IA-vis

Càtedra d'Intel·ligència Artificial i Visualització d'Informació per al Govern Obert

A la Ciutat de València

# Técnicas IA para la predicción de disponibilidad de plazas de parking en Valencia

Francesc Domenech Villalba
Inmaculada Coma Tatay
UNIVERSITAT DE VALÈNCIA| ETSE-UV | REGIDORIA DE TRANSPARÈNCIA I DEFENSA DE LA CIUTADANIA DEL AJUNTAMENT DE VALÈNCIA

# Técnicas IA para la predicción de disponibilidad de plazas de parking en Valencia

## Contenido

# 1. Introducción

En la ciudad de Valencia tenemos un sistema que permite monitorizar el número de plazas disponibles en tiempo real en un conjunto de parkings públicos. Esta información está disponible en la web de datos abiertos del Ayuntamiento de Valencia y es utilizada para mostrarla en los paneles informativos de plazas libres distribuidos en diferentes puntos de la ciudad, lo cuál supone una ayuda en la gestión de la movilidad urbana.

La búsqueda de plazas de aparcamiento puede ser en una fuente significativa de congestión, consumo innecesario de combustible y emisiones contaminantes. En este contexto, surge la necesidad de soluciones inteligentes que optimicen el uso de los espacios de estacionamiento disponibles y mejoren la experiencia de los conductores.

El presente documento aborda el desarrollo y aplicación de algoritmos de predicción de disponibilidad de parkings en Valencia. Estos algoritmos, basados en el análisis de datos históricos, patrones de comportamiento y técnicas de aprendizaje automático, permiten anticipar en tiempo real la ocupación de plazas en diferentes zonas de la ciudad. Su implementación no solo puede contribuir a una movilidad más eficiente, sino que también apoya los objetivos de sostenibilidad y calidad de vida en las áreas metropolitanas.

En este documento se explica el proceso seguido para crear un sistema que predice la disponibilidad de plazas en los parkings de la ciudad de Valencia.

A partir de los datos históricos de uso de los parkings proporcionados por el ayuntamiento, se plantea una solución escalable capaz de predecir la disponibilidad de aparcamientos públicos en la ciudad de Valencia mediante técnicas de Inteligencia Artificial conocidas como *Deep Learning*. En concreto se evalúa el rendimiento de las 3 arquitecturas de redes neuronales recurrentes (RNN, LSTM, GRU) en su versión más básica, frente a sus versiones optimizadas mediante ajuste óptimo de hiperparámetros.

También se hace referencia al trabajo futuro orientado a desarrollar un sistema MLOps, que, partiendo del actual, sea capaz de integrar automáticamente los nuevos datos generados y optimizar de forma continuada sus predicciones mediante despliegues y entrenamientos automatizados.

El código desarrollado está disponible en este trabajo está disponible en:

https://github.com/cescdovi/ts-parking

# 1. Fuente de datos.

El Ayuntamiento de Valencia, en su portal de datos abiertos, publica en tiempo real información sobre la disponibilidad de plazas en los parkings públicos de la ciudad. Sin embargo, no está disponible en abierto el histórico de datos de plazas. Por ello, a través de la oficina de Ciudad Inteligente, nos proporcionan:

- Archivo csv con la información de las plazas de parking de los últimos dos años.
- Archivo csv con el identificador y nombre de los parkings.

Dichos archivos contienen información sobre 20 aparcamientos, que han sido empleados en el análisis que detallamos a continuación. Sin embargo, para la presentación de resultados en este informe se muestran habitualmente 1 ó 2 aparcamientos, por simplicidad, aunque el método es extensible al conjunto completo.

