Modelar el Caos: Predicción de Arribos en un Sistema Urbano Complejo

Matteo Musacchio, Tiziano Demarco

***Resumen*—El sistema Ecobici de la Ciudad de Buenos Aires requiere herramientas de predicción que ayuden a anticipar la demanda en sus estaciones para optimizar la redistribución de bicicletas. En este trabajo se aborda el problema de predecir la cantidad de arribos a cada estación en intervalos de 30 minutos, utilizando únicamente información disponible al momento del despacho, como el historial de partidas, características temporales y estadísticas agregadas del sistema. Se construyó un dataset en formato de series temporales y se evaluaron distintos modelos supervisados, incluyendo XGBoost, LightGBM y redes neuronales densas. Además, se exploró la pérdida de Poisson para modelar la naturaleza discreta del problema. Si bien el modelo basado en XGBoost con pérdida de Poisson logró las mejores métricas ($MAE = 0{,}376$, $R^2 = 0{,}45$), los resultados reflejan la complejidad inherente del sistema, caracterizado por alta dispersión horaria y escasa correlación entre estaciones, lo que limita la capacidad predictiva incluso con modelos avanzados. Se discuten posibles mejoras metodológicas y formas de simplificar el problema para mejorar estos resultados.**

I. Introducción

El sistema de bicicletas públicas de la Ciudad de Buenos Aires, más conocido como *Ecobici*, forma parte de una estrategia para fomentar medios de transporte sustentables. Para garantizar su funcionamiento eficiente, es clave contar con herramientas que permitan anticipar la demanda en las distintas estaciones, evitando desbalances en la disponibilidad de bicicletas.

En este trabajo se aborda el problema de predecir la cantidad de arribos que tendrá cada estación en una ventana temporal futura, utilizando información de las partidas ocurridas en una ventana previa. Es decir, se busca estimar cuántas bicicletas llegarán a cada estación en los próximos ΔT minutos, sin conocer los destinos específicos de los viajes al momento de su inicio. Además, se asume que el total de bicicletas en el sistema se conserva, es decir, cada bicicleta que parte eventualmente arriba a otra estación.

Resolver este problema permitiría mejorar la redistribución de bicicletas, reducir costos logísticos y brindar un mejor servicio a los usuarios.

La entrada al modelo consiste en variables históricas del sistema, tales como la cantidad de partidas por estación, información sobre los usuarios, el modelo de bicicleta, variables temporales, y estadísticas agregadas a través de *feature engineering*. Con esta información se entrenaron distintos algoritmos de aprendizaje automático como *XGBoost*, *LightGBM* y redes neuronales densas, con el objetivo de predecir la cantidad de arribos futuros.

mmusacchio@udesa.edu.ar, demarcot@udesa.edu.ar

El informe detalla el proceso completo: desde el análisis exploratorio del conjunto de datos hasta la construcción del dataset de entrenamiento, el diseño de modelos y la evaluación de *performance*, destacando los desafíos y posibles líneas de mejora del enfoque planteado.

II. Conjunto de datos

II-A. Análisis exploratorio

El conjunto de datos utilizado para el desarrollo es un dataset público disponible en el sitio web *Buenos Aires Data*. Se utilizaron específicamente los datos de recorridos realizados desde el año 2020 al 2024, y los registros de *Usuarios Ecobici* correspondientes a esos años. Se reservaron los datos de viajes realizados a partir de Septiembre 2024 para usar como set de prueba del modelo en una instancia posterior de testeo del modelo.

Los datos de recorridos realizados está compuesto de forma que cada fila es un viaje realizado por un usuario desde una estación origen a una estación destino, con datos de fecha y hora de inicio y finalización del viaje, ubicación de las estaciones involucradas y el modelo de bicicleta. Además, cada usuario tiene un ID asociado y otras fuentes como género y fecha de alta al servicio.

En una primera instancia se limpió y se analizó el conjunto de datos para el entendimiento a profundidad del problema a resolver. Se notó que los viajes del año 2020 tuvieron registros anormales (ver Figura 1) respecto a otros años, por lo que se decició omitir los viajes de ese año para el entrenamiento del modelo.

