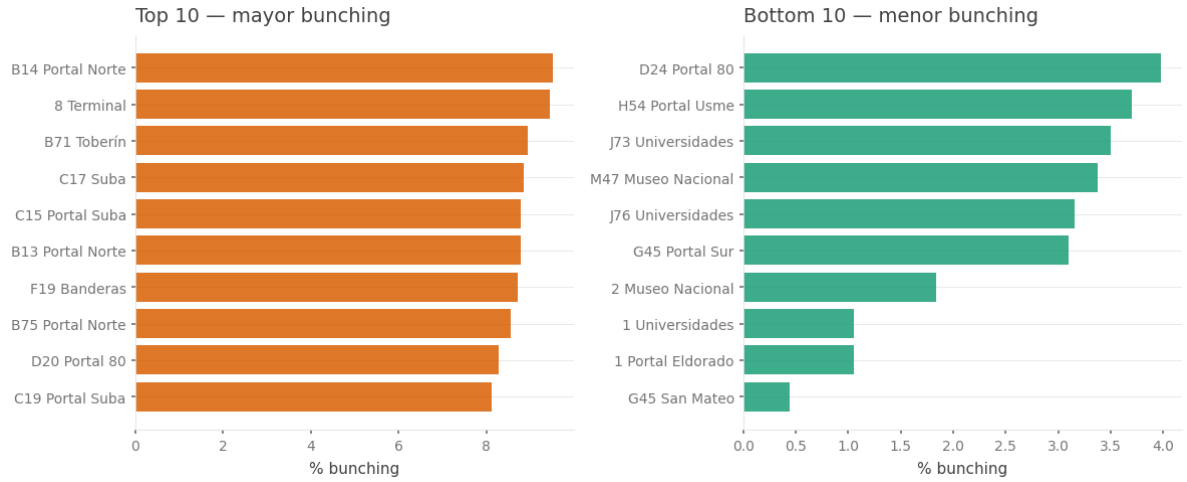
Figura 17. Tasa de *bunching* por zona (corredor troncal)



Fuente: Elaboración propia

A nivel de servicio, la variabilidad es aún más pronunciada. Los diez servicios con mayor propensión al *bunching* alcanzan tasas entre 8,2% y 9,7%, mientras que los servicios menos afectados presentan tasas inferiores al 1%. El servicio *B14 Portal Norte* tiene la tasa más alta (9,7%), seguido por *8 Toberín* (9,4%) y *B71 Toberín* (9,0%). De los 135 servicios troncales, la tasa promedio es 5,7%. La siguiente figura muestra la tasa de *bunching* por servicio.

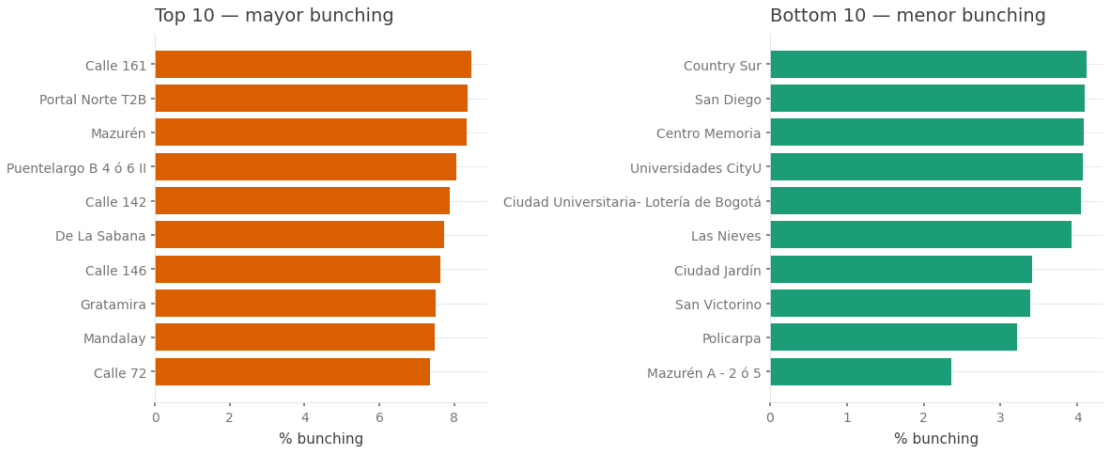
Figura 18. Tasa de bunching por servicio



Fuente: Elaboración propia

El análisis por estación permite identificar puntos específicos de la red con mayor concentración de eventos.

Figura 19. Tasa de *bunching* por estación



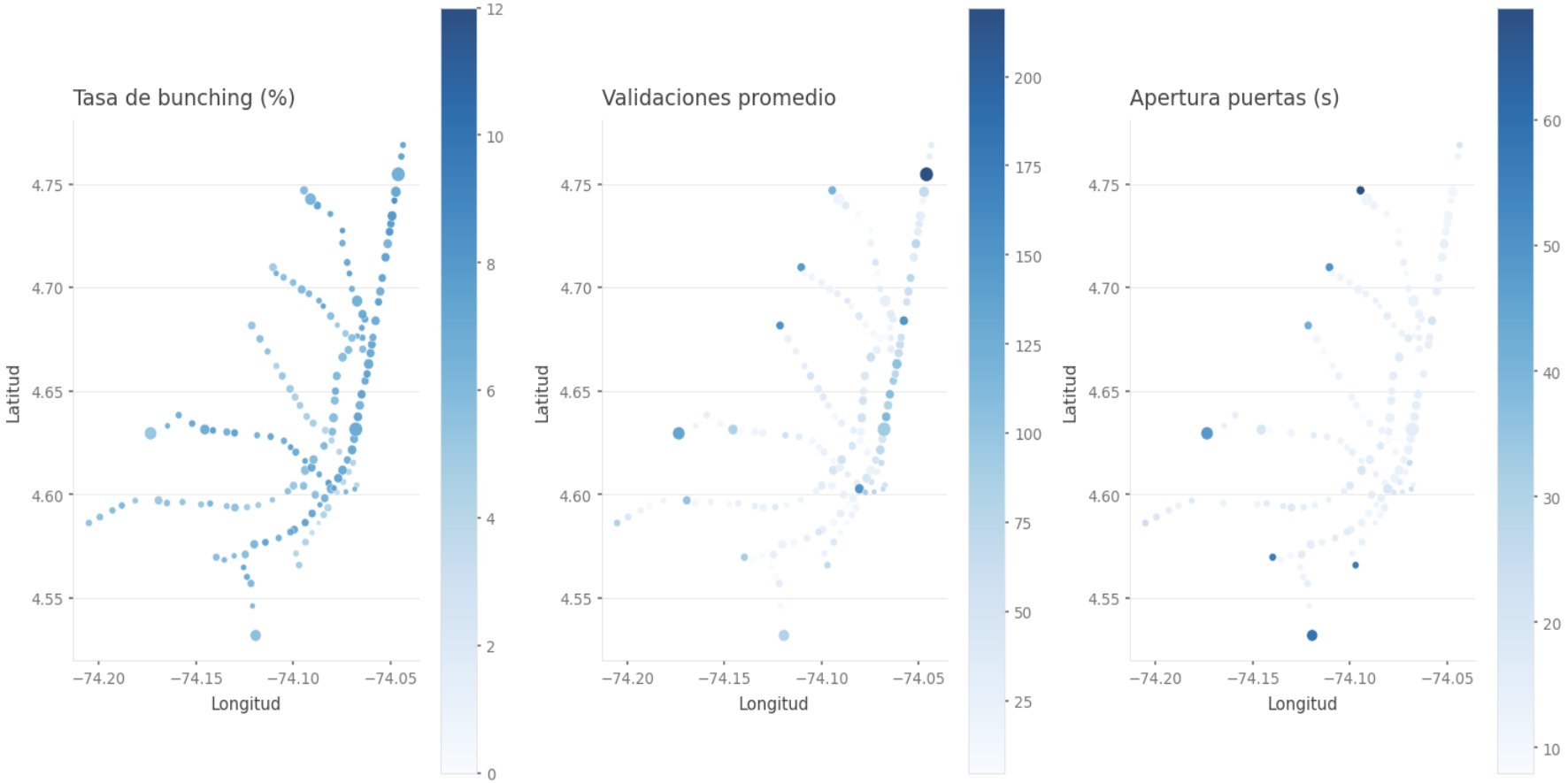
Fuente: Elaboración propia

El análisis espacial se complementa con la visualización georreferenciada de los indicadores clave por estación. La Figura 20 presenta tres mapas de dispersión sobre las coordenadas de la red troncal, donde cada punto corresponde a una estación y su tamaño es proporcional al volumen de registros; los mapas muestran, respectivamente, la tasa de *bunching*, el promedio de validaciones y el tiempo promedio de apertura de puertas. Esta representación permite identificar concentraciones territoriales de los tres indicadores y evidencia que el *bunching* no se distribuye de forma homogénea en la red.

Los tres mapas convergen en un patrón espacial coherente: la tasa de *bunching* se concentra en los corredores *Norte, Caracas y Américas* (donde varias estaciones superan el 7 %), mientras que los corredores *KR 10* y *CL 26* muestran tasas por debajo del promedio. La superposición con el mapa de validaciones confirma que las estaciones con *bunching* más alto coinciden con las de mayor demanda agregada, pero no en proporción uno a uno: hay estaciones con demanda intermedia y tasas de *bunching* elevadas, lo que sugiere que la presión de pasajeros no actúa de forma aislada. El mapa de tiempo promedio de apertura de puertas refuerza esta lectura al mostrar valores más altos en estaciones de intercambio del eje norte y central, donde el tiempo de servicio absorbido en plataforma se acumula y favorece la desalineación con el bus siguiente.

En conjunto, la georreferenciación indica que el *bunching* es un fenómeno territorialmente sesgado hacia los corredores de mayor frecuencia y demanda combinadas, hallazgo que justifica la inclusión de la codificación por estación y de las variables del corredor en la fase de modelado.

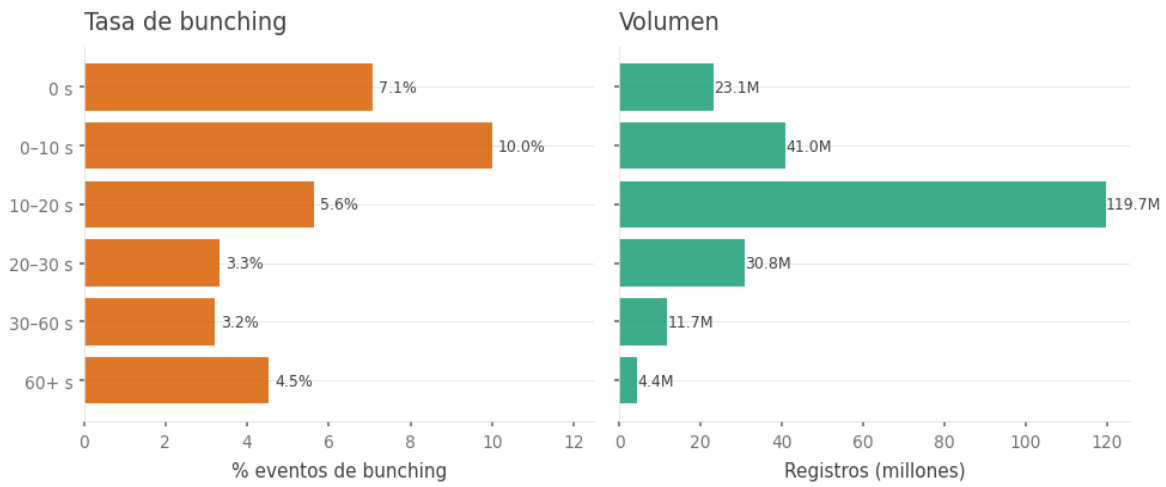
Figura 20. Mapas de dispersión georreferenciados por estación



Fuente: Elaboración propia

El análisis de la tasa de *bunching* por rango de apertura de puertas, presentado en la Figura 21, permite explorar si la duración del evento de embarque/desembarque incide en la formación del fenómeno. La apertura de puertas es la única variable específica del bus, además del intervalo real, y constituye un indicador directo de la demanda en cada arribo, por lo que tramos largos podrían anticipar desalineaciones aguas abajo en el corredor.

Figura 21. Tasa de bunching por rango de apertura de puertas



Fuente: Elaboración propia

La relación entre el *dwell time* y la tasa de bunching no es monótona. El tramo más corto sin contar el caso 0 s (de 0 a 10 s) registra la mayor incidencia, con una tasa del 10,00 % frente al 6,14 % global, mientras que los tramos intermedios (20–30 s y 30–60 s) presentan tasas claramente inferiores al promedio (3,31 % y 3,20 %). Los tramos largos (60+ s) muestran un repunte moderado (4,52 %) y los registros con apertura de puertas igual a cero se sitúan ligeramente por encima del promedio (7,07 %).

Este patrón es coherente con la dinámica operativa del *bunching*. Cuando dos buses circulan muy próximos entre sí, el segundo encuentra el andén con poca demanda residual (los pasajeros ya abordaron en el bus líder) y registra una apertura de puertas mínima, lo que explica la concentración de eventos en los tramos cortos. En el otro extremo, los tramos de 20 a 60 segundos corresponden a estaciones con demanda regular o alta: el tiempo de embarque y desembarque absorbe segundos efectivos del intervalo y, paradójicamente, ayuda a mantener la separación con el siguiente vehículo, lo que se refleja en tasas de *bunching* por debajo del promedio. El repunte del tramo 60+ s probablemente recoge estaciones de intercambio o eventos operativos puntuales en los que la demora prolongada altera el ritmo programado.

El comportamiento no monótono observado tiene una implicación directa para el modelado: la apertura de puertas considerada de forma aislada no es un buen discriminador del *bunching*, ya que tasas similares aparecen en extremos opuestos de la distribución. Su valor predictivo se manifestará principalmente al combinarla con la dinámica reciente del servicio (rezagos del

headway y de la propia variable) y con indicadores de demanda contextuales como las validaciones.

5.1.7 Descomposición STL

Se aplicó una descomposición STL (*Seasonal and Trend decomposition using Loess*) con período semanal (7 días) a las series temporales diarias de las variables principales. Los resultados revelan estructuras temporales marcadamente distintas entre variables.

Tabla 10. Descomposición STL de las variables principales

Variable	Tendencia (%)	Estacionalidad (%)	Residual (%)
% Bunching	37,4	27,0	39,6
Validaciones	11,8	69,5	28,9
Apertura puertas	92,8	2,1	2,9
Headway ratio	0,1	0,3	99,4

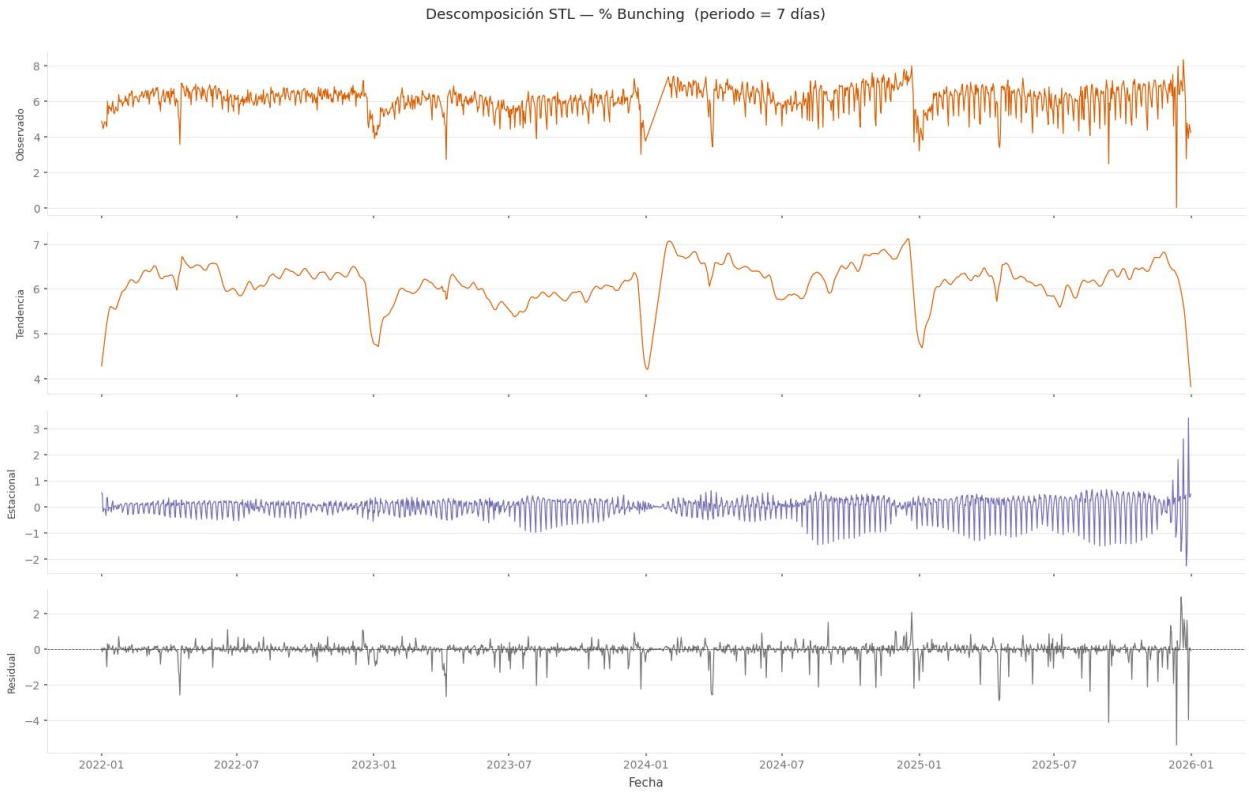
Fuente: Elaboración propia

El hallazgo más relevante corresponde al *headway* ratio, cuya varianza se concentra casi en su totalidad en el componente residual (99,4 %). Este resultado define al bunching como un fenómeno impulsivo y local, cuya estructura no es capturada por la tendencia ni por la estacionalidad semanal. La implicación para el modelamiento es directa: los modelos deben capturar patrones locales de corto plazo más que tendencias o ciclos estacionales, lo que justifica la inclusión de variables de rezago (*lags*) y de promedios móviles (*rolling*) en el *feature engineering*.

En contraste, las validaciones muestran un fuerte componente estacional (69,5 %), consistente con los ciclos semanales de demanda. La apertura de puertas presenta un dominio de tendencia (92,8 %) con estacionalidad y residual marginales (2,1 % y 2,9 %). Contrario a la hipótesis preliminar de comportamiento intermedio entre validaciones y *headway ratio*, el resultado sugiere que la duración media de apertura en la red evoluciona como una señal de baja frecuencia a lo largo del periodo, sin ciclo semanal pronunciado. La interpretación operativa apunta a cambios graduales en la operación o composición de la flota más que a patrones cíclicos.

Este componente estacional fuerte de las validaciones explica también los dos valles recurrentes que la descomposición revela en cada año del periodo: el primero coincide con la Semana Santa y el segundo con la transición de diciembre a enero. En ambos episodios la demanda agregada del sistema cae de forma marcada por el cierre de actividades escolares y laborales y por la salida de la ciudad de una parte importante de los usuarios habituales, y esa relajación de la presión de demanda se traduce en un descenso visible de la tasa de bunching respecto al promedio anual. La coincidencia es consistente con el mecanismo descrito en el marco teórico: con menos pasajeros acumulándose en plataforma, el bucle de retroalimentación positiva que sostiene el agrupamiento pierde fuerza.