## 2. Depuración de datos

El fichero con la información de las plazas de parking contiene 13 millones de registros, correspondientes a los 20 parkings. Es frecuente que con grandes cantidades de datos haya registros erróneos, con valores no válidos o faltantes. Por ello, el primer paso ha consistido en realizar una inspección inicial y limpieza de datos[1]

El fichero de datos proporcionado vemos que contiene las siguientes columnas:

- **_id**: id (alfanumérico) del parking
- **entityId**: id de parking
- **entityType**: indica que el estacionamiento es de tipo parking
- **availableSpotNumber**: total de plazas disponibles del parking
- **availableSpotPercentage**: porcentaje de plazas disponibles
- **totalSpotNumber**: total de plazas del parking
- **idAparcamiento**: id de parking
- **recvTime**: hora de registro del dato
- **TimeInstant**: hora en la que se recibe el dato

Sobre estos datos realizamos los siguientes procesos:

- Se eliminan todas aquellas columnas redundantes o con muchos datos faltantes, ya que no se van a usar, y nos quedamos con*: availableSpotNumber, totalSpotNumber e idAparcamiento*
- Se realizan las conversiones de tipos necesarias: por ejemplo, corregir registros que hacen referencia al mismo parking, pero tienen identificadores distintos, ya que se han registrado de forma errónea.
- Se corrige otra incoherencia detectada: registros en los que varía el nº de plazas totales del parking. Esto no debería ser así, a no ser que se haya ampliado el parking.
- Si hay pocos registros erróneos para un parking, se corrigen, ya que se interpretan como errores de transmisión, y si hay muchos, se descartan los datos de ese parking, ya que no sabemos la razón real por la que ocurre. Bajo este criterio se eliminan todos los registros correspondientes a los parkings con id 1 y 5.

Posteriormente, se define una función que permite ver el rango temporal para cada parking (ver Figura 1), así como los periodos en los que hay datos faltantes, ya que se interpolan de forma lineal. También se observan los registros erróneos, es decir, aquellos que:

- La disponibilidad es superior al nº de plazas totales del parking
- La disponibilidad es negativa

