MOVILIDAD INTELIGENTE: PREDICIENDO LA AFLUENCIA EN EL METRO DE MEDELLÍN

Proyecto integrador segundo semestre - Maestría en Ciencia de datos y analítica

Autores
Equipo # 5
Santiago Areiza Tamayo
Santiago Carvajal Torres
Pedro Gómez Bedoya

Sábado, 29 de noviembre de 2025

INTRODUCCIÓN

El Metro de Medellín es el sistema de transporte masivo que conecta Medellín con varios municipios del Valle de Aburrá. Inaugurado en 1995, moviliza a cientos de miles de pasajeros cada día a través de trenes, metrocables, tranvía y buses integrados.

Uno de sus principales retos son las horas pico, especialmente entre 5:30–8:30 a. m. y 4:30–7:30 p. m., cuando estaciones como San Antonio e Industriales alcanzan niveles de congestión que afectan la comodidad y eficiencia del servicio.

Aunque el Metro posee datos históricos de afluencia desde 2019, la gestión de la capacidad suele ser reactiva. El objetivo es predecir la cantidad de pasajeros por línea y horario para 2025, usando la información de años anteriores y comparando los resultados con los datos reales de ese año.

Anticipar la demanda es clave para mejorar la planificación operativa, ajustar frecuencias antes de la congestión y optimizar recursos.

Para resolver esta problemática se propone abordar la situación empleando las herramientas aprendidas durante el semestre de la siguiente manera:

- **Visualización de datos:** identificar patrones diarios y estacionales.
- **Métodos estadísticos avanzados:** modelos como ARIMA/SARIMA para series temporales.
- **Aprendizaje Automático (Machine Learning)**: modelos supervisados para pronóstico y comparación como *Random Forest* y *XGBoost*.

Este enfoque combina análisis históricos con técnicas modernas para apoyar una movilidad más eficiente e inteligente.

MARCO TEÓRICO

El transporte masivo urbano busca ofrecer movilidad eficiente, segura y sostenible a gran escala. En el caso del Metro de Medellín, la gestión de la capacidad es un desafío constante debido a la concentración de pasajeros en horas pico, lo que genera congestión y disminuye la calidad del servicio. Según el *Transit Capacity and Quality of Service Manual* (TRB, 2013), la calidad en el transporte depende de factores como la frecuencia, el confort y la regularidad, los cuales pueden optimizarse mediante una planificación basada en datos.

El análisis histórico de la demanda permite identificar patrones diarios, semanales y estacionales, fundamentales para anticipar la afluencia y ajustar la operación antes de que se presenten sobrecargas. Los sistemas de transporte modernos utilizan datos de

validaciones, conteos automáticos y registros horarios para construir modelos predictivos que guían la toma de decisiones.

En este contexto, la analítica de series temporales se convierte en una herramienta clave para mejorar la eficiencia operativa. Los modelos ARIMA y su extensión SARIMA (Seasonal ARIMA) son ampliamente utilizados para pronosticar fenómenos con comportamiento periódico, como la demanda de pasajeros en sistemas metro. Estos modelos permiten capturar la tendencia y la estacionalidad presentes en los datos, generando pronósticos de corto plazo confiables. De acuerdo con Box, Jenkins y Reinsel (2015), el enfoque ARIMA es adecuado cuando las observaciones presentan dependencia temporal y patrones repetitivos, como ocurre en el transporte urbano. Estudios recientes han mostrado que los modelos SARIMA ofrecen buenos resultados al predecir la afluencia diaria o por hora en líneas de metro.

Para garantizar un pronóstico robusto y alineado con los requerimientos de la ciencia de datos moderna, se integrarán modelos de Machine Learning (ML) supervisado mediante una estrategia de reducción de series temporales a un problema de regresión tabular. Esto implica la extracción de características (Feature Engineering), como lag features, ventanas móviles y variables exógenas (días festivos, etc.), transformando la serie temporal en un conjunto de datos con el que puedan trabajar modelos como *Random Forest* o *XGBoost*. La combinación de estos modelos de ML con el baseline estadístico (SARIMA) permitirá una comparación rigurosa y la validación de la solución más precisa y operable.

Además, la visualización de datos facilita la detección de cambios estructurales y la evaluación de la efectividad de las medidas adoptadas. Implementar este tipo de modelación en el Metro de Medellín permitiría pasar de una gestión reactiva a una gestión predictiva y preventiva, mejorando la experiencia del usuario, reduciendo la congestión y apoyando la movilidad sostenible en el Valle de Aburrá.

En síntesis, la integración de la analítica estadística, el aprendizaje automático y la visualización de datos representa un paso esencial hacia una movilidad inteligente y basada en evidencia.

DESARROLLO METODOLÓGICO

Entendimiento del problema y pregunta de negocio

El Metro de Medellín es el eje del transporte masivo en el Valle de Aburrá y moviliza diariamente a cientos de miles de pasajeros a través de diferentes modos (metro, metrocable, tranvía y buses integrados). Sin embargo, la demanda no es constante en el tiempo: se concentra fuertemente en las horas pico de la mañana (5:00–7:00 a. m.) y de la tarde (4:00–6:00 p. m.), generando niveles de congestión elevados en estaciones críticas como San Antonio, Industriales, Poblado o Niquía.

En la actualidad, la gestión de la capacidad (frecuencia de trenes, uso de flota y recursos operativos) es principalmente reactiva: las decisiones se toman una vez se observa la congestión, en lugar de anticiparla. Esto limita la posibilidad de suavizar los picos, redistribuir la oferta de forma preventiva y comunicar de antemano a los usuarios las mejores franjas para viajar con menor ocupación.

A pesar de que el Metro cuenta con datos históricos de afluencia por línea y horario desde 2019, esta información no se explota de forma sistemática mediante modelos de series de tiempo y aprendizaje automático. En este proyecto se busca transformar esos datos en un sistema de predicción de afluencia que permita estimar cuántos pasajeros se espera que ingresen al sistema por línea y franja horaria en 2025, utilizando el comportamiento observado en los años anteriores.

En este contexto, la pregunta de negocio que guía el trabajo es:

> ¿Cuál será la cantidad esperada de pasajeros por línea de servicio y franja horaria en 2025, y cómo puede utilizarse esta predicción para anticipar y gestionar las horas pico en el sistema Metro de Medellín?

Con base en ello, se formulan las siguientes hipótesis de trabajo:

- **H1.** La afluencia al Metro de Medellín presenta una estacionalidad marcada a nivel horario, con patrones consistentes de horas pico y horas valle.
- **H2.** La incorporación de variables de calendario (día de la semana, mes, festivo) y del historial reciente de afluencia mejora de forma significativa la precisión de los modelos de predicción frente a un modelo con pocos recursos.
- **H3.** A partir de los modelos es posible identificar anticipadamente las franjas y líneas con mayor riesgo de congestión en 2025, aportando insumos para planear la capacidad y la operación.