Figura 22. Descomposición STL de la tasa de *bunching*

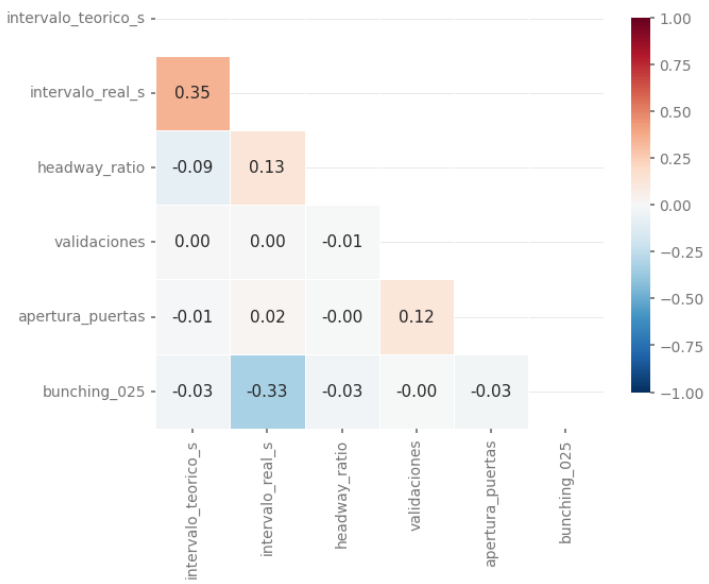


Fuente: Elaboración propia

5.1.8 Análisis de correlación y asociaciones

El análisis de correlación se aborda en dos planos complementarios. En primer lugar, se examina la correlación lineal de Pearson entre las *variables* continuas, lo que permite identificar redundancias y patrones de multicolinealidad relevantes para el preprocesamiento. En segundo lugar, se evalúa la asociación con la variable objetivo *bunching_025*, que es binaria y requiere métricas específicas: la correlación *point-biserial* (equivalente a Pearson para una variable binaria), la correlación *rank-biserial* (no paramétrica, robusta a valores atípicos, derivada del estadístico de *Mann-Whitney*) y la información mutua normalizada (que mide dependencia tanto lineal como no lineal). Reportar las tres métricas en conjunto evita conclusiones sesgadas por una única lectura.

Figura 23. Matriz de correlación entre variables



Fuente: Elaboración propia

La matriz anterior revela que las variables continuas presentan correlaciones moderadas o bajas entre sí, lo que sugiere ausencia de multicolinealidad fuerte. La evaluación de la asociación con la variable objetivo binaria se presenta a continuación con métricas específicas para ese caso.

Tabla 11. Asociación de variables con la variable objetivo

Variable	Point-biserial	Rank-biserial	MI normalizada
intervalo_teorico_s	-0,032	0,097	0,016
intervalo_real_s	-0,327	0,983	0,765
headway_ratio	-0,033	1,000	1,000
validaciones	-0,005	0,007	0,002
apertura_puertas	-0,026	0,221	0,024

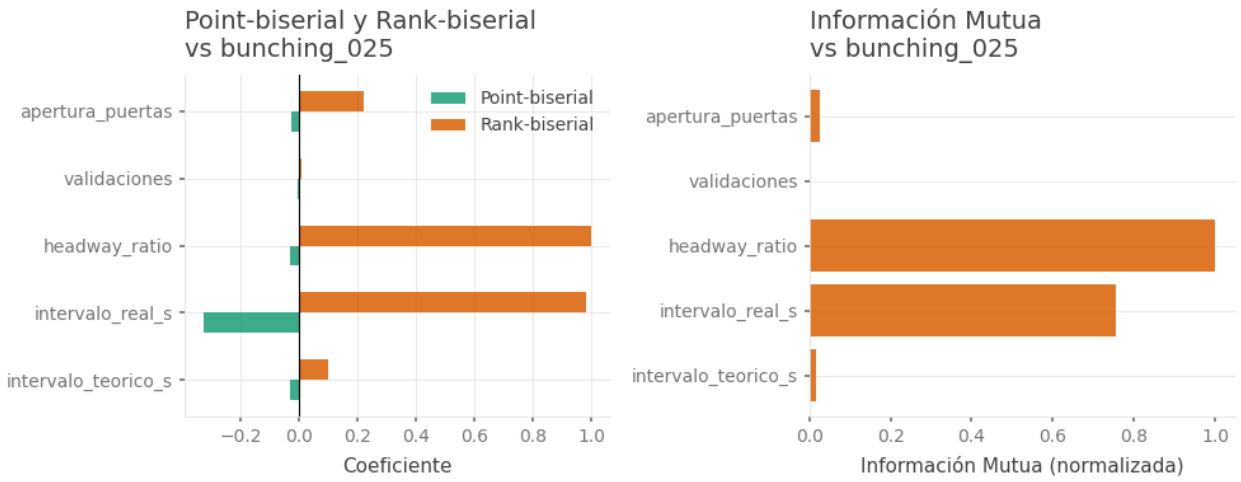
Fuente: Elaboración propia

La Tabla anterior reporta que las dos variables con mayor asociación con la variable objetivo son el *headway ratio* (*rank-biserial* = 1,000, MI = 1,000) y el intervalo real (*rank-biserial* = 0,983, MI = 0,765). Sin embargo, ambas variables constituyen información posterior al evento: el *headway ratio* se calcula a partir del intervalo real, que solo se conoce después de que el bus ha llegado a la estación. Su inclusión como predictores generaría *data leakage*, por lo que se excluyen del conjunto de variables del modelo predictivo. Esto no implica que la información histórica de estas variables se descarte por completo: en la siguiente etapa de *feature engineering* se generan variables de rezago (*lag_hr_1, lag_hr_2, lag_hr_3*, etc.) a partir de los valores del *headway ratio* en los arribos previos del mismo servicio a la misma estación, los cuales sí están disponibles antes del evento en t y aportan precisamente la dinámica reciente que el modelo necesita aprender.

El intervalo teórico muestra una asociación débil pero no trivial (*rank-biserial* = 0,097), lo que justifica su inclusión. Las validaciones presentan una asociación prácticamente nula con el *target* a nivel bivariado (*rank-biserial* = 0,007, MI = 0,002), aunque podrían contener información predictiva en interacción con otras variables o en contexto temporal.

La apertura de puertas registra una asociación de rango moderada-baja (rank-biserial = 0,221), consistente con la mayor concentración de eventos de *bunching* en tramos cortos de apertura observada previamente; no obstante, su información mutua normalizada es baja (0,024) y la correlación *point-biserial* cercana a cero (−0,026), lo que indica que, tomada de forma aislada, no constituye un discriminador fuerte del *target*. Su valor predictivo deberá evaluarse en interacción con la dinámica reciente del servicio (*lags* de *headway* y de la propia variable). Las figuras siguientes presentan la matriz de correlación entre variables y el análisis de asociación con el *target*.

Figura 24. Asociación de variables con el *target*

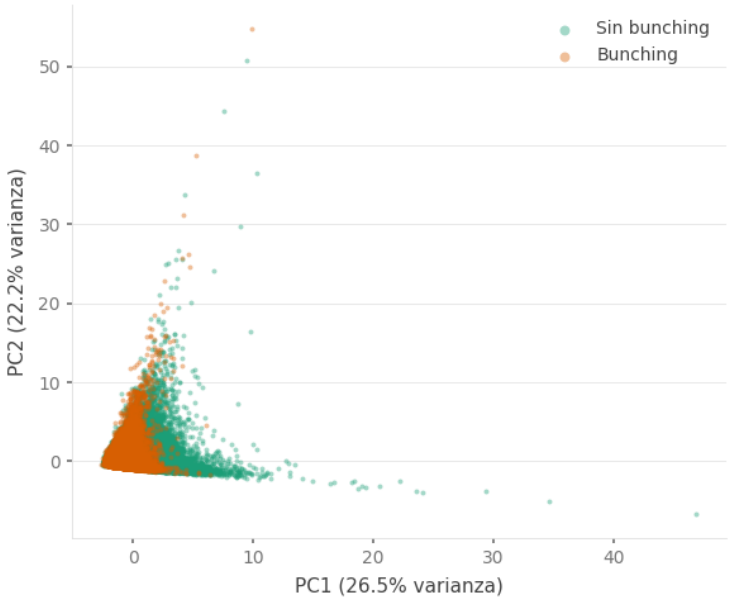


Fuente: Elaboración propia

5.1.9 Análisis de componentes principales e interacciones

Se realizó un análisis de componentes principales (PCA) y un análisis de interacciones bivariadas para explorar relaciones no lineales entre las variables.

Figura 25. Análisis de componentes principales



Fuente: Elaboración propia

En la gráfica anterior se observa un solapamiento considerable entre clases: los puntos de *sin bunching* y *bunching* aparecen muy mezclados, sobre todo cerca del origen. Esto indica que, usando solo las primeras dos componentes, las clases no se separan claramente. Eso sugiere que el fenómeno no es fácilmente separable en un espacio lineal de 2 dimensiones construido a partir de las variables originales.

5.2 Preprocesamiento

El preprocesamiento transforma el *dataset* unificado de la capa *bronze* en los conjuntos de las capas *silver* y *gold*, listos para el modelamiento. Todas las transformaciones siguen una regla fundamental de no ocasionar fuga de información: los parámetros de cada transformación (medianas de imputación, coeficientes de *target encoding*, umbrales de *winsorización*, parámetros de escalado) se calculan exclusivamente sobre el conjunto de entrenamiento y se aplican sin reajustar sobre el conjunto de prueba.

5.2.1 Objetivos y metodología del preprocesamiento

El preprocesamiento se desarrolló con cuatro objetivos específicos:

- **Preparar los datos para el modelamiento predictivo:** transformar el *dataset* unificado de la capa *bronze* en conjuntos estructurados y consistentes para entrenamiento y prueba, adecuados para los distintos algoritmos evaluados.

- **Garantizar la no filtración de información (*anti-leakage*):** asegurar que todos los parámetros de imputación, codificación, *winsorización* y escalado se estimen exclusivamente sobre el conjunto de entrenamiento y se apliquen posteriormente al conjunto de prueba sin reajuste.
- **Adecuar las variables a los requerimientos de cada familia de modelos:** construir un *pipeline* para modelos basados en árboles como *XGBoost*, que no requieren escalado, y otro para modelos sensibles a la escala y distribución de las variables, como regresión logística, *LS-SVM* y redes neuronales.
- I**ncorporar variables derivadas con valor predictivo:** generar variables temporales, espaciales y operacionales que capturen la dinámica local del servicio, la heterogeneidad por estación y servicio, y las interacciones relevantes para la predicción del *bunching*.

El preprocesamiento se estructura como una secuencia de transformaciones aplicada al *dataset* unificado, con el propósito de producir las capas *silver* y *gold* listas para el modelamiento:

- **Partición temporal Train/Test:** se define una fecha de corte de modo que los registros anteriores conforman el conjunto de entrenamiento y los posteriores el conjunto de prueba, respetando la secuencia temporal de los datos.
- **Imputación de valores nulos:** de acuerdo con los resultados del EDA, las validaciones faltantes se imputan con cero, mientras que la apertura de puertas se imputa con la mediana por combinación de servicio e *id_estacion* calculada únicamente sobre Train, con una mediana global de respaldo para combinaciones sin histórico. Los registros de apertura igual a cero se interpretan como inconsistencias, originadas por fallos en los sensores de puerta o problemas en la transmisión de datos y reciben el mismo tratamiento que un valor nulo, de modo que ambos se sustituyen por la mediana del grupo.
- **Codificación de variables categóricas de alta cardinalidad:** las variables con alta cardinalidad se transforman mediante *target encoding* con suavizado, calculado exclusivamente sobre Train, para capturar diferencias sistemáticas en la propensión al *bunching* sin recurrir a codificación *one-hot*.
- **Transformaciones específicas por tipo de modelo:** para un *pipeline* se conservan las variables en su escala original, orientadas a modelos de árboles; para otro pipeline se añade winsorización en percentil 99, transformación logarítmica y escalado, con el fin de estabilizar distribuciones y homogeneizar escalas.
- **Construcción de variables derivadas en la capa *gold*:** se generan variables temporales, rezagos, estadísticas móviles, derivadas, acumuladas e interacciones, calculadas por servicio y orden temporal, para representar patrones locales de corto plazo relevantes para el fenómeno. A estas se suman, en una iteración posterior, variables cross-station

(estado de otros servicios en la misma estación y sentido) y variables *upstream* (aguas arriba) del mismo servicio en las estaciones inmediatamente anteriores de su recorrido.

- **Generación de datasets finales de modelamiento:** como resultado se construyeron los conjuntos de entrenamiento y prueba de las capas *silver* y *gold*, con y sin escalado, preservando la consistencia entre particiones y pipelines.

5.2.2 Restricción al horario operativo

El primer paso del preprocesamiento consiste en restringir el *dataset* unificado a la franja horaria 04:00–23:00, que corresponde al horario regular de operación de la red troncal de TransMilenio. Esta decisión se apoya en criterios operativos y estadísticos complementarios. Desde el punto de vista operativo, el sistema no presta servicio continuo durante la madrugada; la ventana 04:00–23:00 coincide con los rangos publicados por el operador. Los registros fuera de esta franja corresponden a operaciones de mantenimiento o artefactos derivados de la convención GTFS de codificación de servicios nocturnos (*hora_real* ≥ 24:00), ninguno de los cuales es representativo del patrón operativo que el modelo debe anticipar.

Desde el punto de vista estadístico, el análisis exploratorio mostró que las horas 00–03 concentran en conjunto apenas del 0,05% de los registros totales, con volúmenes de entre 10^2 y 10^5 observaciones por hora frente a los ~10^7 que exhiben las horas operativas. Un volumen tan reducido impide estimar con confianza prevalencias horarias, *target encodings* y cuantiles de *winsorización*, todos estadísticos de referencia que se calculan sobre el conjunto de entrenamiento. Incluir estas franjas contaminaría las tablas de referencia con valores atípicos y degradaría las variables temporales (*lags*, ventanas deslizantes, hora codificada) que requieren vecindad temporal densa para ser informativas.

5.2.3 Partición temporal Train/Test

Se adoptó una partición temporal con fecha de corte el 1 de enero de 2025. Todos los registros anteriores a esa fecha conforman el conjunto de entrenamiento (*Train*) y los registros a partir de esa fecha conforman el conjunto de prueba (*Test*). Esta estrategia respeta la naturaleza temporal de los datos y evita la filtración de información futura al conjunto de entrenamiento.

Tabla 12. Partición temporal de los conjuntos Train y Test

Conjunto	Período	Filas	*% bunching*
Train	2022-01-01 a 2024-12-31	206.365.299 (77%)	6,13%
Test	2025-01-01 a 2025-12-31	62.670.966 (23%)	6,19%
Total	2022-01-01 a 2025-12-31	269.036.265	6,14%

Fuente: Elaboración propia

La proporción de *bunching* se mantiene estable entre el conjunto de entrenamiento (6,13 %) y el de prueba (6,19 %), lo que indica que no hay un cambio estructural en la prevalencia del fenómeno

entre los dos períodos. El desglose anual muestra tasas entre 5,84 % (2023) y 6,44 % (2024), valores compatibles con la prevalencia global del 6,14 % observada en el *dataset* unificado.

5.2.4 Tablas de parámetros del *pipeline*

Varias transformaciones del preprocesamiento dependen de parámetros que se calculan una sola vez sobre el conjunto de entrenamiento y se aplican idénticamente al entrenamiento, a la prueba y a cualquier dato futuro de inferencia. Para no mantenerlos como diccionarios en memoria y reinyectarlos como ramificaciones extensas dentro de las consultas, los parámetros se almacenan como tablas de búsqueda y se consultan desde las sentencias SQL que construyen las capas *silver y gold*. Estas tablas funcionan como artefactos del *pipeline* y se mantienen separadas de las tablas de negocio que residen en las capas *bronze, silver y gold*.

Cada tabla sigue el patrón *fit-once*: el cálculo se realiza una única vez sobre el periodo de entrenamiento (2022-01-01 a 2024-12-31), y a partir de ese momento solo se hacen consultas. Esta convención reproduce en SQL el comportamiento que tendría un objeto de *scikit-learn* ajustado con *fit* sobre el conjunto de entrenamiento y aplicado con *transform* al entrenamiento y a la prueba sin reajustar, lo que garantiza que ninguna información del conjunto de prueba contamine los parámetros y se introduzca fuga de información (*data leakage*).

El pipeline materializa tres tablas de parámetros. Una almacena la mediana de apertura de puertas por combinación de servicio y estación, utilizada para imputar registros nulos y ceros; otras dos contienen los valores suavizados ($\alpha = 10$) del *target encoding* para servicio y estación, y reemplazan estos identificadores por su tasa estimada de bunching en el conjunto de entrenamiento. Cuando una combinación del conjunto de prueba no aparece en la tabla, la consulta devuelve nulo y se recurre a un valor de respaldo global (mediana global de apertura de puertas o media global del objetivo), lo que garantiza cobertura completa sin contaminar el ajuste.

5.2.5 *Pipeline* A: modelos de árboles

El Pipeline A prepara los datos para modelos basados en árboles de decisión (*XGBoost, LightGBM*), que no requieren escalado de variables ni normalización de distribuciones. Las transformaciones aplicadas son las siguientes.

Imputación de valores nulos: las validaciones nulas se imputaron con el valor cero. La apertura de puertas se imputó con la mediana por combinación de servicio y estación, calculada sobre el conjunto de entrenamiento y almacenada en una tabla de referencia que reúne 1.752 combinaciones con histórico. Los registros con apertura igual a cero, interpretados como un fallo del equipo embarcado, se reclasificaron como valores faltantes y se imputaron con la misma mediana del grupo. Para combinaciones sin datos históricos se utilizó un valor de respaldo de 13 segundos, correspondiente a la mediana global de entrenamiento. Tras la imputación, los conjuntos de la capa *silver* no contienen valores nulos.

***Target encoding* de variables categóricas:** las variables servicio e identificador de estación son

categóricas de alta cardinalidad (118 servicios y 150 estaciones), lo que hace inviable el uso de codificación *one-hot*. Se aplicó *target encoding* con suavizado aditivo: para cada categoría, el valor codificado se calcula como un promedio ponderado entre la media del objetivo en la categoría y la media global, donde el peso depende del número de observaciones de la categoría y un parámetro de suavizado. Esta estrategia reduce el riesgo de sobreajuste en categorías con pocos registros. Los parámetros se calcularon exclusivamente sobre el conjunto de entrenamiento y se almacenaron en tablas de referencia auxiliares; las variables codificadas resultantes son *te_servicio y te_id_estacion*.

Los cinco servicios con mayor valor de *target encoding* (mayor propensión al bunching) son B14 Portal Norte (9,52 %), 8 Terminal (9,35 %), *F19* Banderas (8,95 %), B71 Toberín (8,94 %) y C15 Portal Suba (8,77 %). La amplitud del rango observado en Train (0,44 % a 9,52 %) confirma que la identidad del servicio contiene información predictiva relevante.

5.2.6 *Pipeline* B: modelos lineales y redes neuronales

El *Pipeline* B extiende las transformaciones del Pipeline A con operaciones adicionales requeridas por modelos sensibles a la escala y distribución de las variables, como *LS-SVM*, regresión logística y redes *LSTM*. Se aplican tres transformaciones sucesivas: *winsorización*, transformación logarítmica y escalado.