[1] *Este apartado corresponde al notebook: 1_Prepare_ts.ipynb*

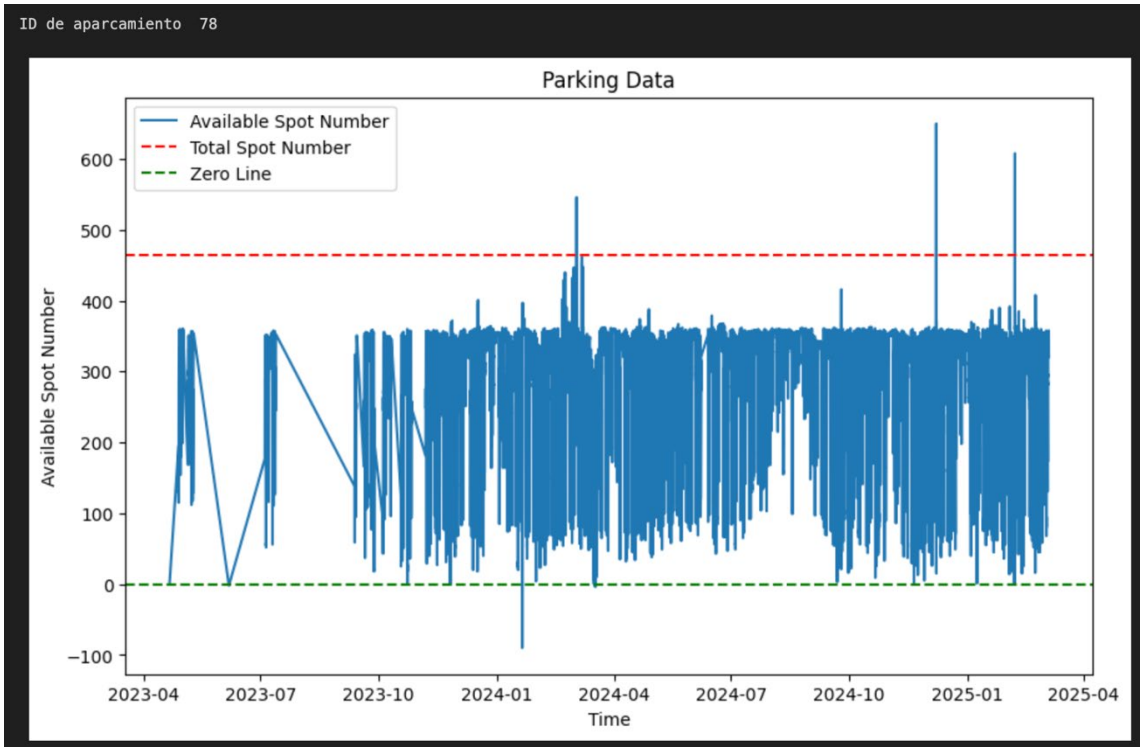


Figura 1: Ejemplo de datos recogidos para el parking con ID = 78 – Severo Ochoa

Se aplica esta función de limpieza de datos para cada aparcamiento, y se eliminan todos los registros asociados a cada parking en los que hay largos períodos de datos faltantes, ya que imputar períodos tan largos puede comprometer la calidad de los resultados. También se eliminan aquellos en los que hay pocos datos o son erróneos. Finalmente, se eliminan los parkings con los siguientes identificadores: -100, 1, 2, 3, 4, 5, 9, 10

Con el análisis realizado hasta ahora, se han detectado los principales problemas en los datos recibidos:

- Registros de un mismo parking asociados a diferentes identificadores, aunque debería ser el mismo
- Registros de un mismo parking donde varía el nº de plazas totales del parking
- Registros erróneos (donde el nº de plazas disponibles es negativo o supera el nº de plazas totales del parking)
- Registros con los tipos mal convertidos.

Posteriormente se continúa con otras correcciones de irregularidades [2], que en problemas de series temporales corresponden a:

- Varios registros asociados a una misma franja horaria. Cuando hay varios registros asociados a una misma hora, en un día concreto, se calcula el promedio de dichos registros.

[2] *Este proceso corresponde al notebook: 2_Preprocess.ipynb*

- Franjas horarias sin registros (valores faltantes, NA). Cuando se detecta una franja horaria sin registros, se imputa usando una media ponderada entre los 4 registros anteriores y ese mismo registro, a esa misma hora, en la semana anterior.

Para ello, se prepara una función, que es una extensión de la que se ha mostrado anteriormente, en la que se muestra un gráfico interactivo (ver Figura 2), en el que en azul se representa la serie temporal, y en rojo, unas rectas discontinuas verticales que permiten identificar visualmente instantes temporales en los que hay datos faltantes. En la parte superior al gráfico, se muestra un conjunto de datos, en el que se visualiza el inicio y el final de cada franja horaria sin datos y a cuantas horas corresponde dicho periodo para facilitar la interpretación y toma de decisiones.

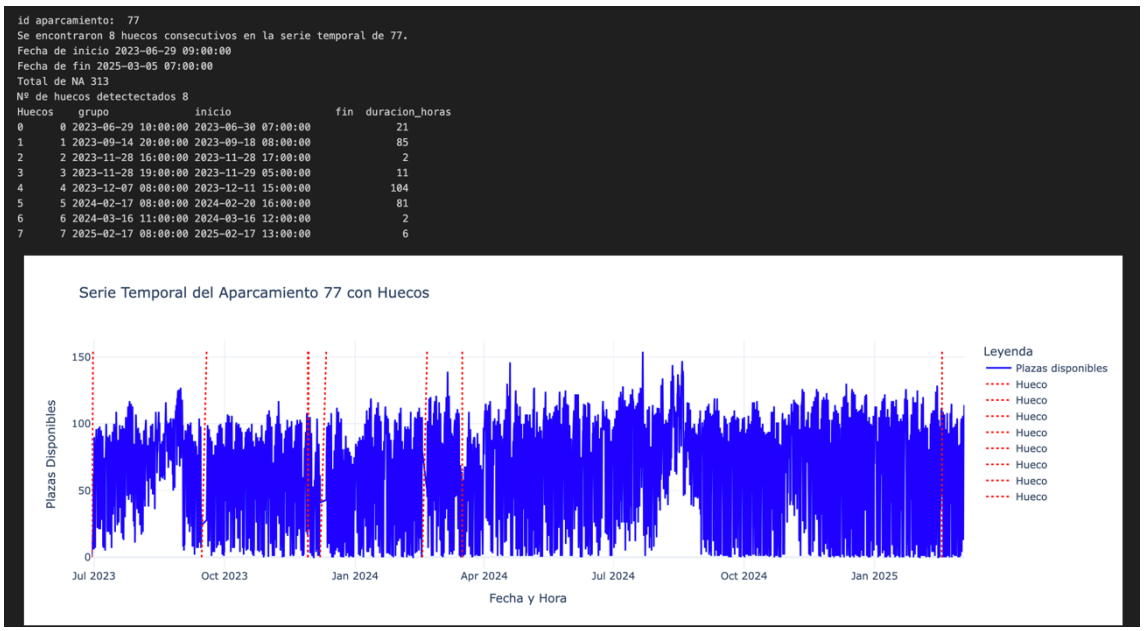


Figura 2: línea temporal para el parking con ID = 77 con detección de huecos faltantes