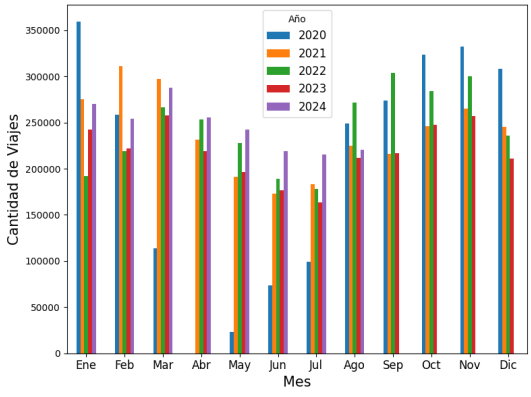

Figura 1: Distribución mensual de viajes por cada año del dataset

Por otro lado, se omitieron los viajes menores a un minuto, ya que se identificaron numerosos casos de viajes

de duración cero o de pocos segundos que carecían de sentido práctico. Al analizar estas muestras, se observó que generalmente consistían en bicicletas retiradas y devueltas de inmediato en la misma estación, probablemente por usuarios que se arrepintieron al instante de utilizar el servicio. Estos registros fueron considerados *outliers* y eliminados del dataset (ver Figura 2).

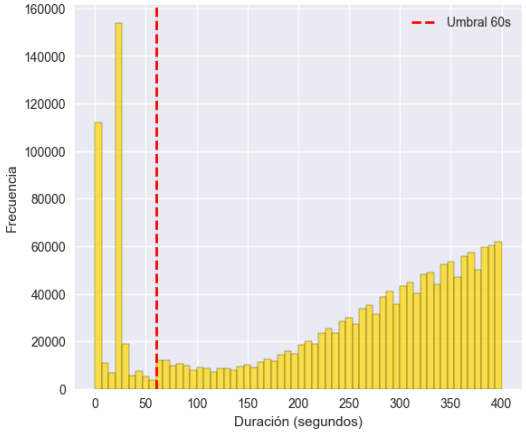

Figura 2: Frecuencia de viajes vs Duración en segundos menores a 400 segundos

También se analizó la cantidad de viajes por hora del día (ver Figura 3), lo cual permitió identificar en qué hay rangos horarios la cantidad de viajes realizados es alta, y en cuáles es más baja. Esta noción va a ser útil para entender la creación de ventanas temporales en la próxima sección.

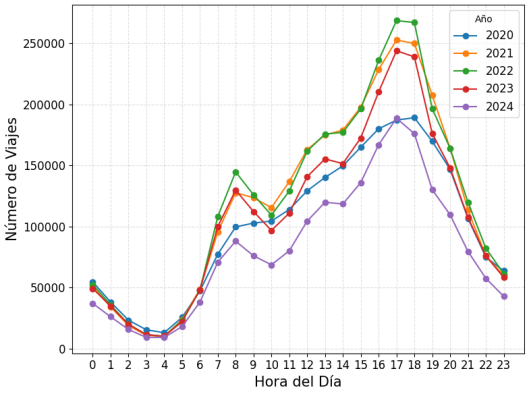

Figura 3: Intensidad de viajes por hora del día

Para explorar posibles patrones entre estaciones, se construyó una matriz de adyacencia donde cada celda A_{ij} indica la cantidad de viajes desde la estación i hacia j. La Figura 4 muestra esta matriz como un mapa de calor.

El objetivo fue identificar relaciones entre pares de estaciones que pudieran aportar expresividad al modelo para facilitar la predicción. Sin embargo, la visualización no muestra agrupamientos ni patrones consistentes, lo que indicaría que los flujos entre estaciones son muy variables y sin una estructura fija. Esto podría sugerir el nivel de dificultad de construir un modelo que logre captar esta variabilidad de forma precisa.

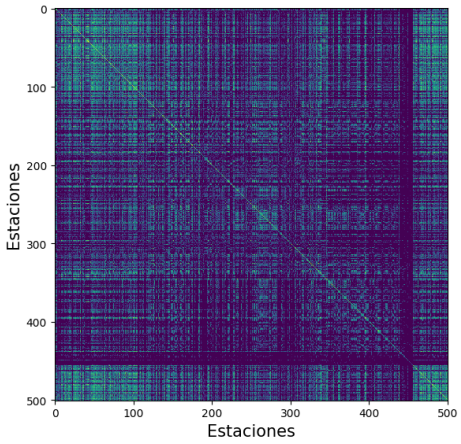

Figura 4: Matriz de adyacencia entre estaciones (cantidad de viajes entre pares de estaciones) en escala logarítmica

II-B. Preprocesamiento de datos

En esta sección se detallan las transformaciones realizadas sobre el conjunto original para la construcción de las variables aleatorias *input* X y *target* Y.