Winsorización en el percentil 99: las variables con colas pesadas (*headway_ratio, intervalo_real_s, validaciones y apertura_puertas*) se truncaron en el percentil 99 calculado sobre Train, para limitar la influencia de valores atípicos extremos sin eliminar registros. Para *apertura_puertas* se excluyeron del cálculo del umbral los registros nulos y los ceros, en coherencia con su tratamiento como valores faltantes.

Tabla 13. Umbrales de winsorización en el percentil 99

Variable	Umbral P99
headway_ratio	9,08
intervalo_real_s	1.278 s
validaciones	282
apertura_puertas	139 s

Fuente: Elaboración propia

Transformación logarítmica y escalado: tras la *winsorización*, se aplicó una transformación log(1 + x) a las variables de cola pesada (*headway_ratio, intervalo_real_s, validaciones y apertura_puertas*) para aproximar distribuciones más simétricas. Sobre los valores transformados se calcularon los parámetros de escalado: *StandardScaler* (media y desviación estándar) para la mayoría de las variables, y *RobustScaler* (mediana e IQR) para el intervalo teórico, cuya distribución resultó menos asimétrica. Los parámetros se ajustaron sobre el conjunto de entrenamiento y se aplicaron al conjunto de prueba sin reajustar. El resultado son los conjuntos de entrenamiento y prueba de la capa *silver* con escalado.

5.2.7 *Feature engineering*: capa *gold*

La capa *gold* incorpora variables derivadas que capturan la dinámica temporal del servicio y las interacciones entre variables. El diseño de esta capa se fundamenta en los hallazgos del EDA: la naturaleza impulsiva del bunching (99,4 % de varianza en el residual STL) justifica la inclusión de rezagos y promedios móviles de corto plazo, mientras que la variabilidad por servicio y estación motiva el *target encoding* y las interacciones. En una iteración posterior se incorporaron además dos familias que describen el estado del corredor más allá del propio par servicio-estación: las variables *cross-station*, que resumen la dinámica agregada de los servicios que pasan por la misma estación y en el mismo sentido de circulación en ventanas de cinco, quince y treinta minutos, y las variables *upstream service-level*, que describen el estado del mismo servicio en las estaciones inmediatamente anteriores de su recorrido. La construcción de estas dos familias requirió derivar primero, a partir de la geometría de la red y de los registros de arribos, el orden de las estaciones dentro de cada corredor y el sentido en que cada servicio lo recorre.

Todas las variables derivadas se calculan sobre cada servicio individualmente, ordenando por marca de tiempo, y los parámetros de escalado para los conjuntos de la capa *gold* con escalado se ajustan exclusivamente sobre el conjunto de entrenamiento. La Tabla 14 siguiente presenta las 24 variables principales del *dataset* de la capa gold.

Las tablas de la capa *gold* preservan además dos columnas de metadata operativa, la marca de tiempo y el identificador de servicio, que no son variables y se excluyen del vector de entrada en cada uno de los modelos. Se conservan únicamente para construir el objetivo multi-horizonte, desplazando el valor de la variable objetivo n arribos hacia adelante dentro de cada par servicio-estación, y para validar la ventana temporal en la fase de modelado.

Tabla 14. Variables del *dataset* de la capa gold

Categoría	**Variables**	**Descripción**
Temporales	*hora_dia, dia_semana, mes*	Contexto temporal del registro
Espacial	*id_estacion*	Identificador de estación
Operacionales	*intervalo_teorico_s, validaciones, apertura_puertas*	Variables observadas del instante
Encoding	*te_servicio, te_id_estacion*	*Target encoding* suavizado
Lags headway	*lag_hr_1, lag_hr_2, lag_hr_3*	*Headway ratio* de los 3 intervalos previos
Lags apertura puertas	*lag_ap_1, lag_ap_2*	Apertura de puertas de los 2 intervalos previos
Lag validaciones	*lag_val_1*	Validaciones del intervalo previo
Rolling headway	*roll_hr_mean_3, roll_hr_std_3*	Media y desv. est. móvil (ventana 3)
Rolling apertura puertas	*roll_ap_mean_3, roll_ap_std_3*	Media y desv. est. móvil (ventana 3)
Rolling validaciones	*roll_val_mean_3*	Media móvil de validaciones (ventana 3)

Derivadas	*delta_hr_1, delta_ap_1*	Diferencia respecto al intervalo previo
Acumulada	*headway_acum*	*Headway ratio* acumulado del servicio-día
Interacciones	*lag_hr1_x_ap*	Interacción entre *headway* previo y apertura de puertas
Cross-station	*station_mean_dwell_5min, station_mean_dwell_15min, station_max_dwell_5min, station_total_dwell_5min, station_n_arrivals_5min, station_n_arrivals_15min, station_n_arrivals_30min, station_bunching_rate_5min, station_bunching_rate_15min, station_dt_to_prev_arrival_s*	Estado agregado del corredor (misma estación y sentido) en ventanas de 5, 15 y 30 min
Upstream service-level	*upstream{1,2,3}_mean_hr_15min, upstream{1,2,3}_std_hr_15min, upstream{1,2,3}_bunching_rate_15min, upstream{1,2,3}_n_arrivals_15min*	Información del mismo servicio en las 3 estaciones anteriores del recorrido (ventana 15 min)
Metadata	*datetime, servicio*	Columnas operativas (no son variables). Preservadas para construir el *target multistep* y validar la ventana temporal.

Fuente: Elaboración propia

La construcción de los *lags* descarta el primer registro de cada par (*servicio, id_estacion*) que no dispone de historial previo. La pérdida resultante es de 17.612 filas sobre 206 millones de Train (0,009 %), despreciable en términos prácticos y sin efecto sobre el balance de clases.

La tabla siguiente presenta los volúmenes finales de las cuatro tablas *gold*, listas para el modelamiento.

Tabla 15. Datasets gold resultantes para modelamiento

Tabla	Filas	% bunching	Variables
gold.train	206.347.687 (77%)	6,13 %	46
gold.test	62.664.981 (23%)	6,19 %	46
gold.train_scaled	206.347.687 (77%)	6,13 %	46
gold.test_scaled	62.664.981 (23%)	6,19 %	46

Fuente: Elaboración propia

La proporción de *bunching* se preserva de forma consistente entre todas las particiones (*Train/Test*) y entre los dos pipelines (A/B), lo que confirma que las transformaciones no alteran el balance de clases ni introducen sesgos entre los conjuntos.

El detalle por variable (grupo, tipo, fuente original, rol, tratamiento de nulos, transformación del *Pipeline* B y descripción operativa) se documenta en una tabla maestra de variables generada durante la fase de selección, y complementa la categorización resumida de la Tabla 13.

Las variables *headway_ratio* e *intervalo_real_s* no se incluyen como variables predictoras en los conjuntos de la capa *gold* por riesgo de fuga de información (son información del evento en t, no del pasado). Sí se utilizan para construir variables de rezago (*lag_hr_1, lag_hr_2, lag_hr_3, roll_hr_mean_3, delta_hr_1, headway_acum),* que recogen la información del pasado de manera segura: en cada arribo en t el modelo recibe los valores del headway ratio de los tres arribos anteriores del mismo par servicio-estación, pero nunca el del propio evento en curso. Se conservan en los conjuntos de la capa *silver* únicamente para el cálculo de los rezagos. La marca de tiempo y el identificador de servicio se preservan en la capa *gold* como *metadata* operativa (no como variables predictoras): son necesarias para construir el objetivo multi-horizonte y para validar la ventana temporal, y los componentes de modelado las excluyen explícitamente del vector de entrada.

Mapeo entre conjuntos y modelos: los conjuntos sin escalar de la capa *gold* alimentan *XGBoost*; los conjuntos escalados de la capa *gold* alimentan *LS-SVM*. La red *LSTM* se apoya en cambio en los conjuntos de la capa *silver*. *LSTM* aprende los rezagos y patrones temporales de manera implícita desde la dimensión temporal de cada secuencia (servicio por fecha): alimentarla con los rezagos pre-computados sería redundante con la información que la red ya extrae del orden de los pasos en la secuencia, por lo que se utiliza el conjunto escalado de la capa *silver* directamente.

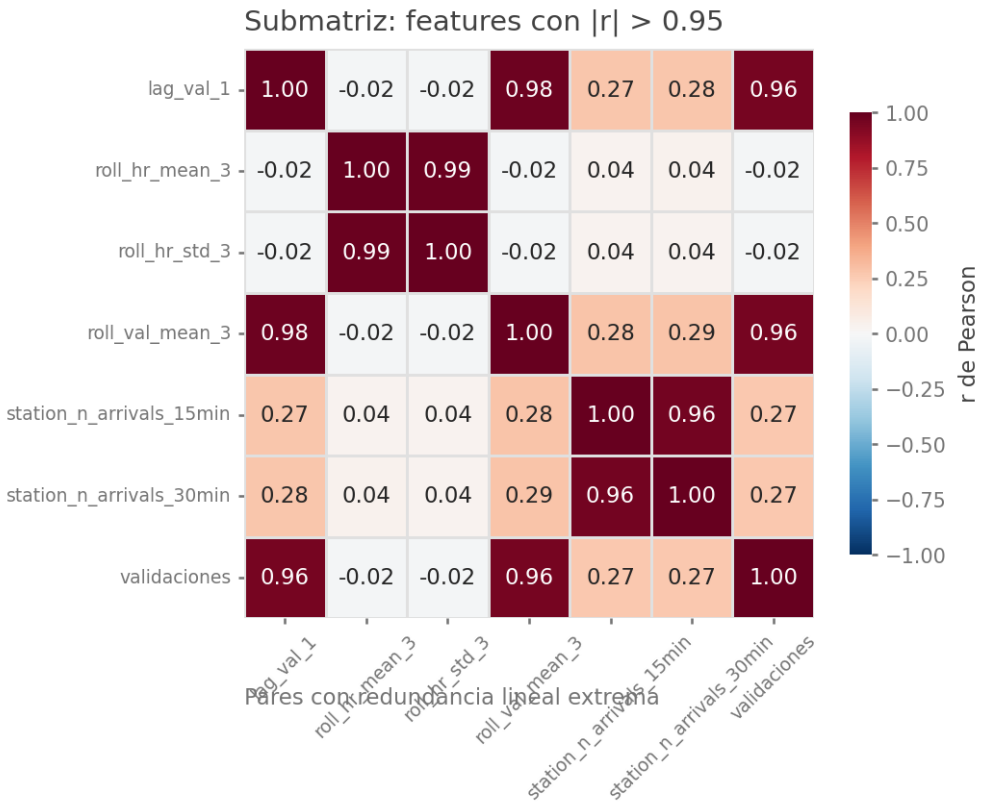
5.2.8 Selección de variables

Sobre las 46 variables candidatas de la capa gold (24 de las familias base y derivadas, 10 *cross-station* y 12 *upstream service-level*) se aplicó en la fase de selección de variables un análisis diagnóstico de redundancia con dos criterios. El primero es un filtro de varianza con umbral 1e-4, que no descarta ninguna variable: incluso las de menor varianza (te_id_estacion ≈ 1,07e-4 y te_servicio ≈ 2,76e-4) lo superan, y las variables *upstream* presentan dispersiones acordes a su naturaleza (las tasas de *bunching upstream* del orden de 0,008 y los conteos de arribos en torno a 2,4). El segundo es un análisis de correlación de Pearson con umbral $|r| > 0,95$ sobre una muestra estratificada de dos millones de registros del conjunto de entrenamiento. Se identifican cinco pares redundantes concentrados en tres familias: las estadísticas móviles del *headway* (*roll_hr_mean_3 y roll_hr_std_3*, $|r| = 0,988$), las validaciones con sus derivados temporales (*validaciones, lag_val_1 y roll_val_mean_3*, con $|r|$ entre 0,956 y 0,982) y el número de arribos *cross-station* en ventanas de quince y treinta minutos (*station_n_arrivals_15min y station_n_arrivals_30min,* $|r| = 0,965$). Ningún par dentro de la familia *upstream* supera el umbral, lo que confirma que las doce variables (cuatro métricas para cada una de las tres estaciones aguas arriba) aportan información en gran medida independiente entre sí.

En cada par redundante se conserva la variable con mayor correlación absoluta con el objetivo *bunching_025*, lo que conduce al descarte de cuatro variables: roll_hr_std_3, roll_val_mean_3, validaciones y station_n_arrivals_15min. El conjunto consumido por *XGBoost* queda así en 42 variables predictoras más el objetivo, con la siguiente distribución: ocho de contexto, seis rezagos, tres *rolling* (w = 3), cuatro entre derivadas, acumulada e interacción, nueve *cross-station* y doce *upstream service-level*. Como indicio convergente con los resultados del capítulo de interpretabilidad, las tres variables *upstream*_bunching_rate_15min* son las que presentan mayor correlación lineal con el objetivo (|r| = 0,127 para upstream3, 0,105 para upstream2 y 0,064 para upstream1), las dos primeras por encima incluso del *target encoding* del servicio (|r| = 0,071); este indicio anticipa el papel central que estas variables tendrán en el ranking de importancia por *gain* del modelo final.

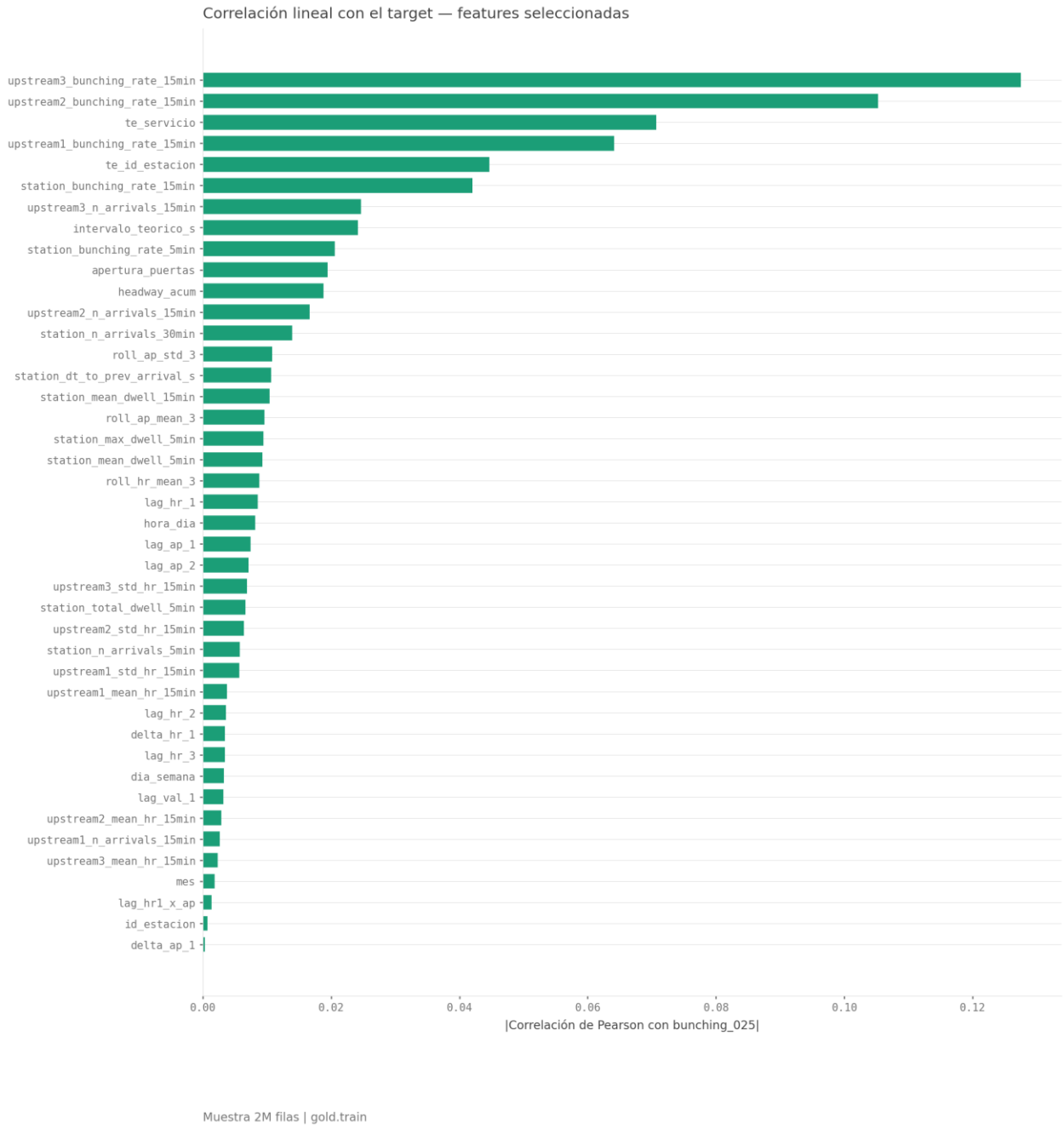
La Figura 26 presenta la matriz de correlación de Pearson restringida a las variables involucradas en al menos un par redundante, y la Figura 27 muestra la correlación lineal con el *target* en el conjunto final de 42 *features*.

Figura 26. Matriz de correlación de Pearson sobre los pares con |r| > 0,95



Fuente: Elaboración propia

Figura 27. Correlación lineal con el target - variables seleccionadas



Fuente: Elaboración propia

Una pregunta razonable sobre el tamaño del conjunto final es si las 42 variables resultan excesivas y exponen al modelo a la maldición de dimensionalidad. Este riesgo es relevante cuando concurren tres condiciones (el método depende de distancias o densidades en el espacio de variables, la muestra es pequeña respecto a la dimensionalidad, y las variables aportan

información mayormente independiente), ninguna de las cuales se cumple aquí. La relación entre el tamaño de la muestra de entrenamiento y el número de variables es la determinante: el modelo central se ajusta sobre 2,5 millones de observaciones, de modo que el cociente *N/d* se sitúa en torno a 60 000 (alrededor de 3 600 observaciones positivas por variable aun considerando sólo la clase minoritaria), órdenes de magnitud por encima de las heurísticas habituales para acotar el riesgo de sobreajuste por capacidad. A ello se añade que *XGBoost* no opera con distancias y selecciona variables de manera implícita en cada partición de árbol: las variables sin valor predictivo reciben *gain* cero y quedan sin uso, mientras que la regularización del modelo (*colsample_bytree, min_child_weight, reg_lambda y reg_alpha*) penaliza explícitamente apoyarse en señales débiles.

En el Anexo 1 presenta la descripción y atributos de cada una de las variables de la capa *gold*.

6. DESARROLLO Y EVALUACIÓN DE MODELOS PREDICTIVOS

Este capítulo presenta el proceso de modelado predictivo para la detección anticipada de *bus bunching* en la red troncal de TransMilenio. Antes de entrar en el desarrollo conviene advertir que el enfoque adoptado se aparta deliberadamente de las metodologías más comunes en la literatura del campo. Los trabajos de referencia (Yu et al.,[5]; Santos et al., [12]; Jiao et al., [6]) operan sobre fuentes que permiten seguir al bus individual a lo largo del corredor mediante un identificador de viaje, y en consecuencia predicen el *bunching* n paradas adelante del mismo vehículo. Las fuentes operativas de TransMilenio disponibles para este proyecto no contienen ese identificador, lo que obligó a reformular el problema en torno al n-ésimo arribo siguiente del mismo servicio a la misma estación. Esta limitación de los datos condiciona toda la estrategia de modelado (desde la unidad de predicción hasta la elección de las variables y la comparación con *benchmarks* publicados) y se asume desde el principio como una de las restricciones estructurales del trabajo. Con ese marco, el capítulo enuncia la formulación operativa del problema y la estrategia *multistep*, justifica la selección de los tres modelos evaluados a la luz de la literatura, describe la validación temporal y el tratamiento del desbalance de clases, y cierra con la tabla comparativa de resultados que sustenta la designación del modelo ganador para el capítulo de interpretabilidad y robustez.