Esta función permite identificar franjas horarias con datos faltantes, pero también “recortar” la serie temporal, es decir, si presenta muchos valores faltantes al inicio de la serie, se empezarán a usar datos cuando se reduzcan considerablemente los valores nulos, para no comprometer el resultado de las predicciones, ya que imputar largos períodos puede ser peligroso.

Aplicando esta función para cada aparcamiento, nos da pie a eliminar los registros asociados a los siguientes parkings, por ser sospechosos de contener datos erróneos: 50, 79, 120.

En resumen, al final de esta etapa, se obtienen los datos sin irregularidades (un único dato por hora desde el inicio hasta el final de la serie temporal). Recordemos que se mantienen 3 variables: el id de aparcamiento, el porcentaje de plazas disponibles en un rango entre 0 y 100, y el instante temporal al que hace referencia el registro.

## 3. Análisis exploratorio de los datos y periodograma.

Esta etapa permite hacerse una idea de cómo están distribuidos los datos, permitiendo detectar patrones temporales repetitivos y comprender cuándo y cómo cambia la ocupación en el tiempo.[3]

Haremos en primer lugar un **análisis exploratorio de datos** (EDA).

Para ello, se calcula para cada parking cuál es el porcentaje de plazas disponibles entre 0 y 100, agrupando por: día del mes, día de la semana y mes del año.

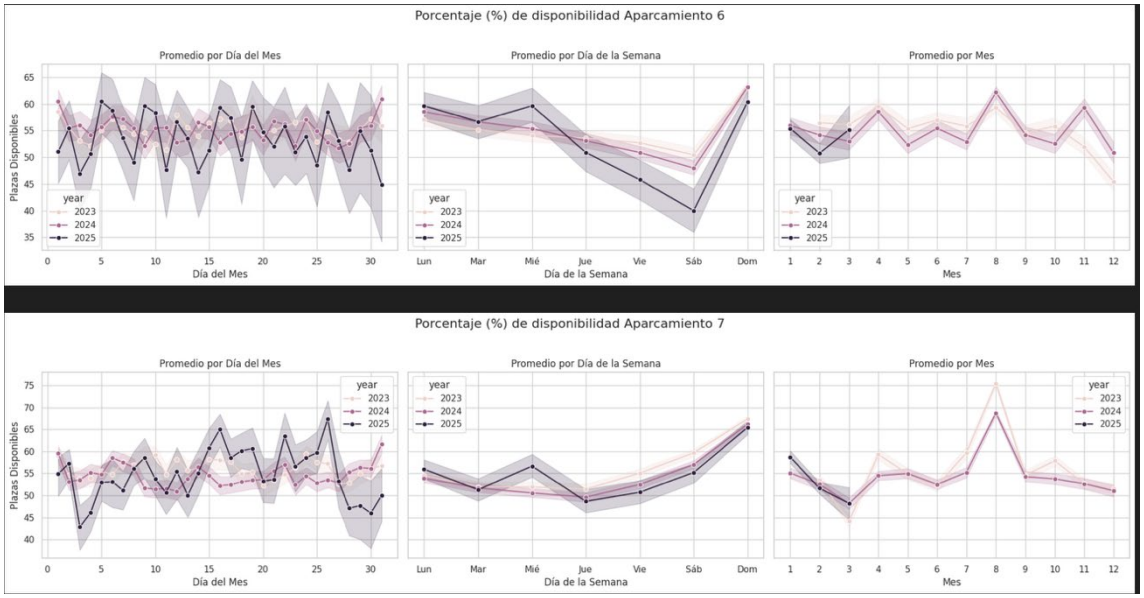


Figura 3: análisis EDA para parkings con ID = 6 (glorieta-paz) e ID = 7 (porta de la mar)

En el caso del aparcamiento con id 6, glorieta-paz, (ver figura 3, superior) se observa cómo la disponibilidad desciende durante los fines de semana, lo que podría suponer su uso para compras y actividades de ocio. Por el contrario, en el mes de agosto la disponibilidad aumenta, reflejando el descenso de la demanda asociado a las vacaciones.