Volviendo al objetivo del desarrollo, dados los viajes iniciados en una ventana temporal $[T - \Delta T, T]$, se quieren predecir los arribos para cada estación en la siguiente ventana temporal $[T, T+\Delta T]$. Sin embargo, tras intentos iniciales de entrenar modelos con dicha información, el rendimiento fue bajo. Esto motivó incorporar mayor contexto histórico de ventanas temporales anteriores para capturar patrones más relevantes.

Se construyó un nuevo conjunto de datos en formato de series temporales, eligiendo $\Delta T = 30$ min. La variable X *input* se trata de una matriz de datos donde cada fila X_i representa a una estación en una ventana temporal determinada, cuyos features incluyen:

- Cantidad de viajes iniciados desde la estación en la ventana actual.
- Estadísticas de duración de viajes de ventanas anteriores (media y desviación estándar).
- Edad promedio y proporción por género de usuarios.
- Modelo de bicicleta más frecuente.
- Variables temporales: hora de inicio, día de semana, mes, etc.
- Información de contexto histórico agregando *lags* de una a seis ventanas previas para algunas de estas variables.
- Cantidad de arribos en la ventana anterior.

El vector *target* Y contiene la cantidad de arribos a cada estación en la siguiente ventana temporal, emparejando cada registro de X con su valor futuro correspondiente, alineado por estación e intervalo de tiempo.

Para construir este dataset, se realizó una limpieza exhaustiva de datos de múltiples años, armonizando tipos y nombres de columnas que variaban significativamente entre períodos, y concatenando estos con los datasets de usuarios provistos de forma separada, garantizando consistencia en toda la integración.

Para dividir los datos en *train* y *validation*, se seleccionó aleatoriamente un día por semana para validación, permitiendo una representación temporal variada en ambos conjuntos y simulando un escenario de generalización temporal más realista. El rango temporal abarcado fue desde el 2021-01-01 al 2024-08-31, resultando en un dataset de entrenamiento de tamaño $(21{,}243{,}250)$ (83 %) y un conjunto de validación de tamaño $(4{,}336{,}608)$ (17 %) y se aplicó una normalización utilizando *Z-score* sobre cada columna numérica para facilitar el entrenamiento de los modelos y mejorar la estabilidad de los algoritmos de aprendizaje.

Una desventaja de esta estrategia de representación en la que cada fila de X corresponde a una combinación `(estación, ventana temporal)` es que, debido a la distribución de horarios observada en la Figura 3, muchas ventanas presentan muy poca actividad. Esto genera una matriz de entrada altamente dispersa, con una gran cantidad de ceros. Por un lado, esto es deseable, ya se espera que el modelo aprenda el flujo real de los viajes realizados, incluyendo franjas horarias con baja demanda pero por otro lado, complicaría el entrenamiento del modelo al "disipar"la información útil entre una gran cantidad de casos irrelevantes.

Dado que la cantidad total de observaciones es el producto entre el número de días con registros de viajes entre todos los años del dataset, la cantidad de ventanas por día y el número de estaciones, el dataset final resulta masivo pero con una baja densidad informativa. Esta situación dificultaría la capacidad del modelo para captar patrones significativos.

II-C. Simplificación del Problema

Además, se exploraron otras formas de estructurar el conjunto de datos para evitar la dispersión de los datos y condensar la información relevante. Por un lado, se intentó filtrar las combinaciones `(estación, ventana)` para conservar únicamente aquellas que presentaban actividad (es decir, con al menos una partida o arribo), con el objetivo de eliminar registros nulos. Por otro lado, se experimentó con restringir el problema a la predicción de arribos en una única estación (en particular, aquellas con mayor concurrencia) utilizando como entrada las ventanas de todas las estaciones.