6.1 Planteamiento del problema de modelado

El problema se aborda como una tarea de clasificación binaria sobre la variable *bunching_025*, definida como un indicador que toma el valor uno cuando el cociente entre el intervalo real de paso y el intervalo teórico programado, denominado *headway_ratio*, es inferior a 0,25. La unidad de predicción es el par (servicio, id_estacion): para cada arribo del servicio S a la estación E en el instante t, se busca anticipar si el n-ésimo arribo siguiente del mismo servicio a la misma estación presentará *bunching*. Los *lags* y los estadísticos de ventana que alimentan al modelo se calculan en consecuencia particionando por (servicio, *id_estacion*) y ordenando cronológicamente, lo que garantiza coherencia estricta entre la información disponible en t y el *target* a predecir.

Esta formulación difiere de la convencional en la literatura clásica del campo, que predice *bunching* n paradas adelante del mismo bus a lo largo del corredor. La diferencia obedece a dos razones. La primera es práctica: las fuentes operativas de *bronze* utilizadas en este trabajo no contienen un identificador de viaje confiable que permita seguir al vehículo individual de estación en estación. La segunda es metodológica: la dimensión que el centro de control de TransMilenio realmente monitorea en operación es justamente el comportamiento de cada ruta en un punto fijo del corredor, no la trayectoria de cada bus individual. Por estas razones la predicción se referencia al siguiente arribo en el mismo punto y no al siguiente paso del mismo vehículo. Una consecuencia es que las métricas reportadas no son directamente comparables con los *benchmarks* de la literatura en términos de *lead-time* absoluto, aunque sí lo son a igual horizonte n en métricas de clasificación como *F1* y *PR-AUC*.

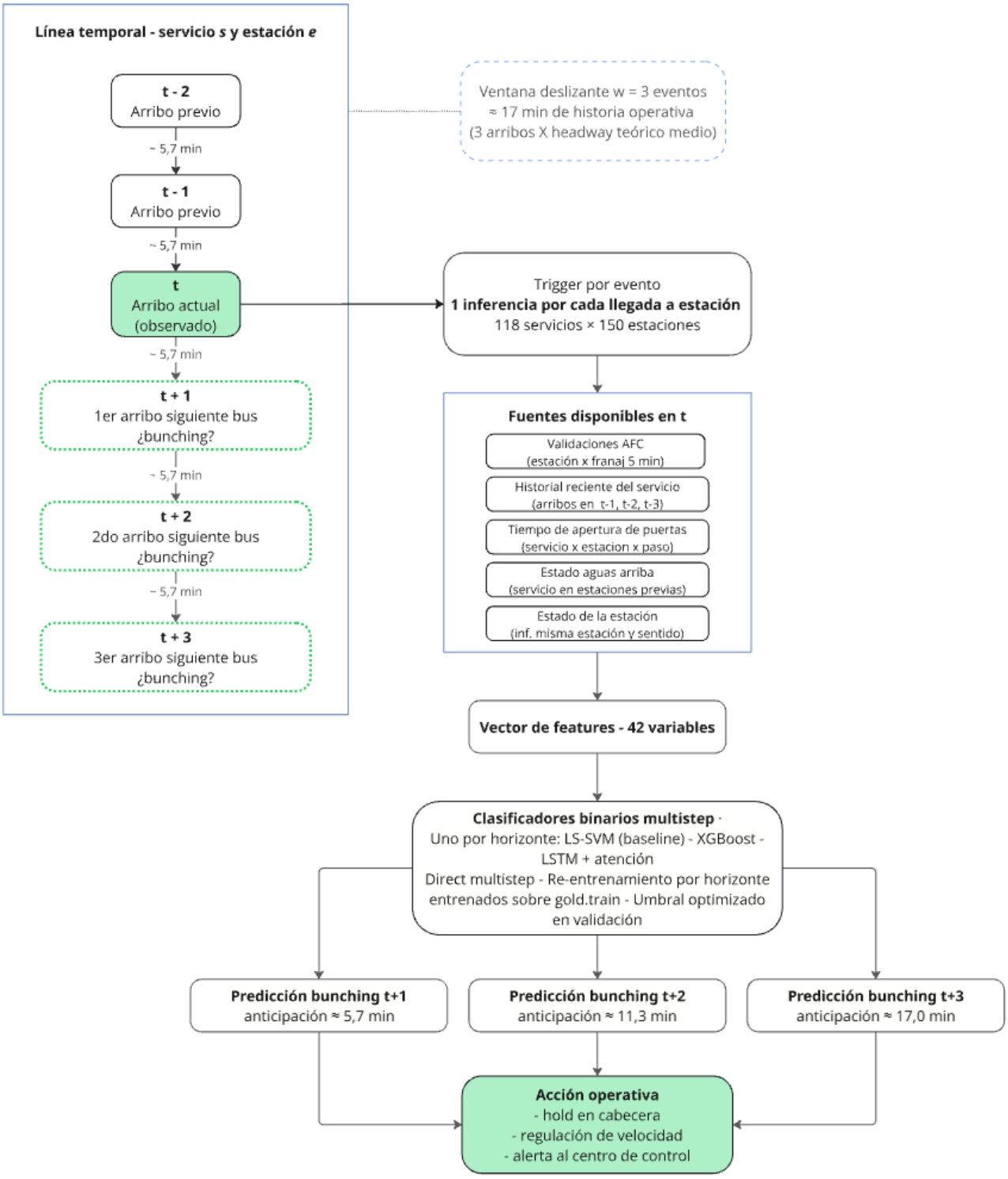
Para que la salida del modelo tenga interpretación operativa directa, los horizontes expresados en arribos se traducen a minutos mediante la mediana del intervalo entre arribos consecutivos del mismo servicio a la misma estación, que es de 5,67 min/arribo sobre el conjunto de entrenamiento. Esto sitúa los tres horizontes evaluados en t+1 ≈ 5,7 min, t+2 ≈ 11,3 min y t+3 ≈ 17,0 min de antelación efectiva. Estos lead-times son los que el centro de control tendría disponibles para aplicar acciones correctivas como holding o *skip-stop*.

Conviene precisar qué tipo de gestión habilita esa anticipación, aunque el diseño del controlador queda fuera del alcance de este trabajo. La literatura de control de *bus bunching* distingue varias palancas: la retención (*holding*) del bus de cabeza (el que circula adelante con el siguiente alcanzándolo) en una estación, que ensancha de nuevo el intervalo a costa de un retraso controlado para sus pasajeros; el salto de parada (*skip-stop*), en el que el bus rezagado omite paradas de baja demanda para recuperar marcha; el control de velocidad de crucero entre estaciones; el despacho de un bus de refuerzo cuando el hueco ya se ha vuelto grande; y la limitación del tiempo de abordaje en estación. Dado que el modelo de este proyecto identifica que la principal señal de un *bunching* inminente en una estación es que el mismo servicio viene llegando irregular en las estaciones inmediatamente anteriores (es decir, que la cascada ya está en curso aguas arriba), el enfoque de control coherente es preventivo y se ejerce aguas arriba: retener el bus de cabeza del par afectado en una estación anterior, antes de que el hueco colapse al llegar al punto observado, en lugar de reaccionar una vez que los dos buses ya circulan pegados. Ahora bien, el modelo no prescribe la acción. Por la formulación adoptada (que predice el comportamiento de la ruta en un punto fijo y no la trayectoria de un vehículo concreto, al no disponer de un identificador de viaje confiable) su salida es una alerta de riesgo por (servicio, estación), no una instrucción de retención sobre un bus específico; el operador cruzaría esa alerta con su sistema de localización en tiempo real para decidir qué bus retener, en qué estación anterior y por cuánto tiempo. El modelo opera, por tanto, como una capa de priorización y alerta temprana que precede al controlador, no como el controlador mismo. Integrarlo en un lazo cerrado (que reciba la predicción y devuelva la acción óptima de *holding* o *skip-stop*, optimizando directamente la regularidad del intervalo realmente observado) es la extensión natural y se retoma como línea de trabajo futuro en el capítulo de conclusiones.

Para predecir múltiples horizontes se adopta la estrategia Direct: tres modelos independientes, uno por horizonte, cada uno entrenado con el mismo vector de variables observado en *t* pero contra targets distintos construidos mediante el desplazamiento negativo de los registros para *n* = 1, 2 y 3. Ningún modelo usa la predicción del anterior como entrada, lo cual evita la propagación de errores que sufre la estrategia recursiva. El costo es entrenar tres veces, pero cada modelo aprende directamente la relación entre el estado en *t* y el *bunching* a *n* pasos.

La Figura 28 sintetiza visualmente esta formulación: la línea temporal de arribos consecutivos del servicio *s* a la estación *e*, la ventana deslizante de tres arribos previos que aporta el historial reciente, las fuentes disponibles en *t* y la salida *multistep* del modelo en los tres horizontes evaluados.

Figura 28. Esquema operativo del problema de predicción *multistep*



Fuente: Elaboración propia

6.2 Validación temporal y manejo del desbalance

La validación respeta la flecha del tiempo mediante una partición temporal estricta con fecha de corte el 1 de enero de 2025. El conjunto de entrenamiento abarca los datos previos a esa fecha (76,7 % de los registros) y el conjunto de prueba contiene los datos del periodo posterior (23,3 %). Esta partición es importante porque el *bunching* es un fenómeno con dinámicas temporales fuertes (patrones por hora del día, día de la semana y estacionalidad mensual) y una validación aleatoria habría permitido al modelo usar información futura para predecir el pasado, sobreestimando su desempeño. Todas las transformaciones del *pipeline* de preprocesamiento (imputación, *target encoding*, escalado, *winsorización*, *lags*, *rolling*) se ajustaron exclusivamente sobre el conjunto de entrenamiento y se aplicaron sin re-ajustar sobre el conjunto de prueba, garantizando ausencia de fuga de información. La validación cruzada del tipo *walk-forward* se descartó porque el volumen de datos disponible (~206 millones de registros en la ventana de Train) ofrece estabilidad estadística suficiente con un único *hold-out* temporal.

Dentro del conjunto de entrenamiento se reserva además un subconjunto de validación, obtenido mediante un muestreo estratificado por la clase en proporción 80/20. Este subconjunto se emplea para calibrar el umbral de decisión y, en los modelos que lo requieren, para la búsqueda de hiperparámetros, mientras que el conjunto de prueba no interviene en ninguna de esas decisiones y se reserva para la evaluación final. La distinción es importante: la partición *Train/Test* es temporal y respeta la flecha del tiempo, mientras que la separación interna entre ajuste y validación es un muestreo aleatorio estratificado dentro de *Train*, sin contacto con los datos de 2025.

La prevalencia de la clase positiva es de 6,13 % en *Train* y 6,19 % en *Test*, lo que configura un problema de clasificación con desbalance moderado-severo. Un clasificador que prediga siempre la clase mayoritaria alcanzaría una exactitud del 94 % sin aportar utilidad operativa. Para mitigar este desbalance se adoptan tres estrategias específicas por modelo: en *XGBoost* se utiliza el parámetro *scale_pos_weight*, que pondera la función de pérdida asignando mayor peso a los errores en la clase minoritaria; en *LSTM* se emplea la Focal Loss con parámetros $\gamma = 2{,}0$ y $\alpha = 0{,}25$, consistente con la configuración de Jiao et al. [6]; en *LS-SVM* se entrena directamente sobre la formulación con etiquetas en el conjunto {-1, +1} sin sobremuestreo explícito, asumiendo que la regularización del *kernel* atenúa parcialmente el sesgo. En los tres modelos se calibra adicionalmente el umbral de clasificación sobre el conjunto de validación para maximizar *F1*, en lugar de utilizar el umbral por defecto de 0,5.

6.3 Métricas de evaluación

La evaluación del desempeño se basa en métricas informativas bajo desbalance. Se descarta la exactitud como métrica principal por la razón recién señalada y se adoptan *PR-AUC* y *F1* como métricas primarias. *PR-AUC* mide el área bajo la curva *precision-recall* integrada sobre todos los umbrales y ofrece robustez estadística superior frente al desbalance, mientras que *F1* captura el

desempeño en el punto de operación efectivo (combina *precision* y *recall* en un único valor). La elección de *F1* como segunda métrica primaria responde a que en la operación del corredor ambos tipos de error tienen costo: no detectar un evento de *bunching* deja al centro de control sin oportunidad de aplicar una acción correctiva, mientras que emitir una falsa alarma activa intervenciones (retención, salto de parada, regulación de velocidad) que afectan al servicio sin necesidad. *F1*, al ser la media armónica de precisión y *recall*, penaliza simultáneamente ambos errores y por tanto refleja mejor que cualquiera de las dos por separado el costo operativo esperado del clasificador.

Para la designación del modelo ganador se adopta un score combinado igual a 0,6 × *PR-AUC* + 0,4 × *F1*, evaluado al horizonte t+1, con *PR-AUC* en solitario como *tie-breaker* en caso de empate. Este score no es una métrica estadística estándar sino una métrica compuesta definida específicamente para este proyecto como criterio de ranking entre modelos: combina las dos métricas primarias en un único valor escalar para facilitar la comparación cuando ninguna domina por separado. La ponderación 0,6/0,4 (a favor de *PR-AUC*) responde a las consideraciones que se detallan en el párrafo siguiente. Los resultados individuales de *PR-AUC* y *F1* se reportan también por separado para que la lectura del desempeño no dependa de esta agregación específica. Esta ponderación se justifica por la complementariedad de ambas métricas: *PR-AUC* mide la calidad del orden de las predicciones (independiente del umbral) y *F1* mide el desempeño en el punto de operación efectivo (combina *precision* y *recall* en un único valor).

La asimetría 0,6 / 0,4 a favor de *PR-AUC* obedece a tres razones. Primero, *PR-AUC* integra información de todos los umbrales de decisión, mientras que *F1* se calcula sólo en uno; contiene por tanto un superconjunto de la información de *F1*. Segundo, *F1* es susceptible a sobre-optimización del umbral sobre validación, lo que inflaría el score si esta métrica tuviera el mismo peso. Tercero, la elección 0,6 / 0,4 (en lugar de 0,5 / 0,5) refleja la complementariedad parcial de las dos métricas: *PR-AUC* y *F1* comparten una porción importante de información sobre la calidad del clasificador y, por tanto, ponderarlas a partes iguales sobreestimaría el peso de esa información compartida; la ponderación asimétrica reduce ese efecto sin renunciar a la lectura operativa que aporta *F1* en el punto de operación efectivo.

6.4 Modelos evaluados

La selección de modelos responde a tres criterios: la existencia de evidencia directa en la literatura de predicción de *bus bunching*, la cobertura de tres familias metodológicas distintas (*kernel methods, ensembles* de árboles y redes recurrentes con atención) y la disponibilidad de implementaciones maduras compatibles con el volumen de datos del proyecto. A continuación, se justifica el rol de cada modelo.

6.4.1 *LS-SVM*

El *Least Squares Support Vector* Machine con *kernel* RBF replica la metodología de Yu et al. [5], trabajo seminal en predicción de *bus bunching* con datos de Beijing. En este proyecto cumple un

doble papel: servir como referencia fiel al paper de origen y, a la vez, medir cuánto aporta el *feature engineering* a un método de *kernel*. Para ello se evalúa con dos conjuntos de variables: un Conjunto B reducido de seis variables tipo Yu-proxy[1] (*lag_hr_1, lag_hr_2, lag_ap_1, lag_val_1, roll_hr_mean_3, delta_hr_1*) que reproduce el espíritu de las variables originales, y un Conjunto A con el vector completo de 42 variables de la capa gold (las 21 variables base más las 9 *cross-station* y las 12 *upstream service-level* descritas para XGBoost). El desempeño del Conjunto A es el que se reporta en la tabla comparativa.

6.4.2 XGBoost

XGBoost (*Extreme Gradient Boosting*) se adopta como modelo central del proyecto siguiendo la trayectoria establecida por Santos et al. [12], que demostró su efectividad en datos operativos de Curitiba. Los modelos de *gradient boosting* sobre árboles de decisión han mostrado consistentemente buen desempeño en problemas tabulares con variables heterogéneas y desbalance de clases, y su capacidad para capturar interacciones no lineales y manejar codificaciones de alta cardinalidad lo hace especialmente apropiado para la combinación de rezagos, promedios móviles, derivadas, interacciones y variables agregadas presente en la capa *gold*. XGBoost consume el vector completo de la capa *gold*: 42 variables predictoras organizadas en tres familias. La primera son las 21 variables base (calendario, intervalo teórico programado, *dwell time*, *target encoding* de servicio y de estación, rezagos y promedio móvil del *headway* ratio y del *dwell time*, derivadas de primer orden, *headway* acumulado y la interacción lag_hr_1 × *apertura_puertas*) que sobreviven al filtro de varianza < 1e-4 y correlación |r| > 0,95. La segunda son 9 variables *cross-station*, que resumen la dinámica agregada de todos los servicios que pasan por la misma estación y en el mismo sentido de circulación en los últimos 5, 15 y 30 minutos. La tercera son 12 variables *upstream service-level*, que describen la regularidad del mismo servicio en las tres estaciones inmediatamente anteriores de su recorrido en los últimos 15 minutos. Se entrena con búsqueda de hiperparámetros mediante Optuna (50 *trials*), optimizando *PR-AUC* sobre validación con parada temprana sobre el mismo criterio.

6.4.3 *LSTM* con atención

La red *Long Short-Term Memory* bidireccional con mecanismo de atención de Bahdanau representa la familia de aprendizaje profundo y se incluye siguiendo a Jiao et al. [6], que reporta el mejor desempeño publicado para predicción de *bus bunching* con esta arquitectura. A diferencia de los modelos tabulares, que tratan cada observación como independiente, la *LSTM* recibe los datos como secuencias de los 10 arribos previos del mismo par servicio-estación, de modo que puede modelar internamente las dependencias temporales entre arribos sucesivos. Por esta razón consume la versión escalada de la capa *silver* y no la de la capa *gold*: los rezagos, los promedios móviles y demás derivadas temporales pre-computadas en *gold* serían redundantes con la información que la red ya extrae del propio orden de los pasos en cada secuencia, por lo que se descartan en favor del vector base. El conjunto excluye además el

[1] ***Yu-proxy***: conjunto de seis variables que reproduce, con las fuentes disponibles en TransMilenio, el vector de variables original de Yu et al. [5] (intervalo precedente, demanda de abordaje y tiempo de viaje).

identificador de estación, cuya identidad ya queda recogida en los *targets encodings* de servicio y estación y que, como entero de alta cardinalidad, introduciría ruido en una red neuronal; las demás variables se leen en su versión escalada porque las redes neuronales requieren entradas normalizadas. Por restricción computacional sólo se reportan los horizontes t+1 y t+3, que cubren los extremos del rango operativo de interés.