Análisis Exploratorio

● Entendimiento de los datos

La base de datos utilizada corresponde a los registros históricos de afluencia del Metro de Medellín entre 2019 y 2025, suministrados en un archivo Excel con una hoja por año. Cada hoja contiene, para cada día calendario, el número de pasajeros registrados por línea de servicio y por franja horaria, concentrándose principalmente en las horas operativas del sistema.

La unidad de análisis es el número de pasajeros que ingresan al sistema por línea y hora, lo que permite estudiar la demanda a un nivel de detalle adecuado para identificar horas pico y horas valle. Adicionalmente, la estructura de los archivos incluye totales diarios y encabezados en múltiples niveles, lo que requiere un proceso de limpieza y transformación antes de realizar el análisis.

Este entendimiento inicial permite aclarar el alcance temporal (2019–2025), el nivel de granularidad (línea–hora–día) y el tipo de variable objetivo sobre la cual se realizarán los modelos de predicción.

- ## Preparación de los datos

  La preparación de los datos se realizó en varias etapas, utilizando principalmente la librería pandas de Python:

  1. **Lectura y unificación de hojas**
     - Se cargaron las hojas anuales (Afluencia_2019, …, Afluencia_2025) desde el archivo Excel original.
     - Debido a que los encabezados venían en dos niveles (multiindex), se aplanaron y renombraron las columnas para obtener una estructura tabular estándar.

  2. **Transformación a formato largo**
     - A partir de las columnas de horas (por ejemplo, 05:00, 05:15, etc.), se aplicó una operación de melt para convertir la tabla de formato ancho a un formato largo, de modo que cada fila representara una combinación:
       Día – Línea de servicio – Hora – Número de pasajeros.

  3. **Limpieza y estandarización**
     - Se convirtieron los campos de afluencia a tipo numérico, imputando valores faltantes o no válidos como cero cuando fue necesario.
     - Se normalizaron los nombres de las líneas de servicio, unificando variantes de escritura (mayúsculas, acentos) bajo un mismo formato estándar (por ejemplo, “LÍNEA A”, “linea a” → “Línea A”).

  4. **Construcción de la variable de fecha y consolidación temporal**
     - A partir del día y la hora se construyó una variable de fecha y hora completa, ajustando el formato según el año (considerando cambios en el orden día/mes en las fuentes originales).
     - Posteriormente, se formateó la fecha en un formato homogéneo y se eliminaron los registros cuya fecha no se pudo interpretar correctamente.
     - Todas las hojas anuales limpias se concatenaron en un único DataFrame consolidado, que contiene la serie completa 2019–2025.

  5. **Generación de variables derivadas para el análisis**
     - A partir de la fecha se crearon variables temporales como Año, Mes, Día de la semana y Hora.
     - Se incorporó una variable binaria Es_Festivo, utilizando el calendario oficial de festivos en Colombia, para distinguir días laborales de días especiales.

- ## Análisis descriptivo

  Con el dataset consolidado y preparado, se llevó a cabo un análisis descriptivo orientado a identificar los principales patrones de demanda:

  1. **Tendencia temporal global**
     - Se construyó una serie diaria de afluencia total del sistema sumando todos los registros por fecha.
     - Sobre esta serie se calculó un promedio móvil de 7 días para suavizar la variabilidad y resaltar la tendencia general, lo que permite observar claramente el impacto de eventos como la pandemia y la posterior recuperación de la demanda.

  2. **Patrones de estacionalidad diaria y semanal**
     - Se calculó la afluencia promedio por hora del día para todo el periodo, identificando las horas pico de la mañana y de la tarde, así como las horas valle con menor ocupación.
     - Se analizó la afluencia promedio por día de la semana, diferenciando el comportamiento de días laborales frente a fines de semana y observando posibles cambios en la distribución de la demanda.

  3. **Comparación entre líneas de servicio**
     - Se estudiaron los patrones horarios de afluencia por línea, destacando una línea dominante (por ejemplo, Línea A) y comparándola con el resto. Esto permite identificar qué líneas concentran mayor volumen de pasajeros y en qué horarios su comportamiento difiere.

  4. **Influencia de días festivos**
     - Mediante la variable Es_Festivo, se compararon las distribuciones de afluencia en días festivos frente a días no festivos, con el fin de cuantificar cómo cambia la demanda en jornadas especiales.

Este análisis descriptivo proporciona una visión clara de los niveles de carga del sistema y de sus patrones de variación en el tiempo, constituyendo la base para la selección y calibración de los modelos de predicción que se desarrollarán en etapas posteriores.

Selección de modelos

Dado que el objetivo es modelar y predecir la afluencia de pasajeros a lo largo del tiempo, se seleccionaron modelos de la familia ARIMA, ampliamente utilizados en series de tiempo con dependencia temporal y posibles componentes estacionales. Así mismo, debido al auge del *machine learning* y su excepcional capacidad de detectar patrones en las series de tiempo, se escogieron los siguientes modelos:

- Modelos
    - **ARIMA(p,d,q)**: modelo básico sin componente estacional ni variables exógenas, que sirve como línea base.

    - **SARIMA(p,d,q)(P,D,Q)_7**: extensión de ARIMA que incorpora **estacionalidad**, donde m=7 refleja la estacionalidad semanal, apropiada para capturar patrones semanales en la afluencia de pasajeros.

    - **SARIMAX(p,d,q)(P,D,Q)_7 con variables exógenas**: modelo estacional que además incorpora la variable **Es_Festivo** para capturar el impacto específico de los días festivos sobre la demanda.

    - ***Random forest* con variables exógenas**: modelo de *machine learning* que incorpora la variable **Es_Festivo**, con el mismo objetivo que lo mencionado para los modelos de la familia ARIMA.

    - ***XGBoost* con variables exógenas**: modelo de *machine learning* que incorpora la variable **Es_Festivo**, igual que para *random forest* y los modelos ARIMA.

En lugar de fijar manualmente los órdenes (p,d,q) y (P,D,Q,m), se utilizó la función auto_arima del paquete pmdarima, qué busca de forma automática la combinación de parámetros que minimiza criterios de información (como el AIC). Se ejecutó auto_arima por separado para cada configuración (ARIMA sin estacionalidad, SARIMA estacional sin exógenas y SARIMAX estacional con Es_Festivo), lo que permitió comparar modelos con una base de selección objetiva y guiada por los datos.