III. METODOLOGÍA

III-A. Algoritmos utilizados

XGBoost Regressor: Algoritmo de boosting que entrena árboles secuencialmente para corregir errores del modelo anterior. La predicción se construye como suma de árboles:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F}$$

Se usaron funciones de pérdida cuadrática y de Poisson:

$$l(y, \hat{y}) = \hat{y} - y \log(\hat{y})$$

Red Neuronal Feed Forward: Modelo compuesto por capas densas con activación ReLU. Cada capa transforma la salida anterior como:

$$h^{(l)} = \text{ReLU}(W^{(l)} h^{(l-1)} + b^{(l)})$$

y la salida final es $\hat{y} = h^{(L)}$. Se entrenó con pérdida MSE y mediante descenso por gradiente con optimizador Adam.

LightGBM: Variante de boosting que optimiza velocidad y memoria usando histogramas discretizados. También se modela como suma de árboles:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i)$$

y se evaluó con pérdida cuadrática y de Poisson, al igual que en XGBoost.

III-B. Selección de hiperparámetros

Dada la limitación de recursos, se realizó una búsqueda manual de hiperparámetros en XGBoost, probando distintas combinaciones de `n_estimators` (500, 1000, 1500), `learning_rate` (0.1, 0.05) y `max_depth` (5 a 15). Se observó que, gracias al *early stopping*, convenía usar mayor cantidad de estimadores, mientras que un `learning_rate` de 0.05 y una profundidad de 10 ofrecieron el mejor desempeño en validación. Los valores finales utilizados fueron:

```
'n_estimators': 1500, 'max_depth': 10,
'learning_rate': 0.05, 'subsample': 0.8,
'colsample_bytree': 0.8, 'max_bin': 128,
```

Esta estrategia permitió ajustar el modelo de forma eficiente priorizando estabilidad y capacidad de generalización.

III-C. Métricas de rendimiento

Para evaluar los modelos de predicción de arribos por estación se utilizaron las siguientes métricas:

Mean Squared Error (MSE): mide el error cuadrático medio entre predicciones y valores reales, penalizando más los errores grandes:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

Mean Absolute Error (MAE): calcula el error absoluto promedio, indicando cuántas bicicletas en promedio erra el modelo:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|$$

Coeficiente de determinación (R^2): indica qué proporción de la variabilidad de los arribos es explicada por el modelo:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}$$

Feature Importance: si bien no es una métrica de evaluación directa, se utilizó para analizar la contribución de cada

característica al modelo entrenado, permitiendo interpretar cuáles variables tienen mayor impacto en la predicción de arribos y orientar decisiones de *feature engineering*.

Estas métricas permiten comparar de forma consistente el desempeño de los distintos modelos entrenados.

IV. Resultados

IV-A. Modelo M0

Como *baseline*, se entrenó un XGBoost de regresión utilizando el dataset con filas por estación e intervalos de 30 minutos, prediciendo los arribos dados los despachos del intervalo anterior, cumpliendo con la tarea fundamental y sin *feature engineering* adicional. El entrenamiento se detuvo con *early stopping* en la iteración 696, con un $R^2 = 0{,}4583$ en *train* y $R^2 = 0{,}4227$ en *validation*, y un *MAE* de 0.4069 y 0.4125 respectivamente. La variable más importante resultó ser la cantidad de despachos previos, como se hubiese esperado.

En la Figura 5 se observa una tendencia creciente entre las predicciones y los valores reales, aunque con alta dispersión, confirmando la presencia de un nivel elevado de ruido en el *target*, consistente con la intuición de que los movimientos de los usuarios en el sistema presentan una alta variabilidad muy difícil de capturar.

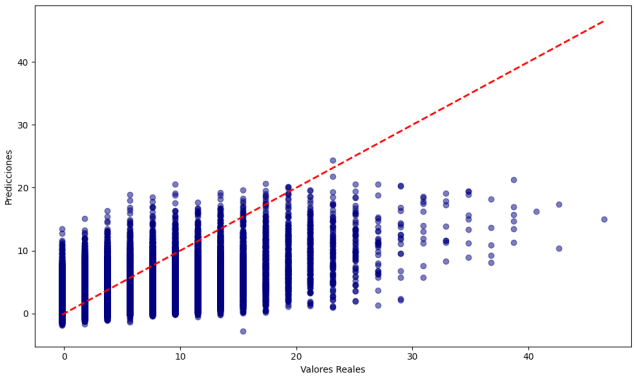

Figura 5: Predicciones versus valores reales para el conjunto de validación en el modelo M0.