6.5 Proceso de entrenamiento

El flujo de entrenamiento común a los tres modelos se organiza en cuatro pasos. El primero es la lectura de la capa de datos correspondiente: la capa *gold* para los modelos tabulares (*XGBoost* y *LS-SVM*) y la capa *silver* para la red *LSTM*, que construye sus propias secuencias temporales a partir de las variables escaladas. El segundo es la separación interna del conjunto de entrenamiento (*Train*) en un subconjunto de ajuste y un subconjunto de validación, según la partición estratificada descrita en el numeral de validación temporal. El tercer paso es el ajuste del modelo sobre el subconjunto de ajuste, junto con la búsqueda de hiperparámetros que se detalla más adelante en los modelos que la requieren. El cuarto es la calibración del umbral de decisión, que se fija maximizando el *F1* sobre el subconjunto de validación. Tanto la búsqueda de hiperparámetros como la calibración del umbral se realizan únicamente sobre el subconjunto de validación, de modo que el conjunto de prueba permanece sin utilizar hasta la evaluación final.

La búsqueda de hiperparámetros se aborda de forma diferenciada por modelo en función de su costo computacional. Para *LS-SVM* y *XGBoost* se utiliza Optuna (*framework* de optimización bayesiana)[34] con 25 y 50 *trials* respectivamente, optimizando *PR-AUC* sobre validación con parada temprana sobre el mismo criterio: en *LS-SVM* se exploran los parámetros del *kernel* RBF (γ y σ) y en *XGBoost* se ajustan profundidad de árbol, tasa de aprendizaje, regularización L1/L2, peso por desbalance de clases y otros parámetros del *boosting*. El costo por *trial* es del orden de segundos a pocos minutos en ambos casos, lo que hace viable la búsqueda. Para el *LSTM* con atención se ejecuta también una búsqueda con Optuna, pero con un presupuesto reducido (cinco *trials* sobre tamaño de capa oculta, *dropout* y tasa de aprendizaje), dado que cada entrenamiento toma varios minutos en GPU; la configuración seleccionada (capa oculta = 96, dos capas bidireccionales, *dropout* ≈ 0,15, tasa de aprendizaje ≈ 1e-3, secuencias de longitud T = 10) se mantiene cercana a la arquitectura reportada por Jiao et al. [6].

Esta decisión obedece a tres razones convergentes. Primero, el costo: cada entrenamiento del *LSTM* toma 2–6 minutos por horizonte en GPU, y un *Optuna* razonable de 30 trials por tres horizontes implicaría del orden de varias horas a un día completo de cómputo continuo, frente a los segundos por *trial* de los modelos tabulares. Segundo, el espacio de búsqueda es significativamente mayor en redes neuronales (arquitectura, optimizador, regularización, configuración de secuencias) y un *Optuna* acotado apenas exploraría una fracción mínima de ese espacio. Tercero, las ablaciones manuales ejecutadas en el notebook (*LSTM* sin atención, *LSTM* unidireccional, GRU con atención) muestran diferencias de *PR-AUC* inferiores a 0,005 puntos entre variantes, lo que sugiere que el espacio cercano a la configuración base es plano y un Optuna

agresivo no aportaría margen significativo.

Las ventanas temporales de entrenamiento se ajustan por modelo según su costo computacional y su tiempo de entrenamiento, manteniendo en todos los casos el mismo punto de corte entre entrenamiento y prueba (1 de enero de 2025). *XGBoost*, por su escalabilidad sobre datos tabulares y su entrenamiento acelerado por histogramas, se entrena con la totalidad del *dataset* disponible (2022 a 2025), lo que le permite capturar mayor variabilidad estacional sin un costo excesivo. La red *LSTM* con atención se entrena sobre la ventana definida del 1 de julio de 2024 a 1 de enero de 2025, ya que el costo por época en GPU y la construcción de las secuencias hacen inviable extender el histórico completo. El *LS-SVM*, cuyo *solver* de *kernel RBF* escala de forma cuadrática en memoria y cúbica en cómputo con el número de muestras, se restringe a una ventana acotada de los tres meses, del 1 de julio al 11 de octubre de 2024, suficiente para ajustar el modelo de referencia sin exceder los límites de memoria. El conjunto de prueba es común a los tres modelos: el periodo posterior al corte (1 de enero a 31 de diciembre de 2025).

Tabla 16. Configuración experimental de los tres modelos

Aspecto	***LS-SVM***	**XGBoost**	***LSTM* + Atención**
Capa de datos	*gold (train_scaled)*	*gold (train)*	*silver (train_scaled)*
Ventana de entrenamiento	2024-07-01 a 2024-10-01 (3 meses)	2022-01-01 a 2025-01-01 (todo el histórico)	2024-07-01 a 2025-01-01 (6 meses)
N variables de entrada	42 (Conj. A) / 6 (Conj. B)	42 (21 base + 9 *cross-station* + 12 *upstream*)	41 (silver sin *id_estacion*); secuencias T=10
Hiperparámetros	Optuna (γ, σ) 25 *trials*	Optuna 50 *trials*	Bahdanau, hidden=96, 2 capas, *dropout* ≈ 0,15 (Optuna, 5 *trials*)
Horizontes evaluados	t+1, t+2, t+3	t+1, t+2, t+3	t+1, t+3 (restricción computacional)

Fuente: Elaboración propia

6.6 Resultados comparativos

La Tabla 17 presenta los resultados de los tres modelos sobre el conjunto de prueba al horizonte t+1. Cada modelo se ejecutó con un número reducido de repeticiones experimentales (semillas aleatorias distintas) para acotar la variabilidad estocástica. La tabla incluye además el score combinado 0,6·*PR-AUC* + 0,4·*F1* utilizado como criterio de ranking y la designación del modelo ganador.

Tabla 17. Desempeño en Test al horizonte t+1

Modelo	Rep.	Recall	Precisión	*F1*	*PR-AUC*	Score	Ranking
LS-SVM (Conj. A)	5	13,6 ± 0,7 %	23,7 ± 0,9 %	0,173 ± 0,007	0,155 ± 0,006	0,162	3
LSTM + Atención	3	45,8 ± 1,4 %	42,4 ± 1,3 %	0,440 ± 0,012	0,476 ± 0,015	0,462	2
XGBoost	5	58,4 ± 0,9 %	60,2 ± 0,8 %	0,593 ± 0,007	0,671 ± 0,008	0,640	1 (ganador)

Fuente: Elaboración propia

XGBoost domina las cuatro métricas evaluadas con un margen claro. Su *recall* del 58,4 % y precisión del 60,2 % indican que detecta aproximadamente seis de cada diez eventos reales de *bunching* y que, simultáneamente, seis de cada diez alertas emitidas corresponden a un evento real. El *F1* resultante (0,593) y el *PR-AUC* (0,671) lo posicionan claramente por encima de los demás modelos en el score combinado (0,640). La red *LSTM* con atención ocupa el segundo lugar (score 0,462), con un perfil balanceado entre *recall* (45,8 %) y precisión (42,4 %); su desempeño confirma la utilidad de la arquitectura recurrente para capturar dependencias temporales, pero queda por debajo de *XGBoost* en la separación entre clases. *LS-SVM* cierra el ranking (score 0,162) con valores significativamente menores: aun entrenado sobre el Conjunto A con el vector completo de 42 variables, y con una ventana de entrenamiento acotada a tres meses por sus límites de escalabilidad, el método de *kernel* no resulta competitivo frente a los otros dos modelos. Los valores reportados corresponden al promedio de varias repeticiones experimentales con semillas aleatorias; junto al promedio se reporta la desviación estándar, que en todos los casos se mantiene baja (entre 0,7 y 1,5 puntos porcentuales en *recall* y precisión, y entre 0,006 y 0,015 en *PR-AUC* y *F1*). La consistencia entre repeticiones indica que las diferencias entre modelos no son atribuibles a variabilidad estocástica del entrenamiento.

6.7 Selección del modelo ganador

La Tabla 18 extiende el análisis *multistep* para *XGBoost* reportando *PR-AUC*, *F1*, *recall* y precisión en los tres horizontes evaluados (*t+1, t+2 y t+3*).

Tabla 18. Desempeño multistep

Modelo	Horizonte	*Lead-time*	Repeticiones	*Recall*	Precisión	*F1*	*PR-AUC*
XGBoost	t+1	5,7 min	5	58,4 ± 0,9 %	60,2 ± 0,8 %	0,593 ± 0,007	0,671 ± 0,008
XGBoost	t+2	11,3 min	5	44,2 ± 1,3 %	46,3 ± 1,2 %	0,452 ± 0,011	0,505 ± 0,012
XGBoost	t+3	17,0 min	5	28,5 ± 1,5 %	32,1 ± 1,4 %	0,302 ± 0,014	0,318 ± 0,015

Fuente: Elaboración propia

Tres observaciones se desprenden del análisis *multistep*. Primero, la degradación entre horizontes es marcada y se acentúa de manera notable en *t+3*: el *PR-AUC* desciende de 0,671 en t+1 a 0,505 en *t+2* (caída del 25 %) y luego a 0,318 en *t+3* (caída acumulada del 53 % respecto a *t+1*, equivalente a perder más de la mitad del poder discriminativo del modelo en sólo 17 minutos de anticipación). El *F1*, el *recall* y la precisión siguen trayectorias paralelas: en *t+3*, el modelo detecta apenas 28,5 % de los eventos reales y sólo el 32,1 % de las alertas que emite corresponde a un agrupamiento efectivo.

Segundo, la degradación es relativamente uniforme entre precisión y *recall* en cada horizonte, lo que indica que el modelo pierde poder discriminativo de forma simétrica y no se vuelve simplemente más conservador o agresivo a medida que el horizonte se aleja; el problema está en la información disponible en t, no en el punto de operación elegido.

Tercero, la magnitud de la caída entre *t+1* y *t+3* tiene una implicación operativa directa: el horizonte útil del modelo en términos de alertas individuales por servicio se concentra alrededor de *t+1* (cinco a seis minutos vista), donde precisión y *recall* se sostienen por encima del 55 %. En *t+2* (once minutos vista) el desempeño cae a un perfil intermedio que admite uso como indicador agregado de riesgo, pero no como sistema de alertas individuales fiable. En *t+3* (diecisiete minutos vista) la calidad del *ranking* se aproxima a la mitad del valor de *t+1* y el modelo deja de ser apto para alertas por arribo: sólo puede emplearse en agregaciones amplias (corredor × franja horaria de 15 a 30 minutos) o como referencia comparativa de la tendencia del corredor, no como insumo de decisiones específicas. La elección de *t+1* como horizonte de referencia para la selección del ganador queda por tanto reforzada por estos resultados, y el *lead-time* operativo realmente aprovechable del modelo es estrecho. Las desviaciones estándar se mantienen bajas en los tres horizontes (entre 0,008 y 0,015 en *PR-AUC* y *F1*, y entre 0,9 y 1,5 puntos porcentuales en *recall* y precisión), confirmando que la degradación observada no es atribuible a variabilidad estocástica del entrenamiento sino al alejamiento intrínseco del horizonte de predicción.

A partir del score combinado se designa *XGBoost* como modelo ganador con un margen amplio sobre *LSTM* con atención y *LS-SVM*. Esta decisión se sustenta tanto en su desempeño superior en las cuatro métricas como en su estabilidad a través de los horizontes *multistep* y entre repeticiones, manifestada en las desviaciones estándar bajas reportadas en las tablas 17 y 18.

El análisis del capítulo siguiente se enfoca exclusivamente en este modelo: se examinan las variables que dominan sus decisiones (interpretabilidad mediante *gain* y *SHAP*) y la estabilidad de su desempeño bajo perturbaciones de los datos de entrada (robustez ante ruido y eliminación de variables). *LSTM* y *LS-SVM* no se profundizan en ese análisis porque su desempeño en este contexto los descarta como candidatos para uso operativo.

7. INTERPRETABILIDAD DEL MODELO Y ROBUSTEZ

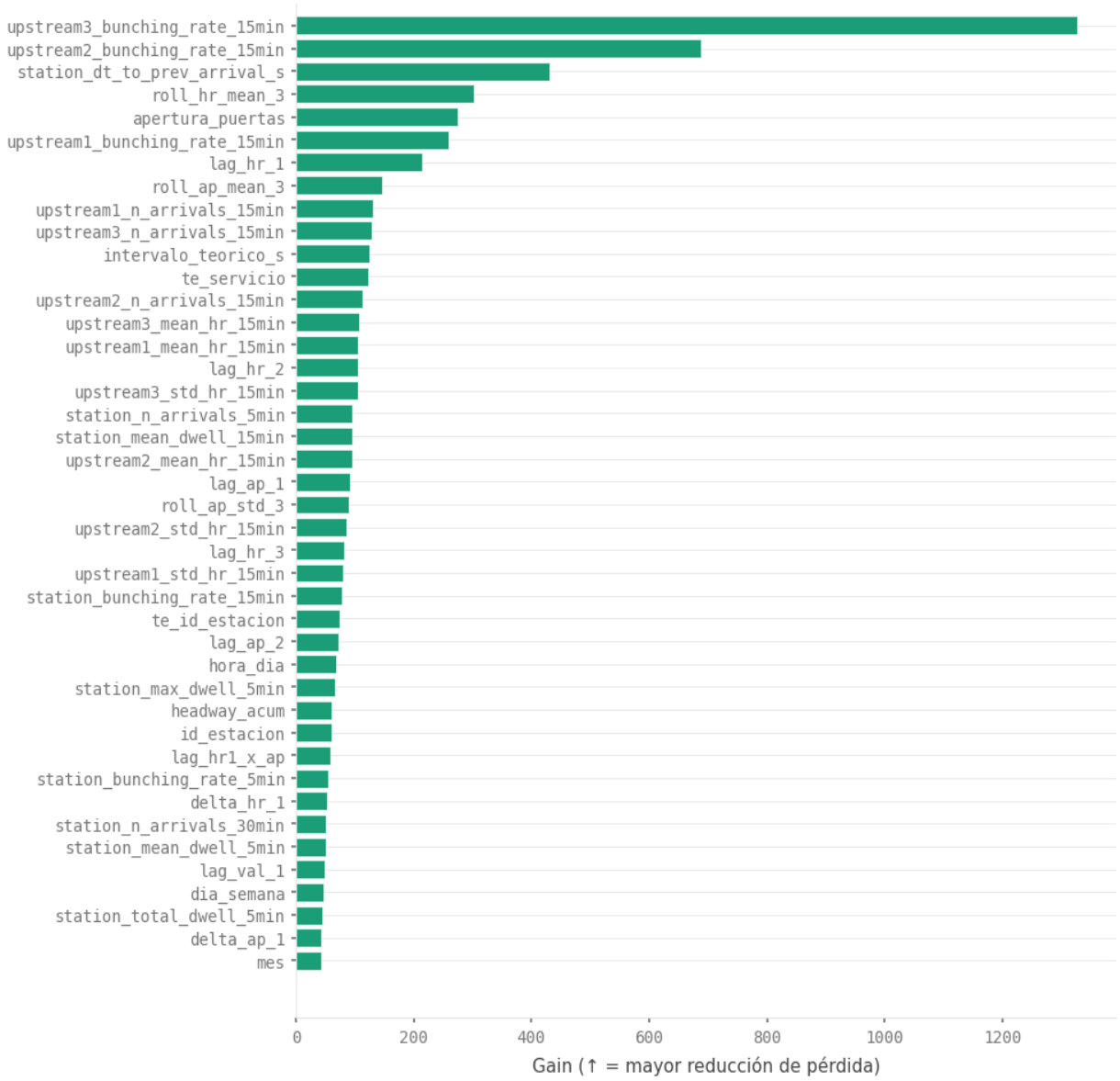
El capítulo anterior designó a *XGBoost* como modelo ganador a partir del *score* en *t+1*. Las métricas de desempeño, sin embargo, no son suficientes para considerar un modelo confiable en operación: una *PR-AUC* alta puede apoyarse en señales redundantes o colapsar ante pequeñas variaciones del entorno de medición. Este capítulo aborda dos preguntas complementarias sobre el modelo seleccionado: qué variables dominan sus decisiones (interpretabilidad) y cuánto se degrada su desempeño bajo perturbaciones de los datos de entrada (robustez).

7.1 Interpretabilidad

La interpretabilidad del modelo ganador se examina desde tres perspectivas complementarias. La primera utiliza el *gain*, métrica intrínseca de XGBoost que cuantifica la mejora promedio en la función de pérdida que aporta cada variable cuando es seleccionada como punto de corte en los árboles del ensamble. La segunda aplica *SHAP (SHAPley Additive exPlanations),* método post-hoc derivado de la teoría de juegos cooperativos que descompone cada predicción individual en contribuciones aditivas por variable, ofreciendo garantías de consistencia y aditividad ausentes en métricas heurísticas. Para árboles se utiliza la implementación *TreeSHAP*, computacionalmente eficiente y exacta. Agregando los valores absolutos sobre el conjunto de prueba se obtiene un ranking global con dirección del efecto. La tercera perspectiva analiza casos individuales seleccionados del conjunto de prueba para entender cómo el modelo combina las variables en predicciones específicas, con énfasis en los modos de fallo (falsos positivos y falsos negativos) que suelen revelar más sobre las limitaciones del modelo que los aciertos.

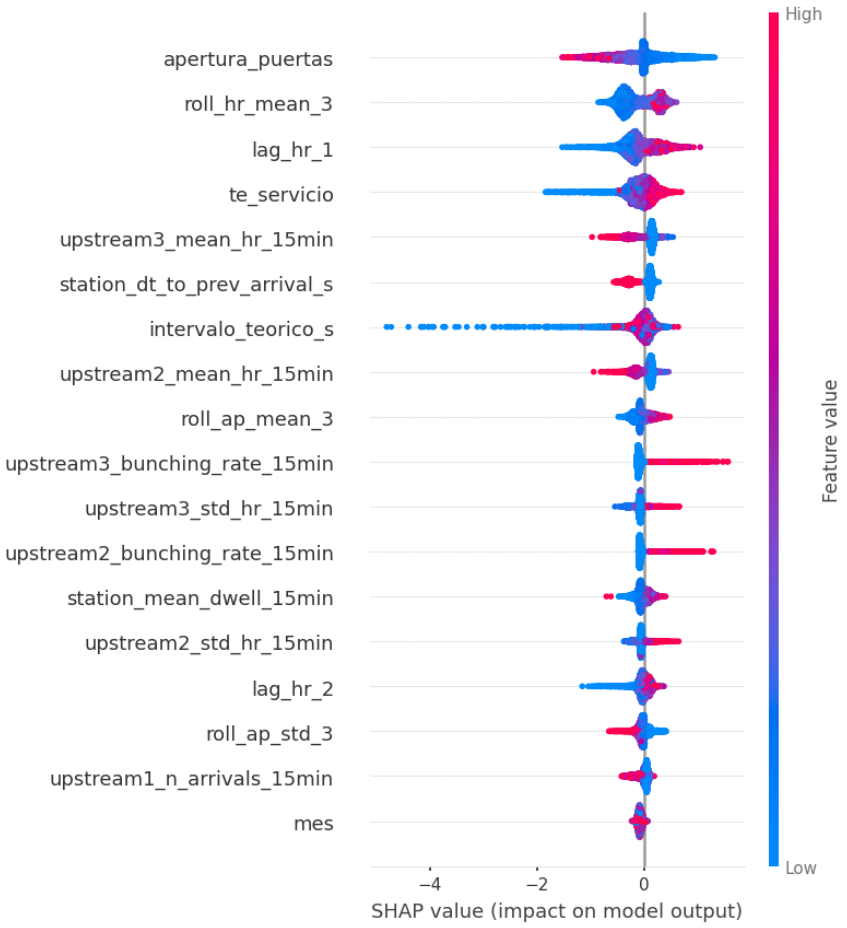
La Figura 29 muestra el ranking de las 42 variables ordenadas por *gain*. La Figura 30 presenta el ranking equivalente por *SHAP*, junto con su distribución de contribuciones *(beeswarm*) que revela tanto el aporte como la dirección del efecto: tonos cálidos corresponden a valores altos de la variable, tonos fríos a valores bajos.