En cuanto a los modelos de *machine learning*, para *random forest* se creó un modelo con 500 árboles con una profundidad de 10 niveles para evitar demasiado sobreajuste. Por otro lado, para *XGBoost* inicialmente se creó un modelo con 10000 iteraciones, una tasa de aprendizaje de 0.01 para realizar saltos pequeños, una profundidad máxima de 5 niveles, la métrica para la parada temprana o *stop early* fue *RMSE* y con un máximo de 50 iteraciones para detenerse si no hay un cambio en la métrica objetivo.

Además, para *XGBoost*, tras hacer pruebas comparativas con *Random Forest*, se optó por hacerle un ajuste de hiperparámetros mediante búsqueda aleatoria o *random search* para asegurarse de tener un modelo robusto, que generalice bien y que tenga un mejor rendimiento al hacer las predicciones en el conjunto de pruebas.

- Carácteristicas
    - Variable objetivo (target)
        - Número de pasajeros: serie temporal que representa la afluencia de pasajeros en el tiempo, obtenida tras el preprocesamiento y consolidación de los datos en df_consolidado y df_modelo.

    - Variables de calendario y auxiliares (para EDA y modelado)
      A partir del campo de fecha se generaron:

        - Fecha_dt: fecha y hora en formato datetime.
        - Año, Mes, Día_Semana, Hora: variables derivadas para análisis exploratorio (identificación de patrones por día, mes, hora, etc.).
        - Estas variables se utilizaron principalmente en el análisis descriptivo y la preparación de datos.
    - Variable exógena para los modelos SARIMAX, *Random forest y XGBoost*:
        - Es_Festivo: variable binaria construida usando el paquete holidays para Colombia. Toma valor 1 si la fecha corresponde a un día festivo oficial y 0 en caso contrario.
        - Esta variable se incluye como regresor exógeno (exog) en el modelo SARIMAX, permitiendo capturar incrementos o disminuciones puntuales en la afluencia asociados a días festivos.

En la fase de modelado con SARIMAX se utilizó como variable explicativa principal Es_Festivo, manteniendo la serie Número de pasajeros como objetivo.

Por otra parte, para los modelos de *machine learning*, se hizo una fase previa antes del entrenamiento que consistió en la extracción de características o *feature extraction*, en la cual se extrajeron características como la media, la mediana, la sumatoria de los valores, el valor máximo y mínimo, entre otras, para cada línea de servicio utilizando la librería *tsfresh*. Esto se hizo tanto para el conjunto de entrenamiento como para los conjuntos de validación y pruebas.

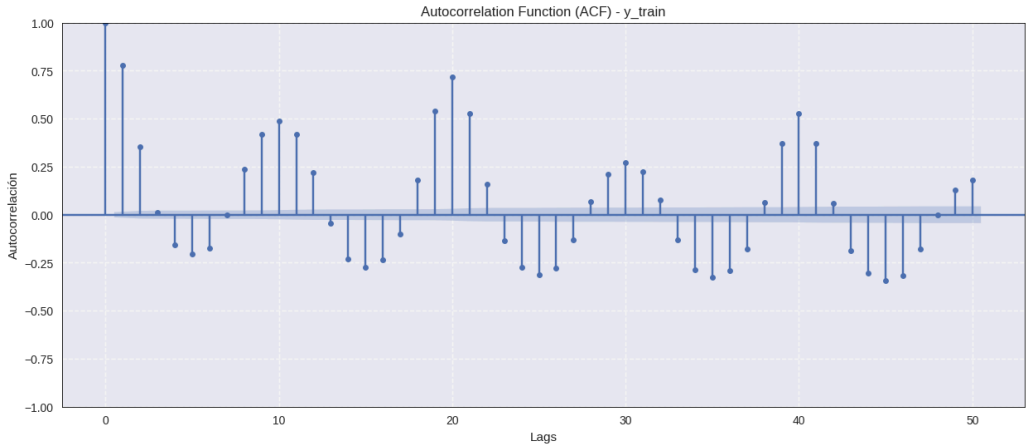
Posterior a eso, se establecieron los retrasos o *lag features* los cuales fueron de 1, 2 y 7 días para los tres conjuntos. Del mismo modo, se establecieron las ventanas móviles o *rolling windows* con los valores de las medias, las desviaciones estándar y los valores mínimos y máximos. Después, se hizo la gestión de los valores nulos o NaN imputando dichos valores con la media de su respectiva línea de servicio.

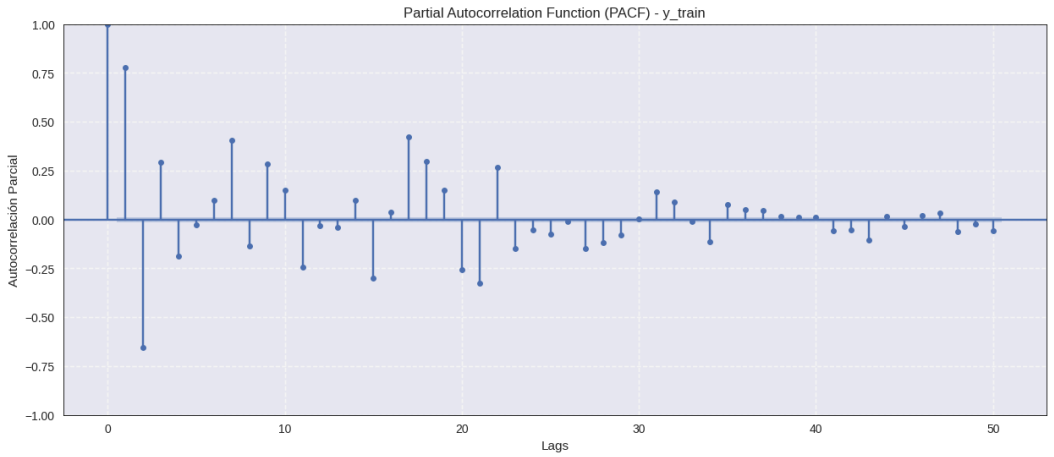
Finalmente, se combinaron las características en los *datasets* finales que se usarían para el entrenamiento, validación y pruebas, enriquecidos para una mejor predicción de cada serie de tiempo.

- Entrenamiento
    - **División temporal de los datos (Triple Split)**
      Para respetar la naturaleza temporal de la serie, se realizó un split temporal en tres subconjuntos utilizando temporal_train_test_split de sktime:
        - TRAIN (80 % inicial): conjunto utilizado para ajustar los modelos y estimar los parámetros.
        - VALIDATION (10 % intermedio): segmento más reciente que el train, usado para comparar el desempeño de los modelos y seleccionar la mejor configuración.
        - TEST (10 % final): tramo más reciente de la serie, reservado exclusivamente para la evaluación final del modelo seleccionado.

Esta estrategia garantiza que la evaluación se realice respetando la causalidad (solo se predicen datos futuros a partir de información pasada).

- **Selección de órdenes con auto_arima**
  - Si observamos el ACF y PACF vemos que tiene una estacionalidad marcada en el ACF y PACF.