Por otro lado, en el caso del parking con id 7, porta de la mar-colón, (ver figura 3, inferior), observamos como la disponibilidad aumenta tanto el fin de semana, como en agosto. Este patrón apunta a que su uso está concentrado principalmente en horario laboral, registrando una ocupación mucho mayor de lunes a viernes en días hábiles y una caída de la demanda cuando la actividad profesional se reduce (fines de semana y vacaciones).

Hacer un análisis exploratorio de datos es bastante útil, pero muchas veces no es suficiente. Por ello se propone calcular el **periodograma**: se trata de una herramienta que permite transformar señales del dominio temporal al dominio de la frecuencia, permitiendo ver como de importante es cada frecuencia para cada serie temporal.

Al calcular el periodograma para el parking con id 6, observamos como hay un comportamiento cíclico cada 24, 28, y 12 horas. Cada uno de ellos puede asociarse a:

[3] *Este apartado corresponde al notebook: 3_EDA.ipynb*

- 24 horas: inicio de un nuevo día (dentro de la rutina)
- 28 horas: trabajos a media jornada, salidas de colegios…
- 12 horas: final de la jornada laboral, transición entre turnos…

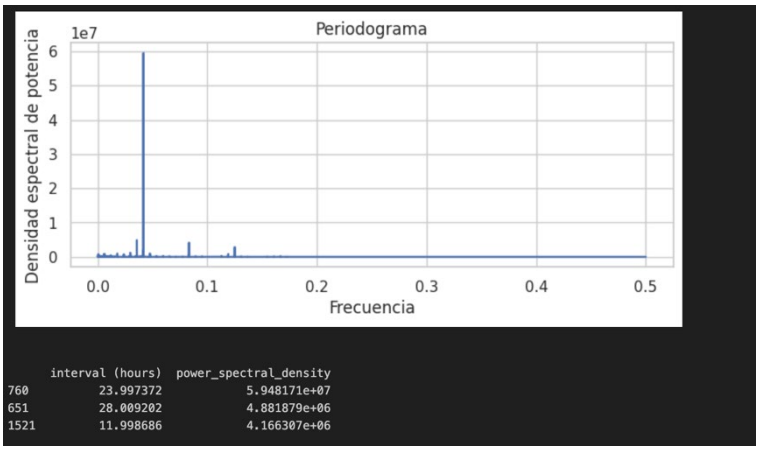


Figura 4: periodograma para parking con ID = 6

Cada barra del gráfico muestra qué frecuencia es más importante en la serie temporal. Si la barra es alta, significa que esa frecuencia destaca y puede indicar que hay patrones que se repiten, cada día, cada medio día, etc.

## 4. División del conjunto de datos e ingeniería de características.

Estos datos que hemos limpiado y analizado van a ser utilizados para entrenar un modelo, esto es, para crear un algoritmo que permita detectar los patrones existentes en la ocupación de parkings y así poder predecir las plazas libres que habrá en cada momento. Para ello vamos a utilizar las llamadas Redes Neuronales Recurrentes (RNN), un tipo un tipo de red neuronal que sirve para trabajar con datos que tienen un orden o una secuencia. Las RNN utilizan la información previa para predecir la siguiente.

Como variable para entrenar los modelos vamos a usar el dato de las plazas disponibles por aparcamiento. Utilizaremos el porcentaje de plazas disponibles, pero en vez de usar el porcentaje entre 0 y 100, se usa entre 0 y 1, para mejorar la convergencia y acelerar el entrenamiento de las redes neuronales recurrentes.

La ventana temporal elegida es de 24 horas, en base a los ciclos detectados en el análisis EDA, es decir, se usan las 24 horas anteriores a la hora actual como entrada al modelo, para predecir el valor actual.