Figura 29. Importancia por *gain* de las 42 variables



Fuente: Elaboración propia

Figura 30. *SHAP summary plot* - distribución de contribuciones por variable



Fuente: Elaboración propia

La Tabla 19 consolida el top 5 de cada técnica para evaluar el grado de acuerdo entre métodos. Una coincidencia alta entre *rankings* es indicación de que la importancia observada es atribuible al modelo, no al método de medición.

Tabla 19. Top 5 de variables según *gain*, *SHAP* y *drop*

Posición	**Gain**	***SHAP* (valor medio)**	**Drop (caída de *PR-AUC*)**
1	*upstream3_bunching_rate_15min*	*apertura_puertas*	*apertura_puertas*
2	*upstream2_bunching_rate_15min*	*roll_hr_mean_3*	*lag_hr_1*
3	*station_dt_to_prev_arrival_s*	*lag_hr_1*	*upstream3_mean_hr_15min*
4	*roll_hr_mean_3*	*te_servicio*	*upstream2_mean_hr_15min*
5	*apertura_puertas*	*upstream3_mean_hr_15min*	*intervalo_teorico_s*

Fuente: Elaboración propia

El acuerdo entre las tres técnicas es alto en torno a un núcleo común: *apertura_puertas* aparece entre las cinco primeras por *gain, SHAP y drop*, y la familia upstream domina el gain (encabezada por *upstream3_bunching_rate_15min*), mientras que *upstream3_mean_hr_15min* figura en el top 5 de *SHAP* y de *drop*. Las diferencias reflejan lo que mide cada métrica. El *gain*, métrica nativa del entrenamiento del árbol, premia a las variables que se usan tempranamente para particionar y resume ese efecto a lo largo de todos los árboles del modelo. *SHAP* cuantifica la contribución marginal de cada variable a la predicción individual y luego la agrega; el *drop* mide el impacto causal aproximado de retirar la variable. Cuando una variable concentra mucho *gain* pero su *SHAP* es comparable a otras, la lectura es que el modelo la aprovecha en pocos puntos de decisión con un efecto muy grande, no de manera diluida.

Para conectar este ranking con la interpretación operativa, las 42 variables se agrupan en cuatro categorías según su rol funcional. La dinámica local del *headway* reúne los rezagos, el promedio móvil, la derivada, el *headway* acumulado y la interacción del propio par servicio-estación. La demanda y operación del bus incluye la apertura de puertas y sus rezagos y derivadas, así como las validaciones rezagadas. El contexto temporal y espacial reúne las variables de calendario y los *target encodings* de servicio y estación. La cascada espacio-temporal agrupa las variables *cross-station* y *upstream service-level*, que describen la regularidad del corredor en la misma estación y aguas arriba del servicio.

Tabla 20. Distribución de importancia por categoría operativa

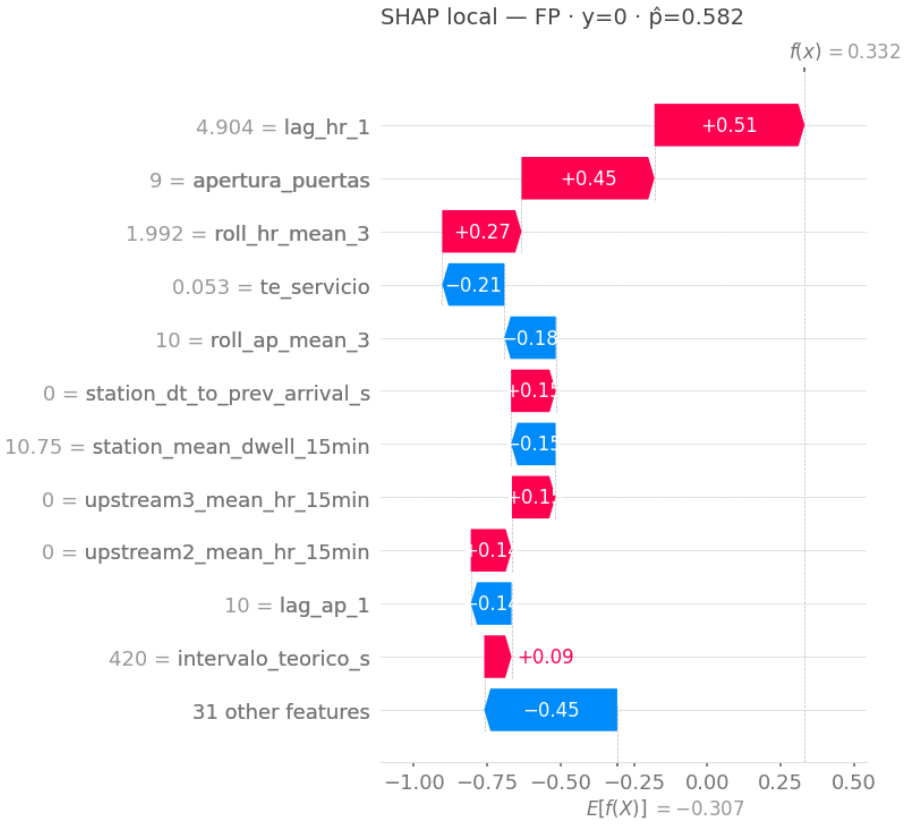
Categoría operativa	**N Variables**	**% de importancia (*SHAP*)**	**Top 5 (n)**
Cascada espacio-temporal (cross-station + upstream)	21	43,9 %	1
Dinámica local del *headway*	7	21,7 %	2
Demanda / operación local del bus	7	17,8 %	1
Contexto temporal y espacial	7	16,6 %	1

Fuente: Elaboración propia

La cascada espacio-temporal concentra el 43,9 % de la importancia agregada por *SHAP*, lo que la sitúa como la categoría con mayor peso total entre las cuatro definidas, por delante de la dinámica del *headway* (21,7 %), la demanda y operación del bus (17,8 %) y el contexto temporal y espacial (16,6 %). Este peso total debe interpretarse con cautela porque la categoría cascada agrupa veintiuna variables, frente a siete en cada una de las otras tres categorías; al ponderar por número de variables, la importancia media por variable es comparable entre cascada y dinámica local del *headway*, mientras que el contexto temporal y espacial mantiene una de sus siete variables en el top 5 (*te_servicio*), lo que indica una influencia individual no despreciable a pesar de su peso agregado menor. En el ranking de las cinco variables con mayor *SHAP* individual aparece sólo una de la cascada (*upstream3_mean_hr_15min*), pero su aporte agregado sigue siendo el mayor porque la familia distribuye importancia entre veintiuna variables. Los resultados sugieren que el modelo se apoya en el estado reciente del corredor más allá del par servicio-estación inmediato.

El análisis local sobre casos individuales del conjunto de prueba complementa la lectura global. La Figura 31 y Figura 32 presentan dos casos seleccionados: un falso positivo (alerta emitida sin que el *bunching* ocurriera) y un falso negativo (evento real no detectado). En cada figura se descomponen las contribuciones de las variables del caso particular, distinguiendo las que empujan la predicción hacia *bunching* (en rojo) de las que la alejan (en azul).

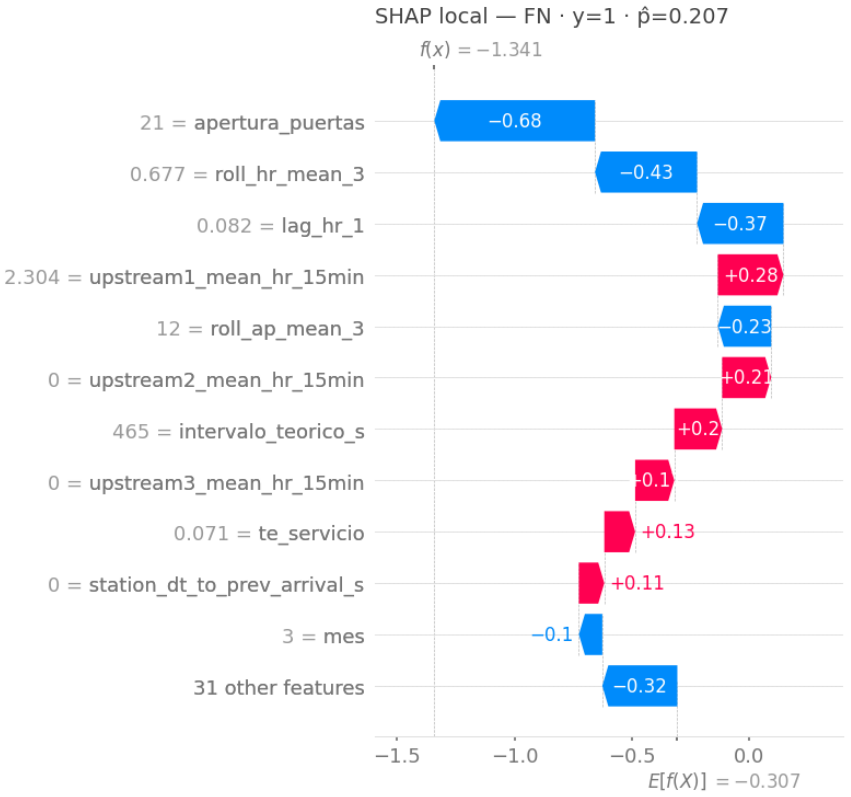
Figura 31. *SHAP* local - falso positivo



Fuente: Elaboración propia

En el caso del falso positivo, las contribuciones *SHAP* más altas hacia la predicción de *bunching* provienen de la dinámica reciente del propio servicio en la estación: el rezago inmediato del *headway*, el tiempo de apertura de puertas y el promedio móvil del *headway* en los últimos arribos. Las tres señalan una irregularidad operativa concentrada en el corto plazo, principalmente porque el arribo anterior llegó con un *headway* notablemente superior al programado y el promedio móvil confirmó esa desalineación. El modelo interpretó la combinación como antesala efectiva del agrupamiento, pero el arribo siguiente respetó el intervalo programado y la alerta resultó injustificada. La lectura plausible es que el modelo confunde una desalineación puntual del servicio precedente con la transición efectiva hacia *bunching*, en un contexto donde las variables *upstream* del corredor no apoyaron la predicción.

Figura 32. *SHAP* local - falso negativo



Fuente: Elaboración propia

En el caso del falso negativo, el cuadro *SHAP* es opuesto: las contribuciones más fuertes empujan la predicción hacia "no *bunching*", lideradas por el tiempo de apertura de puertas, el promedio móvil del *headway* y el rezago inmediato del *headway*. Las variables *upstream* sí aportaron señal positiva hacia *bunching*, en particular las del primer tramo aguas arriba, pero su magnitud fue insuficiente para compensar el peso de las variables locales. El evento terminó por materializarse a partir de una perturbación no recogida en la dinámica observada, probablemente un incidente puntual o una condición externa (clima, evento masivo o congestión no recurrente). Este tipo de fallo es estructural respecto al alcance actual de los datos: la combinación de variables disponibles no puede anticipar perturbaciones que se originan fuera del patrón histórico de los servicios en el corredor.

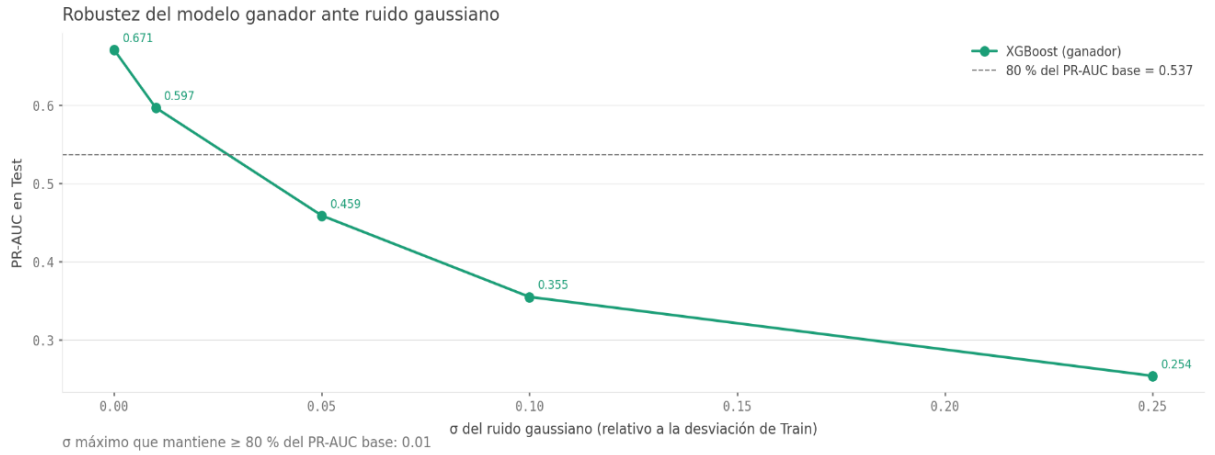
Los resultados son coherentes con la lectura global del modelo, pero ilustran una propiedad importante de los modelos basados en árboles: las variables con mayor *gain* no necesariamente dominan cada predicción individual, sino que su importancia se manifiesta en los casos donde sus valores discriminan eficazmente. La elección de los dos casos (FP y FN sin cascada activa) hace que el peso local recaiga en variables del segmento siguiente del *ranking*.

7.2 Robustez ante perturbaciones de los datos

La robustez se evalúa con tres pruebas de granularidad creciente. La primera mide la caída del *PR-AUC* al añadir ruido gaussiano de magnitud creciente sobre todas las variables simultáneamente (σ relativo a la desviación estándar de cada variable en Train). La segunda identifica qué variables individuales son más sensibles al ruido. La tercera estima el impacto causal aproximado de cada variable mediante su sustitución por la mediana del conjunto de prueba, equivalente a un escenario operativo donde la variable está temporalmente no disponible.

La Figura 33 muestra la curva *PR-AUC* frente a σ con la línea horizontal del 80 % del *PR-AUC* base, criterio adoptado para definir el nivel de ruido tolerable. La Tabla 21 reporta los valores numéricos.

Figura 33. Robustez ante ruido gaussiano global



Fuente: Elaboración propia

Tabla 21. *PR-AUC* en función del ruido gaussiano global

σ (relativo)	*PR-AUC*	Caída relativa
0,00	0,671	—
0,01	0,597	−11,0 %
0,05	0,459	−31,6 %
0,10	0,355	−47,1 %
0,25	0,254	−62,1 %

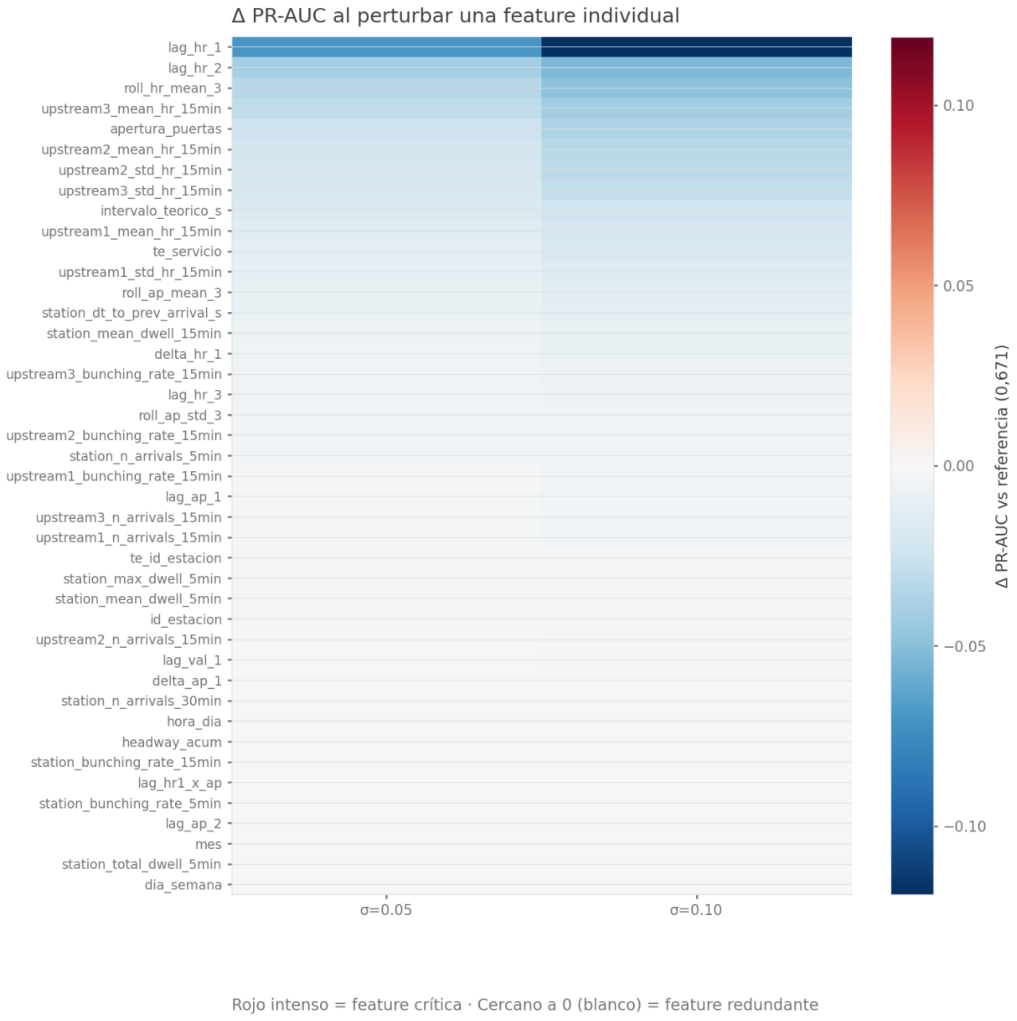
Fuente: Elaboración propia

El modelo se muestra muy sensible al ruido. El umbral del 80 % del *PR-AUC* base (equivalente a 0,537) se cruza entre $\sigma = 0{,}01$ y $\sigma = 0{,}05$: con perturbaciones del 1 % sobre la desviación estándar de cada variable la *PR-AUC* desciende a 0,597, todavía por encima del umbral, pero con perturbaciones del 5 % cae a 0,459, por debajo del umbral del 80 %. La degradación continúa con

perturbaciones mayores: $\sigma = 0{,}10$ reduce la *PR-AUC* a 0,355 (−47 %) y $\sigma = 0{,}25$ la lleva a 0,254 (−62 %); aunque sigue por encima del clasificador aleatorio (*PR-AUC* ≈ 0,06), representa una pérdida sustancial del poder discriminativo. La sensibilidad observada implica que las mediciones de las variables de entrada deben tener una precisión inferior al 1 % de su rango natural para que el modelo conserve un desempeño cercano al óptimo.