  - Para cada tipo de modelo (ARIMA, SARIMA, SARIMAX) se ejecutó auto_arima sobre el conjunto TRAIN, pasando y_train (y, en el caso de SARIMAX, también X_train con Es_Festivo).
  - auto_arima exploró diferentes combinaciones de (p,d,q) y (P,D,Q,m) buscando el mejor compromiso entre ajuste y complejidad según criterios de información (AIC/BIC).

- **Elección del modelo:** Para la familia de modelos ARIMA y con base a la respuesta del auto_arima y el gráfico ACF y PACF, se decidió tomar los modelos (5,1,0).

- **Ajuste de modelos en Train y validación**
  - Los tres modelos ajustados se evaluaron sobre el conjunto **VALIDATION** utilizando métricas de error clave:

- **MAE (Error Absoluto Medio):** Mide el error promedio en unidades de pasajeros.
- **RMSE (Raíz del Error Cuadrático Medio):** Penaliza fuertemente los errores grandes, siendo sensible a picos de demanda no capturados.

■

Modelo	MAE	RMSE
ARIMA	32909.4	38361.9
SARIMA	32909.4	38361.9
SARIMAX	32376.9	37791.8

Los resultados mostraron que el Modelo 3 (SARIMAX con la variable exógena Es_Festivo) obtuvo los menores valores de MAE y RMSE, posicionándose como el mejor entre las alternativas evaluadas.

Adicionalmente, al analizar nuevamente las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF), se evidenció que la serie sí presenta patrones temporales relevantes, con picos significativos en distintos rezagos y bloques de correlación positivos y negativos. Esto confirma que el proceso no es ruido blanco y que la dinámica temporal posee dependencias que pueden ser capturadas por modelos ARIMA/SARIMA.

Sin embargo, a pesar de esta estructura autocorrelativa, el modelo final todavía no logra ajustarse completamente al comportamiento real, especialmente en la zona de test. La brecha entre predicción y observación sugiere que:

- La serie tiene alta variabilidad diaria y cambios abruptos difíciles de capturar con componentes lineales.
- Existen patrones estacionales irregulares que no siguen un ciclo fijo o estrictamente repetitivo.
- La variable exógena utilizada (Es_Festivo) mejora el ajuste, pero es probable que no capture toda la complejidad del fenómeno.
- El modelo SARIMAX, aunque adecuado según el ACF/PACF, puede estar limitado para describir relaciones no lineales o efectos contextuales más fuertes.

En conjunto, esto indica que, si bien SARIMAX aprovecha la estructura temporal revelada por los gráficos, su capacidad predictiva sigue siendo insuficiente para una serie tan compleja, por lo que modelos más flexibles como árboles de gradiente XGBoost pueden predecir mejor.

Ahora bien, para los modelos de *machine learning* el entrenamiento se hizo con el mismo *split* o división de los datos descrita para los modelos de la familia ARIMA. Inicialmente se

crearon dos modelos “iniciales” de *Random Forest* y *XGBoost*, cuya parametrización inicial fue descrita anteriormente al inicio de este apartado. Esto se hizo con el objetivo de entrenar ambos modelos con parámetros estándar para luego comparar sus métricas de error (MSE y RMSE) y, con base en esta comparación, elegir el mejor para realizar el respectivo ajuste de hiperparámetros.

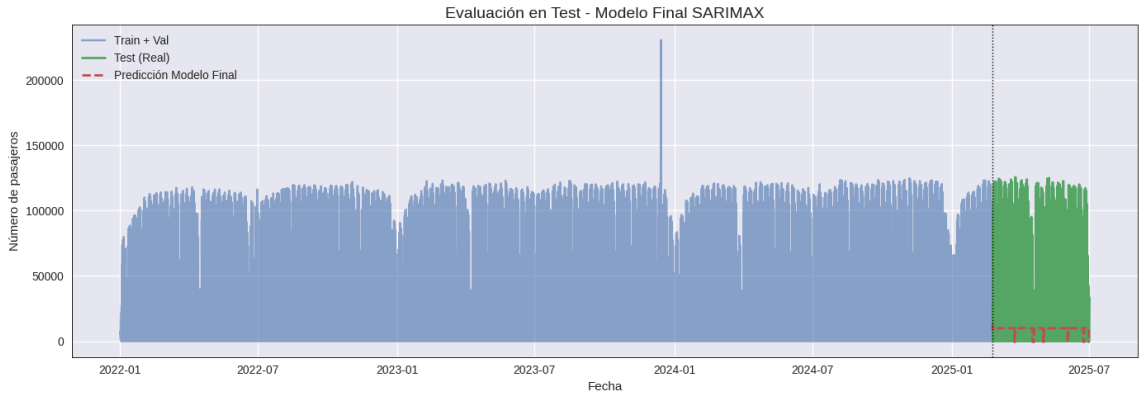
Tras el entrenamiento y al realizar la evaluación de las métricas en el conjunto de validación para ambos modelos se obtuvo los siguientes resultados:

Modelo	MSE	RMSE
Random Forest	187178.96	432.64
XGBoost	153600.69	391.92

Frente a estos resultados, se escogió el modelo *XGBoost* para realizar el ajuste de hiperparámetros mediante la búsqueda aleatoria o *random search*, donde se seleccionó el mejor modelo ajustado para nuevamente realizar la evaluación de las métricas en el conjunto de validación donde se obtuvo un **MSE** de **84501.09** y un **RMSE** de **290.69**, mostrando una diferencia muy significativa con el modelo inicial de *XGBoost*, dejando como claro ganador dicho modelo comparado tanto con los modelos de *machine learning* iniciales como los modelos de la familia ARIMA entrenados y evaluados en este trabajo.