IV-B. Modelo M1

Luego se incorporó *feature engineering* con características de intervalos anteriores para explorar dependencias temporales, como una extensión al planteo fundamental del problema. Además, se agregó una *feature* de *cluster* geográfico utilizando K-means para capturar patrones espaciales. Esto resultó en una mejora con $R^2 = 0{,}4458$ en *validation*, y un *MAE* de 0.3967.

En la Figura 6 se ve que las nuevas variables generadas figuran entre las más relevantes, indicando que aportaron información valiosa al modelo. Sin embargo, como se evidencia en las métricas y en la dispersión de las predicciones, el resultado sigue siendo insatisfactorio, resaltando la dificultad del problema y la necesidad de explorar otras estrategias de modelado para mejorar el rendimiento.

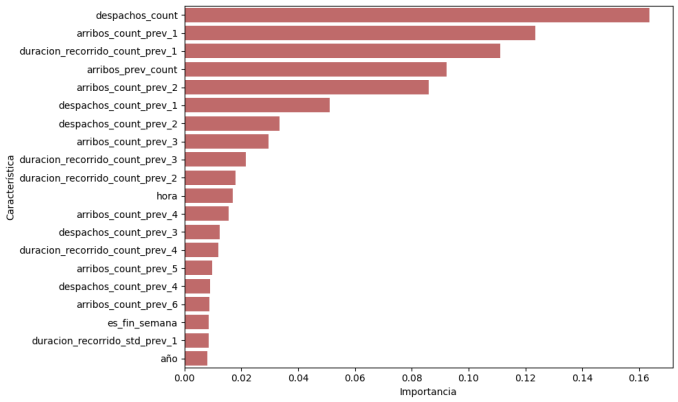

Figura 6: Importancia de características tras la incorporación de información de intervalos anteriores (modelo M1).

IV-C. Modelo M2

Finalmente, se utilizó un modelado de Poisson, más adecuado para variables de conteo no negativas, en donde la cantidad de arribos por estación en un intervalo de tiempo puede ser interpretada como la ocurrencia de eventos independientes en una ventana temporal fija. Esto sugiere que una distribución de Poisson puede modelar de forma más natural este fenómeno, facilitando el aprendizaje del modelo sobre la estructura subyacente de los datos. Este enfoque mejoró levemente las métricas a un $R^2 = 0{,}4508$ en *validation*, con un *MAE* de 0.3756, mostrando un ajuste más preciso a la magnitud de los arribos.

IV-D. Comparación de modelos

En la Tabla I se resumen los resultados en el conjunto de validación para todos los modelos evaluados, incluyendo LightGBM y MLP entrenados con la misma estructura de datos para referencia, que al notar que el desempeño de ambos es menos preciso que el modelo M2, se dejaron de tener en cuenta.

Cuadro I: Comparación de modelos en el conjunto de validación.

Modelo	**MAE**	**MSE**	$\mathbf{R^2}$
XGBoost M0 (baseline)	0.4125	0.5772	0.4227
XGBoost M1 (features)	0.3967	0.5540	0.4458
XGBoost M2 (Poisson)	0.3756	0.5075	0.4508
LightGBM	0.4031	0.5647	0.4351
MLP	0.3901	0.5422	0.4402

IV-E. Simplificación del problema

Volviendo a lo planteado en la Sección II-C, se exploraron dos variantes para simplificar la estructura del problema y reducir la dispersión de datos en el dataset.

Por un lado, se evaluó restringir la predicción de arribos a una única estación utilizando como entrada las ventanas

de todas las estaciones. Los resultados variaron significativamente según la estación elegida, observándose que aquellas con mayor cantidad de recorridos tendían a presentar métricas más estables. En general, se obtuvieron valores de R^2 en validación entre 0.30 y 0.44, sin mostrar una mejora clara frente a los modelos presentados previamente, lo que indica que la variabilidad entre estaciones sigue siendo un factor relevante para la dificultad del problema.

Por otro lado, se probó eliminar del dataset todas las filas correspondientes a intervalos sin actividad para concentrar el entrenamiento en ventanas temporales activas. Sin embargo, se concluyó que esta estrategia no resulta adecuada, ya que en la práctica es necesario predecir también los intervalos con valores nulos de arribos, por lo que esta simplificación no refleja el escenario real de evaluación. El modelo entrenado con esta estructura mostró un rendimiento bajo, alcanzando apenas un $R^2 = 0{,}33$ en validación, luego de eliminar el 67.3 % del dataset

IV-F. Extensión Web

Como extensión del trabajo, se desarrolló una página web que permite explorar las predicciones generadas de forma interactiva. En la sección *Mapa*, el usuario puede seleccionar una fecha y hora específicas, y visualizar sobre un mapa interactivo el desempeño del modelo para cada estación como se puede ver en la Figura 7.