La Figura 34 desagrega la sensibilidad por variable individual: para cada variable se mide la caída de *PR-AUC* al añadir $\sigma = 0{,}10$ sólo sobre esa columna, dejando intactas las demás. Las variables cuya perturbación más degrada el desempeño coinciden en gran medida con las identificadas como más importantes por *gain* y *SHAP*, lo que refuerza la coherencia del modelo: las variables de las que más depende son también las que más impactan al perturbarse, sin redundancia significativa entre ellas.

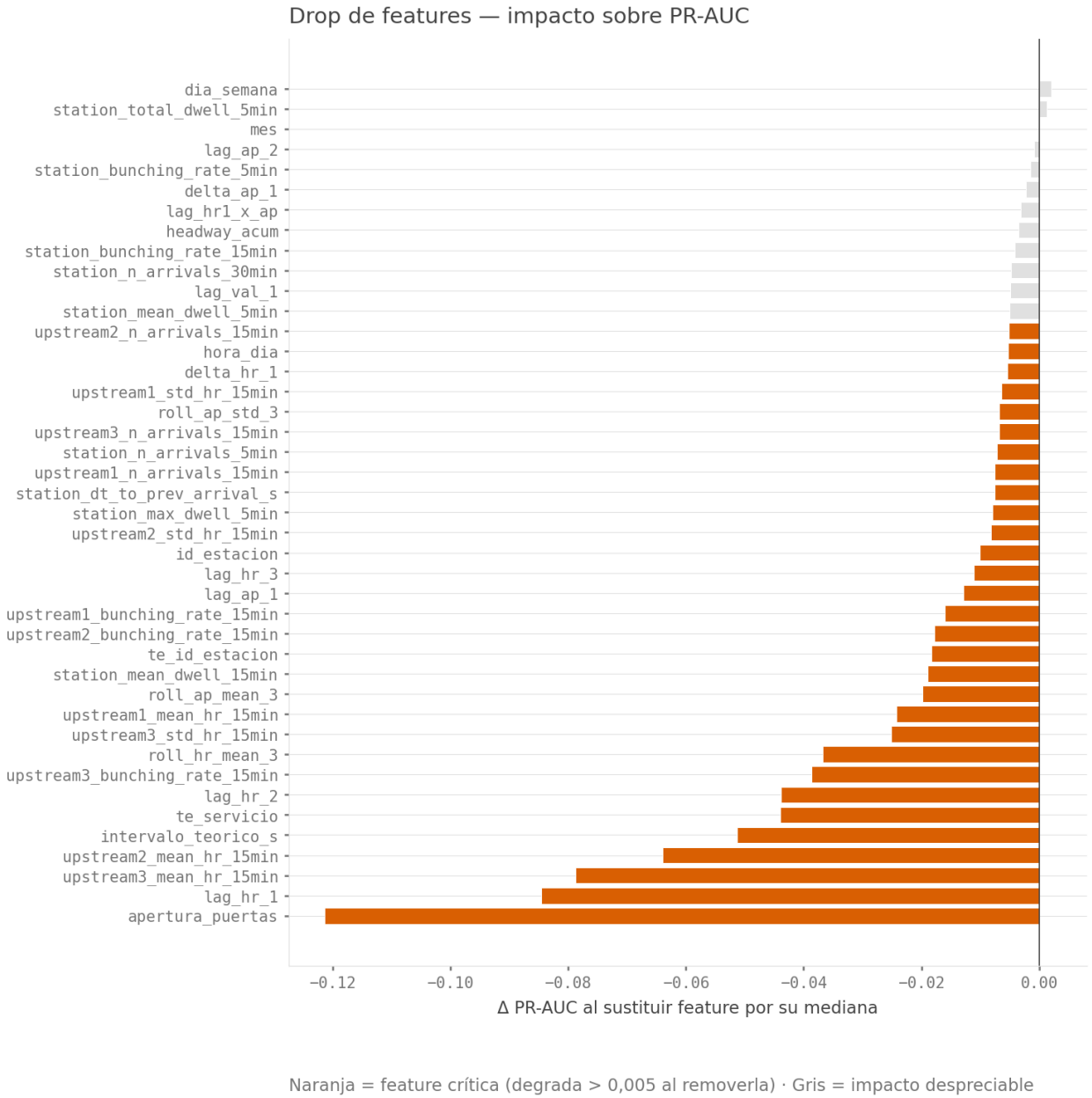
Figura 34. Sensibilidad por variable individual (σ = 0,10)



Fuente: Elaboración propia

La Figura 35 presenta el resultado del experimento de *drop*: para cada variable se sustituye su valor por la mediana del conjunto de prueba (eliminando así su contribución informativa) y se recalcula el *PR-AUC*. La caída resultante mide el impacto causal aproximado de remover cada variable. La Tabla 22 lista las cinco variables cuya remoción genera la mayor degradación.

Figura 35. Impacto de la sustitución de cada variable por su mediana



Fuente: Elaboración propia

Tabla 22. Top 5 de variables cuya remoción degrada más el desempeño

Variable	***PR-AUC* tras *drop***	**Caída de *PR-AUC***
apertura_puertas	0,550	−0,121
lag_hr_1	0,586	−0,085
upstream3_mean_hr_15min	0,593	−0,078
upstream2_mean_hr_15min	0,606	−0,065
intervalo_teorico_s	0,620	−0,051

Fuente: Elaboración propia

La variable *apertura_puertas* se confirma como la más indispensable: su sustitución por la mediana reduce el *PR-AUC* en 0,121 puntos absolutos, cerca del 18 % del valor base (0,671 a 0,550). La siguen *lag_hr_1* (−0,085), *upstream3_mean_hr_15min* (−0,078) y *upstream2_mean_hr_15min* (−0,065), lo que confirma que la regularidad reciente del mismo servicio aguas arriba aporta información condicional difícil de sustituir por el resto del vector. *intervalo_teorico_s* (−0,051) cierra el top 5; *roll_hr_mean_3* y *te_servicio* quedan algo por debajo. En el extremo opuesto, las variables menos importantes (*dia_semana, mes, station_total_dwell_5min, lag_ap_2, station_bunching_rate_5min, lag_hr1_x_ap*) muestran una caída nula o despreciable al ser eliminadas, lo que indica que el modelo podría operar con un subconjunto reducido sin pérdida apreciable y abre la posibilidad de un ejercicio futuro de simplificación para reducir la complejidad del despliegue.

8. CONCLUSIONES Y TRABAJOS FUTUROS

Este capítulo cierra el documento con la síntesis de los hallazgos del proyecto y la proyección de las líneas que el trabajo deja abiertas.

8.1 Conclusiones

8.1.1 Aportes y hallazgos principales

Los resultados confirman que existe señal predictiva real para anticipar el *bus bunching* en horizontes cortos. Al horizonte *t+1* (de cinco a seis minutos vista) el modelo ganador alcanza un *PR-AUC* de 0,671 y un *F1* de 0,593, con *recall* del 58,4 % y precisión del 60,2 %. El desempeño se degrada de forma marcada al alejarse el horizonte: a *t+3* (cerca de 17 minutos vista) el *PR-AUC* desciende a 0,318 y pierde casi la mitad del poder discriminativo. El modelo conserva al menos el 80 % del *PR-AUC* base solo frente a perturbaciones leves, del orden del 1 % de la desviación de cada variable (con un ruido del 5 % cae por debajo de ese umbral), y concentra su capacidad predictiva en un grupo reducido de variables encabezado por la apertura de puertas, que por sí sola explica cerca del 18 % del *PR-AUC* base.

El análisis de interpretabilidad confirma que el modelo es coherente con la teoría y no una caja negra: la cascada *espacio-temporal (cross-station y upstream)* encabeza el *gain* y *apertura_puertas* domina por *SHAP* y por sustitución (*drop* de −0,121, cerca del 18 % del *PR-AUC* base). La evaluación de robustez, en cambio, revela una alta sensibilidad al ruido (el desempeño cae bajo el 80 % del valor base con solo un 5 % de perturbación), lo que condiciona su uso operativo a fuentes de datos de alta precisión. En síntesis, el modelo es interpretable y plausible, pero confiable solo bajo una medición fiel de sus entradas.

Como aporte al estado del arte, el proyecto deja un elemento diferencial: cuando las fuentes operativas no permiten seguir al bus individual, el problema puede reformularse en torno al *n-ésimo* arribo siguiente del mismo servicio a la misma estación, alternativa documentada para investigadores que enfrenten la misma limitación de datos.

Esta reformulación explica también la distancia frente a los desempeños reportados en la literatura internacional, donde varios trabajos disponen de identificador de viaje y ventanas de muestreo más finas y alcanzan métricas superiores. El aporte de este trabajo no reside en superar esos valores, sino en acotar el problema bajo restricciones de datos y cuantificar la cota de desempeño alcanzable sin seguimiento del bus individual, lo que ofrece una referencia realista para sistemas con fuentes operativas equivalentes.

8.1.2 Implicaciones operativas para TransMilenio

Aunque el desempeño observado en *t+1* (precisión 60,2 %, *recall* 58,4 %) deja todavía margen de

mejora antes de un uso operativo directo como sistema de alertas individuales por servicio, las cifras habilitan tres caminos prácticos.

Operación con umbral calibrado a precisión. Ajustar el umbral de decisión para privilegiar la precisión, sacrificando *recall,* reduce las falsas alarmas a un nivel manejable. La elección entre privilegiar precisión (menos falsas alarmas, mayor confianza por alerta) o *recall* (más detecciones, más falsas alarmas) depende del costo relativo de cada error: en TransMilenio una acción correctiva implica retención o ajuste de velocidad sobre un servicio, de modo que una falsa alarma deteriora el servicio percibido por los usuarios del bus intervenido, mientras que omitir un evento real deja sin acción al centro de control aunque el operador pueda detectarlo por otros canales. Esta asimetría sugiere, como balance inicial, privilegiar precisión sin renunciar a un piso mínimo de *recall* útil, con el umbral concreto ajustado mediante consulta con el operador y monitoreo en operación piloto.

Indicadores agregados de regularidad. Agregar las alertas en ventanas amplias de corredor y franja horaria (15 a 30 minutos) permite construir indicadores de riesgo que alimenten tableros operativos sin exigir decisiones automáticas. Esta lectura agregada es la única que sigue siendo informativa en horizontes intermedios y largos, dado que el *PR-AUC* en *t+3* cae cerca de la mitad del valor de *t+1* y deja de soportar alertas individuales por servicio.

Integración en control predictivo de horizonte corto. Incorporar el modelo en flujos de control predictivo restringidos al horizonte corto (*t+1* y, con cautela, *t+2*), dejando la decisión final al operador.

8.1.3 Lecciones metodológicas

El desarrollo del proyecto deja tres lecciones aplicables a trabajos similares.

Inversión temprana en arquitectura de datos. La consolidación en *DuckDB* con esquema medallón permitió iterar sobre el *feature engineering* sin re-ejecutar la ingesta, algo insustituible en un trabajo de grado individual con recursos limitados, y demostró ser pertinente para volúmenes del orden de cientos de millones de filas sin acceso a infraestructura distribuida.

Valor de mantener varios modelos en paralelo. Conservar tres modelos en lugar de optimizar uno solo aportó información que no podría obtenerse del mejor modelo en aislamiento, en particular sobre el valor del *feature engineering* pre-computado frente a la inferencia secuencial.

Disciplina estricta de no filtración de información. Ajustar todas las transformaciones exclusivamente sobre el conjunto de entrenamiento y verificarlo mediante chequeos automatizados elevó el costo de implementación, pero garantizó la validez de los resultados de evaluación.

8.1.4 Limitaciones del estudio

El alcance del estudio presenta restricciones que condicionan los resultados.

Ausencia de seguimiento del bus individual. La formulación no sigue al bus a lo largo de su recorrido: al no disponer de un identificador de viaje en las fuentes, el modelo predice el comportamiento de un servicio en una estación fija y no el de un vehículo específico a lo largo del corredor. Es la restricción que más condiciona el desempeño alcanzable.

Dependencia de la calidad de las fuentes. El proyecto depende íntegramente de la calidad de las fuentes operativas suministradas por TransMilenio: errores sistemáticos o cambios no documentados en los protocolos de captura podrían afectar la validez de los resultados sin que el modelado los detecte.

Costo computacional y heterogeneidad de las ventanas de entrenamiento. La distinta escalabilidad de los tres modelos obligó a entrenarlos sobre ventanas temporales de tamaño desigual: *XGBoost* aprovechó la totalidad del histórico (2022 a 2025), mientras que la *LSTM* se limitó a la ventana de seis meses y el *LS-SVM* a una ventana de tres meses, por el costo por época en GPU y por el escalamiento cuadrático en memoria y cúbico en cómputo del *kernel*, respectivamente. Esta asimetría limita la comparabilidad estricta entre modelos, ya que parte de las diferencias de desempeño puede atribuirse al volumen de datos de entrenamiento y no solo a la capacidad intrínseca de cada algoritmo.

8.2 Trabajos futuros

El trabajo deja una base metodológica sobre la cual pueden apoyarse desarrollos posteriores, organizados en tres frentes: líneas prioritarias, extensiones metodológicas y validación operativa.

8.2.1 Líneas prioritarias

Seguimiento del bus individual entre estaciones. La limitación más importante de los modelos evaluados es su ceguera al recorrido del bus individual, consecuencia de que las fuentes no contienen un identificador de viaje. Un trabajo futuro que reconstruya ese identificador desde las fuentes operativas (por ejemplo, mediante reglas heurísticas que enlacen arribos consecutivos del mismo servicio en estaciones contiguas) habilitaría predecir el *bunching* de un bus específico en sus próximas estaciones, formulación que la literatura ha demostrado más predictiva y que permitiría comparación numérica directa con los *benchmarks* publicados.

Mejoras en la captura de datos del sistema operativo. Mejorar la granularidad y precisión de los datos disponibles es una línea de largo plazo: el conteo de pasajeros es hoy un dato agregado por estación y franja de cinco minutos, no por viaje ni por servicio puntual. Dado que la apertura de puertas demostró funcionar como *proxy* efectivo de demanda específica del bus, precisar la dinámica de pasajeros por arribo (tiempo de embarque, descensos, ocupación instantánea)

tendría impacto directo en la calidad del modelo.

8.2.2 Extensiones metodológicas

Estado simultáneo multi-servicio. Extender la formulación al estado simultáneo de otros servicios que comparten el corredor reconocería el carácter sistémico del *bunching*.

Redes neuronales sobre grafos. Aplicar redes neuronales sobre grafos (GNN) al grafo del corredor, donde los nodos representan estaciones y las aristas codifican adyacencias u otros vínculos operativos, permitiría capturar de forma explícita las interacciones entre estaciones, una dimensión con propagación espacial que las arquitecturas evaluadas no aprovechan.

Fuentes externas de contexto. Incorporar datos meteorológicos, calendario de eventos sociales (como manifestaciones) e incidentes de tráfico capturaría perturbaciones que las fuentes operativas internas no observan.

Arquitecturas y modelos adicionales. Evaluar familias adicionales (arquitecturas *Transformer* para series temporales y ensambles heterogéneos) bajo el mismo esquema de evaluación permitiría medir el aporte real de cada alternativa.

8.2.3 Validación y despliegue operativo

Antes de cualquier adopción operativa, el modelo requiere una fase de validación en condiciones reales que el alcance de este trabajo no abordó.

Análisis contrafactual. El estudio no incorpora información sobre las decisiones operativas efectivamente aplicadas por el centro de control durante los eventos observados, lo que impide estimar el valor agregado del modelo frente a las prácticas actuales. Construir ese contraste es condición necesaria para justificar su despliegue.

Piloto operativo acotado. Validar el modelo en una troncal o corredor delimitado, con monitoreo del operador, permitiría calibrar el umbral de decisión y medir su comportamiento fuera del entorno de evaluación retrospectiva.

Evaluación costo-beneficio. Cuantificar el costo de las falsas alarmas (retenciones o ajustes innecesarios) frente al beneficio de los eventos correctamente anticipados cerraría el análisis sobre la conveniencia de accionar decisiones a partir del modelo.

9. REFERENCIAS BIBLIOGRÁFICAS

[1] «Estadísticas de oferta y demanda del Sistema Integrado de Transporte Público - SITP - diciembre 2025». Accedido: 26 de abril de 2026. [En línea]. Disponible en: https://www.transmilenio.gov.co/comunicaciones/publicaciones/2025/estadisticas-oferta-demanda-sistema-integrado-transporte-publico-sitp-diciembre-2025

[2] F. Martínez-Plumed *et al.*, «CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories», *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, n.º 8, pp. 3048-3061, ago. 2021, doi: 10.1109/TKDE.2019.2962680.

[3] «Informe Ejecutivo Encuesta de Satisfacción 2025». Accedido: 26 de abril de 2026. [En línea]. Disponible en: https://www.transmilenio.gov.co/comunicaciones/publicaciones/2025/informe-ejecutivo-encuesta-de-satisfaccion-2025

[4] L. Moreira-Matias, O. Cats, J. Gama, J. Mendes-Moreira, y J. F. de Sousa, «An online learning approach to eliminate Bus Bunching in real-time», *Applied Soft Computing*, vol. 47, pp. 460-482, oct. 2016, doi: 10.1016/j.asoc.2016.06.031.

[5] H. Yu, D. Chen, Z. Wu, X. Ma, y Y. Wang, «Headway-based bus bunching prediction using transit smart card data», *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 45-59, nov. 2016, doi: 10.1016/j.trc.2016.09.007.

[6] J. Jiao, P. Shen, y Y. Zhang, «Headway-Based Bus Bunching Prediction Using *LSTM* with Attention», en *2023 IEEE 8th International Conference on Intelligent Transportation Engineering (ICITE)*, oct. 2023, pp. 451-458. doi: 10.1109/ICITE59717.2023.10733869.

[7] M. Rezazada, N. Nassir, E. Tanin, y A. (Avi) Ceder, «Bus bunching: a comprehensive review from demand, supply, and decision-making perspectives», *Transport Reviews*, vol. 44, n.º 4, pp. 766-790, jul. 2024, doi: 10.1080/01441647.2024.2313969.

[8] C. F. Daganzo, «A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons», *Transportation Research Part B: Methodological*, vol. 43, n.º 10, pp. 913-921, dic. 2009, doi: 10.1016/j.trb.2009.04.002.

[9] D. P. Naranjo Valero, «Análisis del funcionamiento operacional de buses en vagones de BRT a partir de big data. Caso de estudio : Bogotá, troncal de Transmilenio Calle 80», 2018, doi: 10.71590/1992/34824.

[10] C. Peña y E. Moreno, «Delay at Bus Stops of Transmilenio Transport System According to Parameters Measured" in situ". Case Study Bogotá-Colombia», *Procedia - Social and Behavioral Sciences*, vol. 160, pp. 121-129, dic. 2014, doi: 10.1016/j.sbspro.2014.12.123.

[11] Z. Gong, B. Du, Z. Liu, W. Zeng, P. Perez, y K. Wu, «SD-seq2seq : A Deep Learning Model for Bus Bunching Prediction Based on Smart Card Data», en *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, ago. 2020, pp. 1-9. doi: 10.1109/ICCCN49398.2020.9209686.