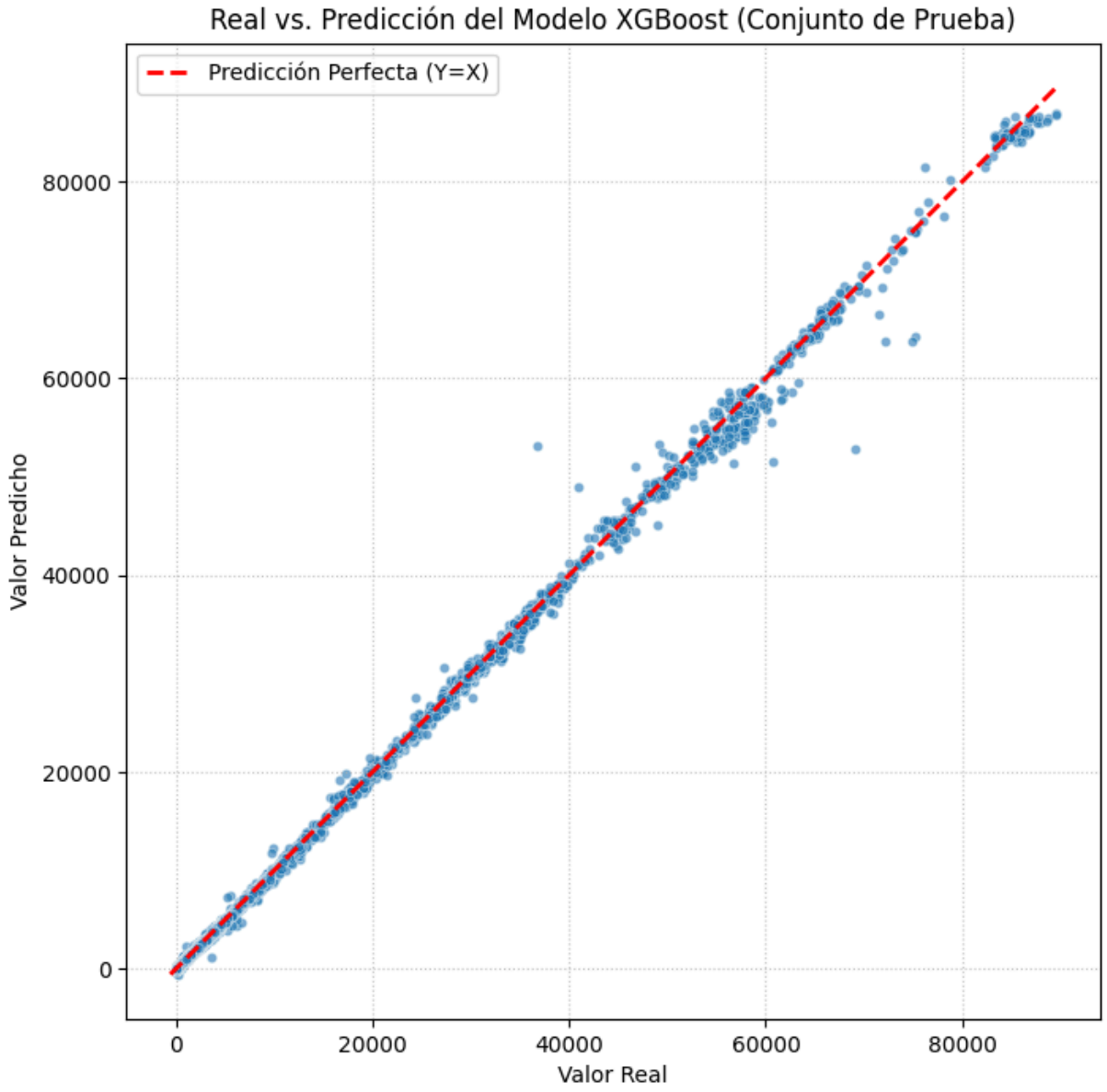
Evaluación

El modelo **SARIMAX seleccionado** fue **reentrenado** con el conjunto **TRAIN + VALIDATION** completo para aprovechar la mayor cantidad de datos posible. Finalmente, se procedió a la evaluación definitiva sobre el conjunto **TEST**, en la cual se obtuvieron los siguientes resultados:

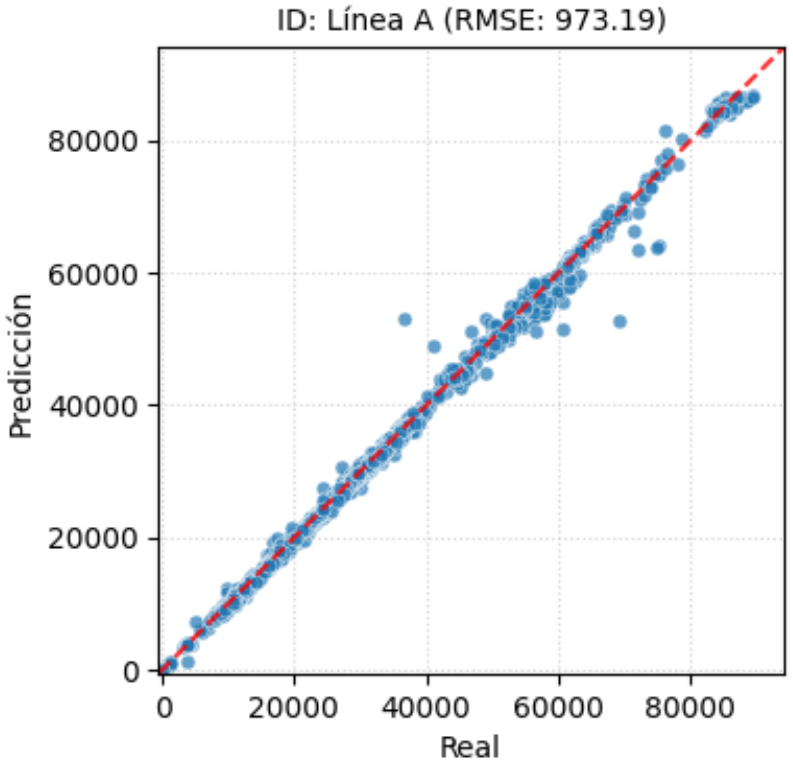


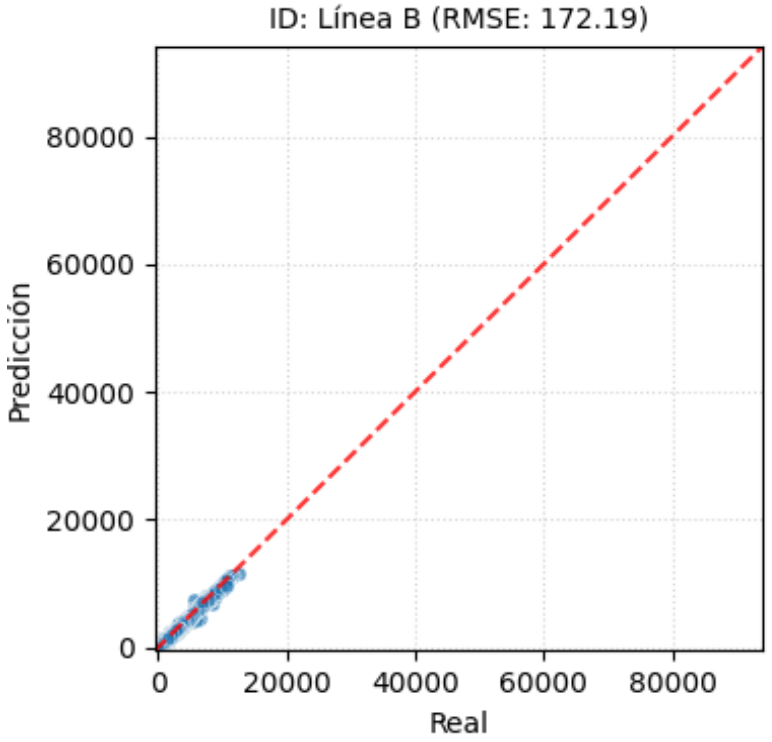
Esta evaluación se hizo para comparar con el modelo *XGBoost* y decidir cuál es mejor. Ahora bien, como se mencionó en la sección de entrenamiento, el modelo con el mejor desempeño fue ***XGBoost con ajuste de hiperparámetros***, por lo cual se procedió a realizar la evaluación en el conjunto de prueba o **TEST**, tras el cual se obtuvo un **MSE** de