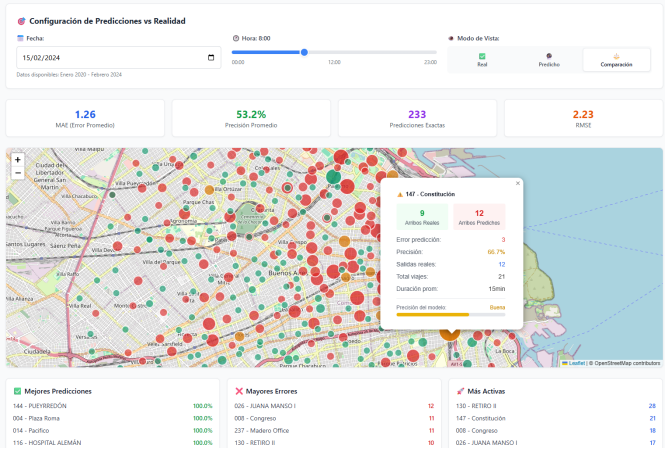

Figura 7: Visualización en la página web de las predicciones del modelo para el día 15 de febrero de 2024 a las 8:00 hs. Al hacer *click* sobre alguna de las estaciones es posible visualizar los arribos reales y los predichos por el modelo, junto a algunas métricas de la estación seleccionada

Las estaciones están representadas mediante círculos de colores que representan la precisión del modelo en su predicción. Los círculos verdes indican una precisión excelente mayor a 80 %, los naranjas una precisión aceptable entre 60 % y 80 % y los rojos una precisión mejorable menor a 60 %.

Además, en la sección *Predicciones*, se puede consultar la predicción puntual de arribos para una estación y momento dados, junto con su porcentaje de confianza y el estado de demanda de dicha estación.

Por otro lado, en las secciones *Resumen* y *Análisis*, se presentan métricas útiles para interpretar el comportamiento global del sistema, tales como flujos horarios, datos históricos, estaciones más demandadas y su disponibilidad. Esta herramienta busca facilitar tanto la exploración del modelo como su uso práctico de los usuarios.

IV-G. Evaluación en Set de Test

Finalmente, se evaluó el *set* de *test* utilizando el modelo M0, dado que era el único que respetaba la consigna fundamental de predicción a partir de la información del intervalo anterior. Los resultados obtenidos fueron un $MSE = 2{,}04$, $MAE = 0{,}95$ y $R^2 = 0{,}27$, reflejando la dificultad intrínseca del problema y el alto nivel de variabilidad en la predicción de arribos en estaciones de bicicletas públicas.

V. Conclusiones

El presente trabajo abordó la tarea de predecir arribos a estaciones del sistema *Ecobici* utilizando información histórica de partidas y contexto temporal. Se evaluaron distintas estrategias de modelado y se observó que, si bien el enfoque basado en XGBoost con feature engineering logró mejoras graduales en las métricas, el problema presenta una alta complejidad estructural debido a la dispersión de los datos, la baja frecuencia en muchas ventanas temporales y la falta de patrones concretos entre estaciones.

Entre los algoritmos utilizados, XGBoost con pérdida de Poisson resultó ser el más efectivo en términos absolutos de error, gracias a su capacidad para modelar variables de arribos de forma más natural. La incorporación de información histórica (*lags*) también demostró ser útil para capturar dinámicas recientes y mejorar la predicción.

Sin embargo, los resultados obtenidos muestran que, incluso con modelos no lineales complejos, existe un límite en la capacidad de predicción dada la variabilidad del sistema. En función de esto, una posible mejora para trabajos futuros sería explorar enfoques como segmentar el modelado por rangos horarios (por ejemplo, entrenar un modelo específico para horas pico y otro para horas de baja demanda), utilizar datos externos como clima o eventos, o bien considerar el análisis del contexto político y económico del país, dado que la cantidad de turistas puede impactar significativamente en el uso del sistema de bicicletas públicas.