[12] V. Santos, C. Santos Pires, D. Nascimento, y A. Queiroz, «A Decision Tree Ensemble Model for Predicting Bus Bunching», *The Computer Journal*, vol. 65, may 2021, doi: 10.1093/comjnl/bxab045.

[13] «On the tradeoff between sensitivity and specificity in bus bunching prediction», *Journal of Intelligent Transportation Systems*, vol. 25, n.º 4, pp. 384-400, ene. 2021, doi: 10.1080/15472450.2020.1725887.
[14] M. Andres y R. Nair, «A predictive-control framework to address bus bunching», *Transportation Research Part B: Methodological*, vol. 104, pp. 123-148, oct. 2017, doi: 10.1016/j.trb.2017.06.013.
[15] C. Cortes y V. Vapnik, «Support-vector networks», *Mach Learn*, vol. 20, n.º 3, pp. 273-297, sep. 1995, doi: 10.1007/BF00994018.
[16] L. Breiman, «Random Forests», *Machine Learning*, vol. 45, n.º 1, pp. 5-32, oct. 2001, doi: 10.1023/A:1010933404324.
[17] J. H. Friedman, «Greedy function approximation: A gradient boosting machine.», *The Annals of Statistics*, vol. 29, n.º 5, pp. 1189-1232, oct. 2001, doi: 10.1214/aos/1013203451.
[18] 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*, en KDD '16. New York, NY, USA: Association for Computing Machinery, ago. 2016, pp. 785-794. doi: 10.1145/2939672.2939785.
[19] S. Hochreiter y J. Schmidhuber, «Long Short-Term Memory», *Neural Computation*, vol. 9, pp. 1735-1780, nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[20] D. Bahdanau, K. Cho, y Y. Bengio, «Neural Machine Translation by Jointly Learning to Align and Translate», 19 de mayo de 2016, *arXiv*: arXiv:1409.0473. doi: 10.48550/arXiv.1409.0473.
[21] I. Sutskever, O. Vinyals, y Q. V. Le, «Sequence to Sequence Learning with Neural Networks», 14 de diciembre de 2014, *arXiv*: arXiv:1409.3215. doi: 10.48550/arXiv.1409.3215.
[22] Y. Deng, X. Luo, X. Hu, Y. Ma, y R. Ma, «Modeling and Prediction of Bus Operation States for Bunching Analysis», *Journal of Transportation Engineering, Part A: Systems*, vol. 146, n.º 9, p. 04020106, sep. 2020, doi: 10.1061/JTEPBS.0000436.
[23] A. Kakarla, V. S. K. R. Munagala, T. Ishizaka, A. Fukuda, y S. Jana, «Travel Time Prediction and Route Performance Analysis in BRTS based on Sparse GPS Data», en *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, abr. 2021, pp. 1-5. doi: 10.1109/VTC2021-Spring51267.2021.9448832.
[24] Y. Matseliukh, V. Lytvyn, Z. Hu, y M. Bublyk, «Predictive Modelling and Factor Analysis of Public Transport Delays in Smart City Using Interpretable Machine Learning», *International Journal of Information Technology and Computer Science*, vol. 17, n.º 6, p. 1, Accedido: 26 de abril de 2026. [En línea]. Disponible en: https://www.mecs-press.org/ijitcs/ijitcs-v17-n6/v17n6-1.html
[25] «Interpretable Machine Learning». Accedido: 26 de abril de 2026. [En línea]. Disponible en: https://christophm.github.io/interpretable-ml-book/
[26] M. Armbrust, A. Ghodsi, R. Xin, y M. Zaharia, «Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics», 2021.
[27] «What is Medallion Architecture?», Databricks. Accedido: 26 de abril de 2026. [En línea]. Disponible en: https://www.databricks.com/blog/what-is-medallion-architecture

[28] M. Raasveldt y H. Mühleisen, «DuckDB: an Embeddable Analytical Database», en *Proceedings of the 2019 International Conference on Management of Data*, en SIGMOD '19. New York, NY, USA: Association for Computing Machinery, jun. 2019, pp. 1981-1984. doi: 10.1145/3299869.3320212.
[29] W. McKinney, «Data Structures for Statistical Computing in Python», presentado en Python in Science Conference, Austin, Texas, 2010, pp. 56-61. doi: 10.25080/Majora-92b*F1*922-00a.
[30] J. Yang, H. Zhou, X. Chen, y L. Cheng, «Applying the Support Vector Machine to Predicting Headway-Based Bus Bunching», pp. 1542-1553, jul. 2019, doi: 10.1061/9780784482292.135.
[31] K. K. Reddy, B. Anil Kumar, y L. Vanajakshi, «Bus Travel Time Prediction under High Variability Conditions», *Current Science*, vol. 111, n.º 4, p. 700, ago. 2016, doi: 10.18520/cs/v111/i4/700-711.
[32] «Resolución 419 de 2015 Empresa de Transporte del Tercer Milenio - Transmilenio S.A». Accedido: 21 de abril de 2026. [En línea]. Disponible en: https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=62552&dt=S
[33] «Portal de Datos Abiertos de TransMilenio». Accedido: 26 de abril de 2026. [En línea]. Disponible en: https://datosabiertos-transmilenio.hub.arcgis.com/
[34] «Optuna - A hyperparameter optimization framework», Optuna. Accedido: 7 de mayo de 2026. [En línea]. Disponible en: https://optuna.org/

ANEXOS

Anexo 1. Variables del dataset gold - Descripción y atributos

Variable	Grupo	Tipo	Fuente original	Rol	Tratamiento de nulos	Transform. Pipeline B	Descripción
hora_dia	Contexto	*INTEGER*	datetime (intervalos)	Feature	Sin nulos	Ninguna	Hora del día (0–23)
dia_semana	Contexto	*INTEGER*	datetime (intervalos)	Feature	Sin nulos	Ninguna	Día de la semana (0=lunes, 6=domingo)
mes	Contexto	*INTEGER*	datetime (intervalos)	Feature	Sin nulos	Ninguna	Mes del año (1–12)
id_estacion	Contexto	*BIGINT*	intervalos_servicios	Feature	Sin nulos	Ninguna	Identificador numérico de la estación
intervalo_teorico_s	Contexto	*BIGINT*	intervalos_servicios	Feature	Sin nulos (filas con ≤0 eliminadas en preprocessing)	log1p → RobustScaler (re-fit silver.train: mediana=5.80, IQR=0.48)	Headway programado en segundos
apertura_puertas	Contexto	*FLOAT*	dwell_time (sensor de puertas)	Feature	Imputada con mediana por (servicio, id_estacion); ceros tratados como faltantes	Winsor(P99=139) → log1p → StandardScaler (re-fit silver.train: mean_log=2.73, std_log=0.50)	Tiempo de apertura de puertas en el arribo (segundos)
te_servicio	Contexto	*FLOAT*	intervalos_servicios	Feature	Sin nulos (fallback = media global para servicios no vistos)	StandardScaler (re-fit sobre silver.train)	Target encoding del servicio (tasa de bunching suavizada)

te_id_estacion	Contexto	*FLOAT*	intervalos_servicios	Feature	Sin nulos (fallback = media global)	StandardScaler (re-fit sobre silver.train)	Target encoding de la estación (tasa de bunching suavizada)
lag_hr_1	*Lag*	*DOUBLE*	Calculada — lag t−1 de headway_ratio sobre (servicio, id_estacion)	Feature	Filas eliminadas cuando lag_hr_1 IS NULL (primer registro por servicio×estación)	Winsor(P99=9.08) → log1p → StandardScaler	Headway ratio del intervalo inmediatamente anterior
lag_hr_2	*Lag*	*DOUBLE*	Calculada — lag t−2 de headway_ratio sobre (servicio, id_estacion)	Feature	Backward fill con lag_hr_1 cuando no disponible	Winsor(P99=9.08) → log1p → StandardScaler	Headway ratio dos intervalos atrás
lag_hr_3	*Lag*	*DOUBLE*	Calculada — lag t−3 de headway_ratio sobre (servicio, id_estacion)	Feature	Backward fill con lag_hr_2 / lag_hr_1 cuando no disponible	Winsor(P99=9.08) → log1p → StandardScaler	Headway ratio tres intervalos atrás
lag_ap_1	*Lag*	*FLOAT*	Calculada — lag t−1 de apertura_puertas sobre (servicio, id_estacion)	Feature	Backward fill con apertura_puertas(t) cuando no disponible	Winsor(P99=139) → log1p → StandardScaler (re-fit silver.train: mean_log=2.73, std_log=0.50)	Apertura de puertas del intervalo inmediatamente anterior (s)
lag_ap_2	*Lag*	*FLOAT*	Calculada — lag t−2 de apertura_puertas sobre (servicio, id_estacion)	Feature	Backward fill con lag_ap_1 cuando no disponible	Winsor(P99=139) → log1p → StandardScaler (re-fit silver.train: mean_log=2.73, std_log=0.50)	Apertura de puertas dos intervalos atrás (s)
lag_val_1	*Lag*	*INTEGER*	Calculada — lag t−1 de validaciones sobre (servicio, id_estacion)	Feature	Hereda la imputación con 0 de validaciones	Winsor(P99=282) → log1p → StandardScaler	Validaciones del intervalo inmediatamente anterior
roll_hr_mean_3	*Rolling*	*DOUBLE*	Calculada sobre headway_ratio (ventana t−3 a t−1)	Feature	Backward fill con lag_hr_1 cuando la	Winsor(P99=9.08) → log1p → StandardScaler	Media móvil del headway ratio en los 3

					ventana está incompleta		intervalos anteriores
roll_ap_mean_3	*Rolling*	*DOUBLE*	Calculada sobre apertura_puertas (ventana t−3 a t−1)	Feature	Backward fill con lag_ap_1 cuando la ventana está incompleta	Winsor(P99=139) → log1p → StandardScaler (re-fit silver.train: mean_log=2.73, std_log=0.50)	Media móvil de apertura de puertas en los 3 intervalos anteriores (s)
roll_ap_std_3	*Rolling*	*DOUBLE*	Calculada sobre apertura_puertas (ventana t−3 a t−1)	Feature	Fill con 0 cuando la ventana está incompleta	log1p → StandardScaler	Desviación estándar móvil de apertura de puertas (ventana 3)
delta_hr_1	Derivada	*DOUBLE*	Calculada: lag_hr_1 − lag_hr_2	Feature	Hereda tratamiento de los lags de headway_ratio	StandardScaler	Tasa de cambio del headway ratio entre t−2 y t−1
delta_ap_1	Derivada	*FLOAT*	Calculada: lag_ap_1 − lag_ap_2	Feature	Hereda tratamiento de los lags de apertura_puertas	StandardScaler	Tasa de cambio de apertura de puertas entre t−2 y t−1 (segundos)
headway_acum	Acumulada	*DOUBLE*	Calculada: Σ(headway_ratio − 1) previos, partición (servicio, fecha)	Feature	Fill con 0 al inicio del día	StandardScaler	Acumulado de desviaciones del headway del servicio en la red hasta t−1 (deriva diaria)
lag_hr1_x_ap	Interacción	*DOUBLE*	Calculada: lag_hr_1 × apertura_puertas	Feature	Hereda tratamiento de sus componentes	StandardScaler	Interacción entre regularidad reciente y demanda en estación (apertura)

station_mean_dwell_5min	*Cross-station*	*DOUBLE*	Calculada — AVG(apertura_puertas) sobre (id_estacion, sentido_cardinal), ventana 5 min	Feature	Fill con 0 cuando la ventana está vacía (inicio de partición o borde de chunk)	Winsor(P99=139) → log1p → StandardScaler (fit gold.train)	Dwell promedio de cualquier servicio en la estación, mismo sentido, últimos 5 min (s)
station_max_dwell_5min	*Cross-station*	*FLOAT*	Calculada — MAX(apertura_puertas) sobre (id_estacion, sentido_cardinal), ventana 5 min	Feature	Fill con 0 cuando la ventana está vacía	Winsor(P99=139) → log1p → StandardScaler (fit gold.train)	Dwell máximo de cualquier servicio en la estación, mismo sentido, últimos 5 min (s)
station_total_dwell_5min	*Cross-station*	*DOUBLE*	Calculada — SUM(apertura_puertas) sobre (id_estacion, sentido_cardinal), ventana 5 min	Feature	Fill con 0 cuando la ventana está vacía	log1p → StandardScaler (fit gold.train)	Tiempo total de plataforma agregado en la estación, mismo sentido, últimos 5 min (s)
station_bunching_rate_15min	*Cross-station*	*DOUBLE*	Calculada — AVG(bunching_025) sobre (id_estacion, sentido_cardinal), ventana 15 min	Feature	Fill con 0 cuando la ventana está vacía	StandardScaler directo (rango [0, 1], sin winsor ni log)	Fracción de arribos recientes con bunching en la estación, mismo sentido, últimos 15 min
station_dt_to_prev_arrival_s	*Cross-station*	*DOUBLE*	Calculada — EPOCH(datetime − LAG(datetime)) sobre (id_estacion, sentido_cardinal)	Feature	Fill con 0 al inicio de cada partición (estación, sentido)	Winsor(P99 propio) → log1p → StandardScaler (fit gold.train)	Segundos desde el último arribo de cualquier servicio del mismo sentido en la estación
station_n_arrivals_5min	*Cross-station*	*BIGINT*	Calculada — COUNT(*) sobre (id_estacion, sentido_cardinal), ventana 5 min	Feature	Fill con 0 cuando la ventana está vacía	log1p → StandardScaler (fit gold.train)	Número de arribos de cualquier servicio del mismo sentido

							en la estación en los últimos 5 min
station_n_arrivals_30min	*Cross-station*	*BIGINT*	Calculada — COUNT(*) sobre (id_estacion, sentido_cardinal), ventana 30 min	Feature	Fill con 0 cuando la ventana está vacía	log1p → StandardScaler (fit gold.train)	Número de arribos del mismo sentido en la estación, últimos 30 min (saturación sostenida)
station_mean_dwell_15min	*Cross-station*	*DOUBLE*	Calculada — AVG(apertura_puertas) sobre (id_estacion, sentido_cardinal), ventana 15 min	Feature	Fill con 0 cuando la ventana está vacía	Winsor(P99=139) → log1p → StandardScaler (fit gold.train)	Dwell promedio agregado en la estación, mismo sentido, últimos 15 min (s)
station_bunching_rate_5min	*Cross-station*	*DOUBLE*	Calculada — AVG(bunching_025) sobre (id_estacion, sentido_cardinal), ventana 5 min	Feature	Fill con 0 cuando la ventana está vacía	StandardScaler directo (rango [0, 1], sin winsor ni log)	Fracción de arribos recientes con bunching en la estación, mismo sentido, últimos 5 min (cascada inmediata)
upstream1_mean_hr_15min	*Upstream service-level*	*DOUBLE*	Calculada — AVG(headway_ratio) del mismo servicio en la estación 1 posición aguas arriba (orden_zona − direction), ventana 15 min	Feature	Fill con 0 cuando no hay arribos del servicio en esa estación en la ventana	StandardScaler (columna escalada de gold.train_scaled)	Headway ratio promedio del mismo servicio en la estación inmediatamente anterior de su recorrido, últimos 15 min
upstream1_std_hr_15min	*Upstream service-level*	*DOUBLE*	Calculada — STDDEV(headway_ratio) del mismo servicio en la estación 1 posición	Feature	Fill con 0 cuando hay menos de 2 arribos en la ventana	StandardScaler (columna escalada de gold.train_scaled)	Desviación estándar del headway ratio del mismo servicio en la

			aguas arriba, ventana 15 min				estación inmediatamente anterior, últimos 15 min
upstream1_bunching_rate_15min	*Upstream service-level*	*DOUBLE*	Calculada — AVG(bunching_025) del mismo servicio en la estación 1 posición aguas arriba, ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	StandardScaler directo (rango [0, 1])	Fracción de arribos del mismo servicio con bunching en la estación inmediatamente anterior, últimos 15 min
upstream1_n_arrivals_15min	*Upstream service-level*	*BIGINT*	Calculada — COUNT(*) del mismo servicio en la estación 1 posición aguas arriba, ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	log1p → StandardScaler (columna escalada de gold.train_scaled)	Número de arribos del mismo servicio en la estación inmediatamente anterior, últimos 15 min
upstream2_mean_hr_15min	*Upstream service-level*	*DOUBLE*	Calculada — AVG(headway_ratio) del mismo servicio en la estación 2 posiciones aguas arriba (orden_zona − 2·direction), ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	StandardScaler (columna escalada de gold.train_scaled)	Headway ratio promedio del mismo servicio dos estaciones aguas arriba, últimos 15 min
upstream2_std_hr_15min	*Upstream service-level*	*DOUBLE*	Calculada — STDDEV(headway_ratio) del mismo servicio en la estación 2 posiciones aguas arriba, ventana 15 min	Feature	Fill con 0 cuando hay menos de 2 arribos en la ventana	StandardScaler (columna escalada de gold.train_scaled)	Desviación estándar del headway ratio del mismo servicio dos estaciones aguas arriba, últimos 15 min

upstream2_bunching_rate_15min	*Upstream service-level*	*DOUBLE*	Calculada — AVG(bunching_025) del mismo servicio en la estación 2 posiciones aguas arriba, ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	StandardScaler directo (rango [0, 1])	Fracción de arribos del mismo servicio con bunching dos estaciones aguas arriba, últimos 15 min
upstream2_n_arrivals_15min	*Upstream service-level*	*BIGINT*	Calculada — COUNT(*) del mismo servicio en la estación 2 posiciones aguas arriba, ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	log1p → StandardScaler (columna escalada de gold.train_scaled)	Número de arribos del mismo servicio dos estaciones aguas arriba, últimos 15 min
upstream3_mean_hr_15min	*Upstream service-level*	*DOUBLE*	Calculada — AVG(headway_ratio) del mismo servicio en la estación 3 posiciones aguas arriba (orden_zona − 3·direction), ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	StandardScaler (columna escalada de gold.train_scaled)	Headway ratio promedio del mismo servicio tres estaciones aguas arriba, últimos 15 min
upstream3_std_hr_15min	*Upstream service-level*	*DOUBLE*	Calculada — STDDEV(headway_ratio) del mismo servicio en la estación 3 posiciones aguas arriba, ventana 15 min	Feature	Fill con 0 cuando hay menos de 2 arribos en la ventana	StandardScaler (columna escalada de gold.train_scaled)	Desviación estándar del headway ratio del mismo servicio tres estaciones aguas arriba, últimos 15 min
upstream3_bunching_rate_15min	*Upstream service-level*	*DOUBLE*	Calculada — AVG(bunching_025) del mismo servicio en la estación 3 posiciones aguas arriba, ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	StandardScaler directo (rango [0, 1])	Fracción de arribos del mismo servicio con bunching tres estaciones aguas arriba, últimos 15 min

upstream3_n_arrivals_15min	*Upstream service-level*	*BIGINT*	Calculada — COUNT(*) del mismo servicio en la estación 3 posiciones aguas arriba, ventana 15 min	Feature	Fill con 0 cuando no hay arribos en la ventana	log1p → StandardScaler (columna escalada de gold.train_scaled)	Número de arribos del mismo servicio tres estaciones aguas arriba, últimos 15 min
bunching_025	*Target*	*INTEGER*	Calculada: headway_ratio < 0.25	Target	Sin nulos	Ninguna (variable objetivo)	Clase binaria: 1 = bunching severo (headway_ratio < 0.25), 0 = sin bunching