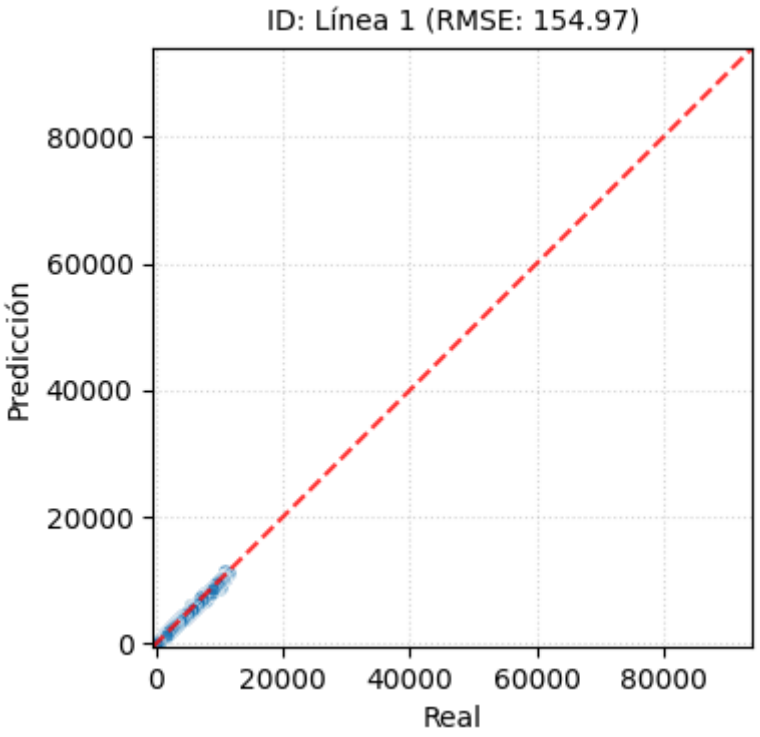
86856.19 y un **RMSE** de **294.7137**. Estas métricas no distan mucho de las obtenidas en el conjunto de validación o **VALIDATION** durante el entrenamiento, lo que refleja que el modelo generaliza muy bien y no está sobreajustado con los datos de entrenamiento. En cuanto a las predicciones, el siguiente diagrama de dispersión refleja que los valores predichos, a pesar de tener algunos errores relativamente grandes en algunas fechas, fueron muy cercanos a los reales:



No obstante, esto es a nivel general, pero si se ven las líneas por separado, se puede observar que el RMSE varía mucho de una línea a otra, haciendo una mención especial a la **línea A** la cual, al ser la más concurrida, presenta un **RMSE** de **973.9**, bastante superior al de las líneas en general, pero comparado con los altos números de pasajeros que transitan por dicha línea diariamente el error no es tan significativo. En las siguientes gráficas podemos ver las 3 líneas más concurridas del Metro de Medellín con su respectivo RMSE y su diagrama de dispersión:

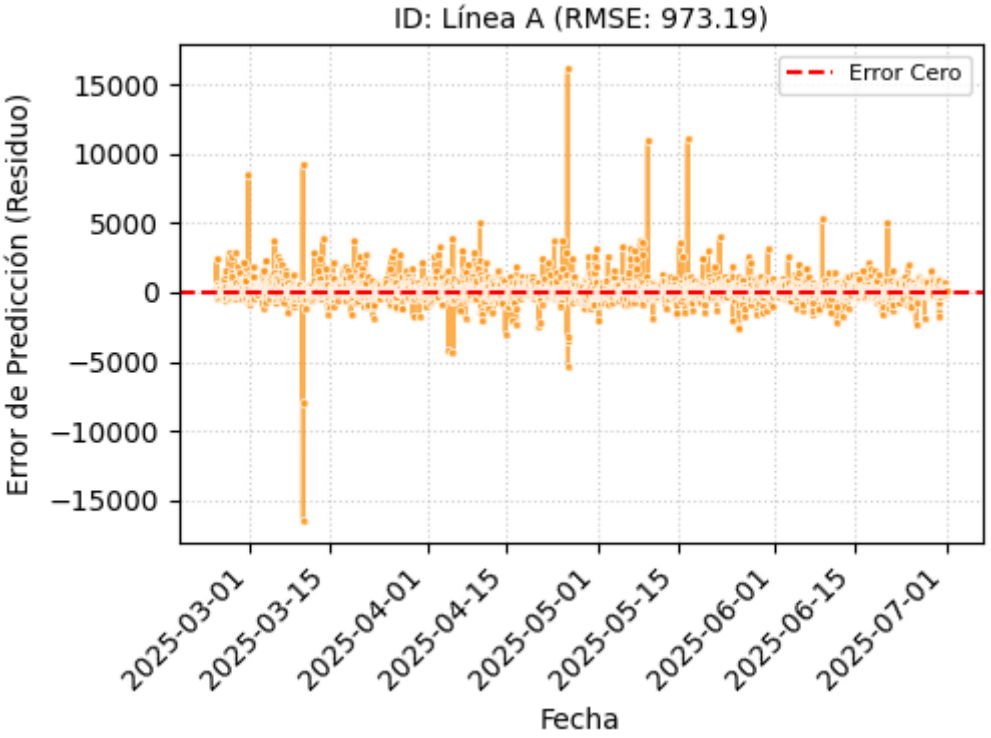

ID: Línea A (RMSE: 973.19)
Predicción
Real
0
20000
40000
60000
80000

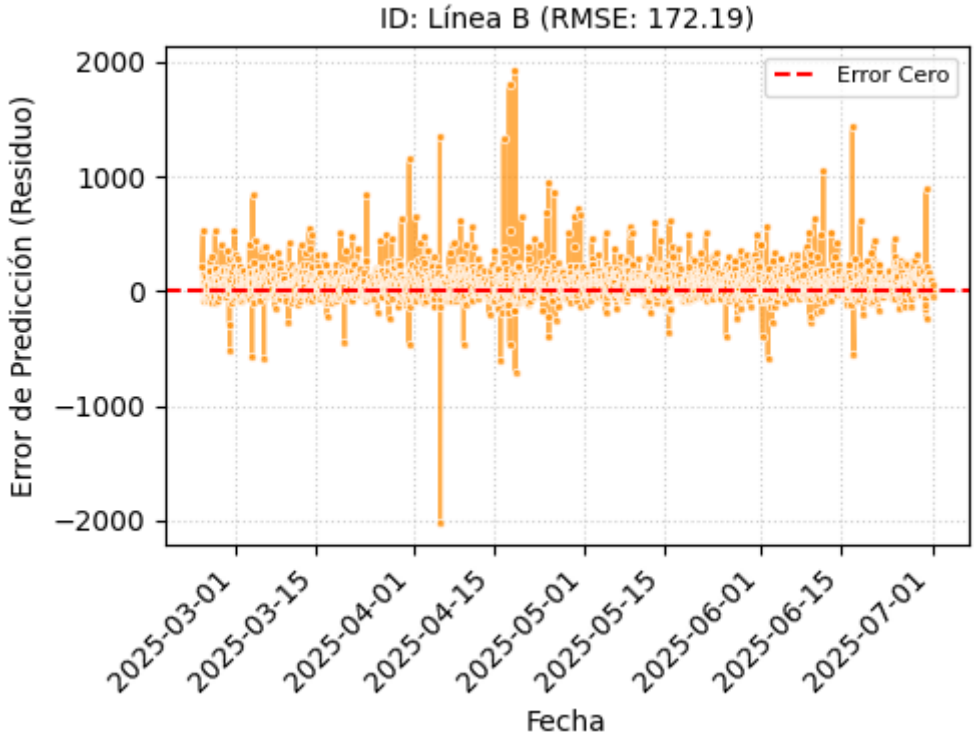

ID: Línea B (RMSE: 172.19)
Predicción
Real
0
20000
40000
60000
80000

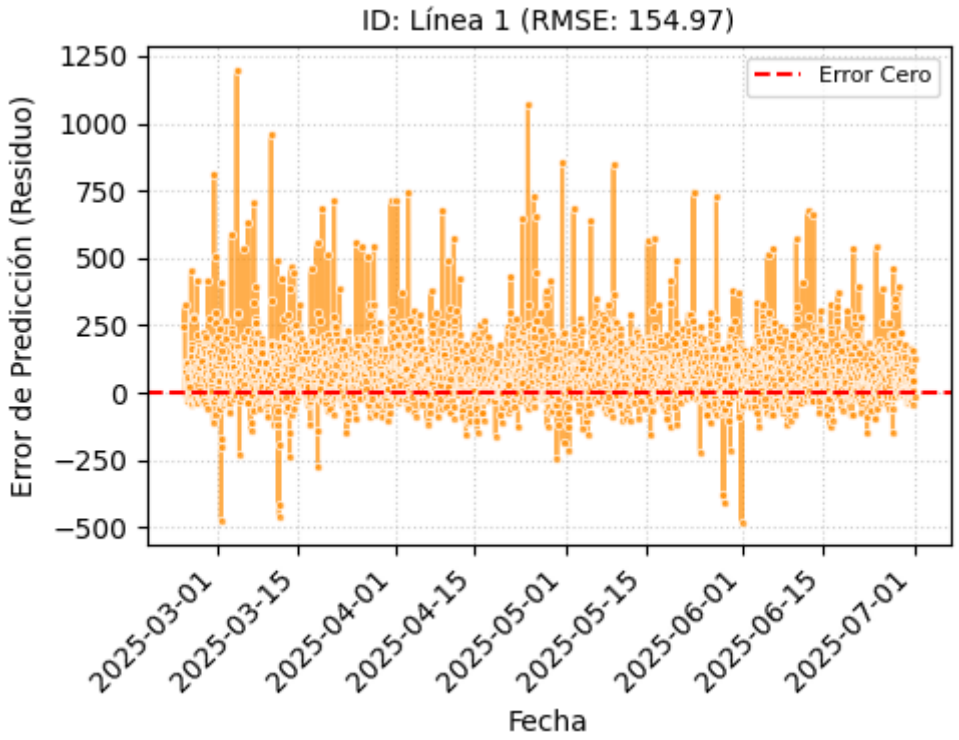


En dichas gráficas se puede ver que entre mayor es el flujo de personas mayor es el error en la predicción y por tanto el RMSE es mayor. No obstante, estos errores no son muy significativos y se puede ver que se mantienen muy cercanos al valor real, con una dispersión muy ajustada (exceptuando la línea A que sí tiene algunos valores bastante alejados de la línea Y=X), por lo que se puede decir que, en términos generales, el modelo es bastante bueno.

Finalmente, para ver mejor los errores, se crearon los siguientes gráficos de comparación de residuos entre el valor predicho y el valor real para cada línea de servicio, en los cuales se puede ver que los errores son relativamente pequeños a comparación de la cantidad de personas que circulan por dichas líneas diariamente, ya que la escala de los valores es baja y, además, muestran que el modelo no tiene un sesgo marcado que muestre una tendencia a fallar por sobreestimar o por subestimar los valores predichos (por facilidad se muestran las mismas 3 líneas de servicio mencionadas anteriormente, es decir, línea A, B y 1):


ID: Línea A (RMSE: 973.19)
Error Cero
Error de Predicción (Residuo)
15000
10000
5000
0
−5000
−10000
−15000
2025-03-01
2025-03-15
2025-04-01
2025-04-15
2025-05-01
2025-05-15
2025-06-01
2025-06-15
2025-07-01
Fecha


ID: Línea B (RMSE: 172.19)
Error Cero
Error de Predicción (Residuo)
2000
1000
0
−1000
−2000
2025-03-01
2025-03-15
2025-04-01
2025-04-15
2025-05-01
2025-05-15
2025-06-01
2025-06-15
2025-07-01
Fecha



TECNOLOGÍA: INGENIERÍA DATOS

Desarrollo del proyecto

El desarrollo del proyecto se realiza en un entorno de análisis de datos basado en Python, utilizando principalmente Jupyter Notebook y VS Code como ambiente de trabajo. Las librerías centrales son:

- pandas y numpy para la ingesta, limpieza y transformación de los datos históricos de afluencia.
- matplotlib y seaborn para la visualización exploratoria.
- statsmodels para los modelos de series de tiempo (ARIMA/SARIMA).
- scikit-learn para los modelos de Machine Learning supervisado.
- holidays para la generación automática de la variable de días festivos en Colombia.
- sktime para realizar la separación de los conjuntos de entrenamiento y prueba respetando la estructura temporal de las series (sin mezclar futuro con pasado).

El flujo de trabajo sigue una lógica de pipeline de datos: lectura de los archivos originales, estandarización de encabezados, transformación a formato largo (día–línea–hora), creación de variables temporales y consolidación del dataset analítico, sobre el cual se ejecutan el EDA y los modelos de predicción, utilizando particiones temporales adecuadas para su evaluación.

Despliegue del proyecto

En un escenario real dentro del Metro de Medellín, se plantea una arquitectura lógica donde los modelos desarrollados se integran a una plataforma analítica corporativa. El flujo general sería:

1. Captura continua de los registros de validación de pasajes en estaciones y líneas (torniquetes, tarjetas Cívica, etc.).
2. Ingesta y almacenamiento centralizado de estos registros en un lago de datos o data warehouse.
3. Procesamiento programado (por ejemplo, cada 15 minutos o de forma horaria) para actualizar las series de afluencia y generar predicciones de corto plazo por línea y franja horaria.
4. Exposición de resultados a través de dashboards y/o APIs internas que permitan a las áreas de Operaciones y Planeación consultar la demanda esperada y tomar decisiones sobre frecuencias y recursos.

En este contexto, los modelos construidos en Python podrían empaquetarse como servicios (por ejemplo, contenedores Docker) e integrarse a un orquestador de tareas (Airflow u otro) que ejecute el pipeline de forma periódica.

Aspectos de ingeniería de datos

- Fuentes de datos y naturaleza
  - ❖ En el proyecto académico, la fuente principal es un archivo Excel histórico (2019–2025) con afluencia por línea y franja horaria: datos estructurados, de naturaleza batch, que se actualizan a intervalos largos.
  - ❖ En un escenario real, las fuentes incluirían:
    - ➔ Registros transaccionales de ingreso de pasajeros (validadores, torniquetes) en tiempo casi real.
    - ➔ Tablas maestras de líneas, estaciones y calendario.
    - ➔ Posibles fuentes externas como clima o eventos especiales (estructuradas o semiestructuradas).

- Ingesta de datos (batch / streaming)
  - ❖ En el proyecto actual, la ingesta se realiza en modo batch, leyendo periódicamente los archivos Excel y transformándolos a un formato analítico mediante scripts de Python.
  - ❖ A nivel productivo, se podría combinar:
    - ➔ Ingesta batch nocturna para recalcular históricos completos.
    - ➔ Ingesta near-real-time / streaming desde la base transaccional o una cola de eventos (por ejemplo, Kafka), para actualizar predicciones intra-día.

- Almacenamiento
  - ❖ Lagos de datos (data lake): almacenamiento de los datos crudos y los datasets enriquecidos en formatos como Parquet/CSV, organizados por año, línea y estación.
  - ❖ Bases de datos: uso de un data warehouse relacional (por ejemplo, PostgreSQL, BigQuery o similar) para las tablas de afluencia agregada y las variables derivadas que alimentan los modelos y los dashboards.
  - ❖ Archivos: generación de archivos planos (CSV/Excel) y reportes para intercambio con otras áreas y para respaldos puntuales.

- Ambiente de procesamiento
  - ❖ Para el trabajo académico, el procesamiento se efectúa en un único equipo (local o en la nube) usando pandas y las librerías de Python mencionadas.
  - ❖ En un escenario de mayor escala, el procesamiento podría migrarse a un entorno distribuido como Apache Spark, permitiendo trabajar con datos a nivel de evento (por validación) y con horizontes temporales más amplios sin sacrificar tiempos de respuesta.

- Aplicaciones (visualización, API, archivos)
  - ❖ **Visualización:** construcción de tableros en herramientas como Power BI, Tableau o una aplicación web (por ejemplo, Streamlit) para mostrar:
    - ➔ mapas de calor por hora y día,
    - ➔ curvas de tendencia,
    - ➔ comparación entre demanda real y predicha por línea.
  - ❖ **APIs:** exposición de un servicio REST que reciba como entrada (fecha, hora, línea) y devuelva la afluencia esperada, de forma que otros sistemas internos (planificación de operación, información al usuario) puedan consumir las predicciones.
  - ❖ **Archivos y reportes:** exportación periódica de reportes en CSV/Excel y gráficos en PDF para soporte de decisiones estratégicas y presentaciones a la alta dirección.

CONCLUSIONES

La demanda del Metro de Medellín demuestra ser altamente predecible, pues la afluencia presenta patrones consistentes por hora, día de la semana y tipo de día. Esta regularidad permite identificar con claridad las horas pico, las horas valle y las líneas con mayor concentración de pasajeros, siendo la Línea A la más crítica. El comportamiento sistemático del sistema confirma que la afluencia no es aleatoria y que existen dinámicas recurrentes que pueden modelarse con alto nivel de precisión.

Durante el proceso de modelado se evidenció que la variable “Es_Festivo” ejerce un impacto importante en la afluencia y contribuye a mejorar el desempeño de los modelos

cuando se incorpora como regresor. Su inclusión permitió capturar variaciones asociadas a días especiales, lo que favoreció una mayor precisión en las predicciones frente a modelos que no consideran información exógena.

Los modelos tradicionales de series de tiempo, como ARIMA, SARIMA y SARIMAX, lograron capturar parte de la estacionalidad inherente al sistema; sin embargo, presentaron limitaciones ante la alta variabilidad diaria y la presencia de patrones no lineales. Aunque SARIMAX mostró el mejor desempeño dentro de este grupo, su capacidad predictiva fue inferior a la de los modelos de machine learning, especialmente frente a fluctuaciones abruptas o comportamientos irregulares.

Por su parte, el modelo XGBoost demostró ser el más robusto y eficiente, especialmente después del ajuste de hiperparámetros. Presentó el menor error (RMSE), mantuvo un rendimiento consistente entre validación y prueba y no mostró señales de sobreajuste. Esto evidencia su capacidad para capturar relaciones complejas y no lineales en la dinámica de la demanda del Metro.

En un escenario real, la implementación de este tipo de modelos permitiría anticipar congestiones, ajustar la frecuencia de los trenes, redistribuir recursos y tomar decisiones operativas con enfoque preventivo y no reactivo. Esta capacidad predictiva se convierte en un insumo valioso para mejorar la planificación y la calidad del servicio ofrecido a los usuarios.

En conjunto, los resultados permiten concluir que el modelo final basado en XGBoost constituye una herramienta sólida y viable para apoyar la toma de decisiones en el Metro de Medellín, generando pronósticos confiables por línea y franja horaria. Su aplicación práctica podría representar un avance significativo hacia una movilidad más inteligente, eficiente y basada en evidencia.

REFERENCIAS

- Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.
- Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts.
- TRB. (2013). *Transit Capacity and Quality of Service Manual* (3rd ed.). Transportation Research Board.
- Wang, B., & Chen, X. (2018). Forecasting of short-term metro ridership with SARIMA models. *Complexity*, 2018, 3189238.