Con respecto a la división del conjunto de datos, se aplica el método hold-out con una particularidad: el rango temporal del conjunto de test es común para todos los parkings, para poder comparar el rendimiento en el mismo período. Es decir, se usa el 70% de los datos para entrenar, un 15% para validar y un 15% para test. Siendo el rango temporal del subconjunto de

test común para todos los parkings y los subconjuntos de entrenamiento y validación se ajusten individualmente a la cantidad de datos de cada parking.

En resumen, tras todos los análisis anteriores, vamos a usar el porcentaje de plazas disponibles por aparcamiento entre 0 y 1, de los parkings con id: 6, 7, 8, 13, 34, 75, 77, 78.

## 5. Algoritmos.

Para entrenar nuestros modelos se han usado las siguientes arquitecturas de redes neuronales (ver Figura 5):

Redes Neuronales Recurrentes (RNN):

- Es la arquitectura más básica: aplica una capa oculta que se retroalimenta a sí misma en cada paso temporal.
- Es capaz de manejar secuencias cortas, pero con limitaciones en retención de memoria a largo plazo.

Long Short-Term Memory (LSTM)

- Es un tipo de RNN adecuada para recordar información durante más tiempo.
- Introduce 3 puertas de control (olvido, entrada y salida) y una celda de memoria interna para controlar el flujo de información.
- La puerta de olvido decide qué parte del estado anterior descartar, mientras que la de entrada regula qué información nueva incorporar.
- Capaz de retener información a largo plazo, pero más costosa computacionalmente.

Gated Recurrent Unit (GRU)

- Alternativa más ligera a LSTM, con menos parámetros y entrenamiento más rápido
- Introduce una puerta de actualización y otra de olvido, que se encarga de controlar

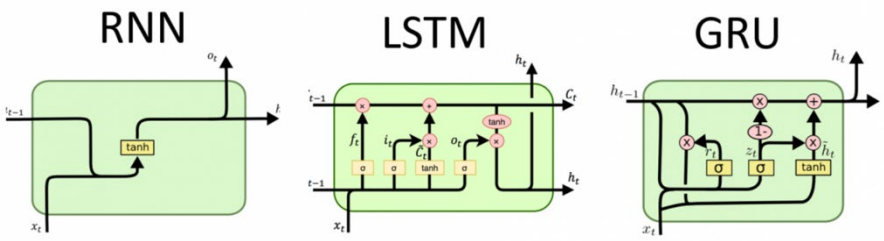


Figura 5: arquitectura de las redes neuronales utilizadas

El objetivo es comparar la versión vanilla de cada una de estas arquitecturas, es decir, con 1 capa y 32 neuronas por capa, frente a una versión con los hiperparámetros óptimos.

En la siguiente tabla se observan los hiperparámetros probados en cada enfoque. Para la versión optimizada se propone cambiar: tasa de aprendizaje, nº de capas y nº de neuronas por capa

| PARÁMETRO | VANILLA | OPTUNA |
|---|---|---|
| *OPTIMIZADOR* | ADAM | ADAM |
| *BATCH SIZE* | 32 | 32 |
| *TASA APRENDIZAJE* | 0.001 | [0.005-0.02] |
| *TASA DECAIMIENTO* | 0.05 | 0.05 |
| *DROPOUT* | 0.2 | 0.2 |
| *N.º DE CAPAS* | 1 | [2,3,4] |
| *N.º NEURONAS POR CAPA* | 32 | [16,32,64] |

Tabla 1: parámetros planteados en cada arquitectura

## 6. Resultados.

Como métrica de evaluación se ha usado el MAE (error medio absoluto), por ser la más interpretable.En la figura 6 se muestra el MAE para cada enfoque planteado: versión vanilla (básica) y versión con hiperparámetros óptimos.

Para cada uno de estos enfoques, se muestra el error cometido para cada parking por cada uno de los modelos (RNN, LSTM, GRU).

Es importante destacar que el MAE se calcula sobre el porcentaje de plazas ocupadas, no sobre el total de plazas del parking, es decir, si el error cometido por el modelo GRU en la versión vanilla para el parking 13 es 2.4, el MAE sobre el total de plazas del parking sería de 5.49 plazas (de un total de 229 plazas). Los errores mostrados en la tabla son porcentajes.

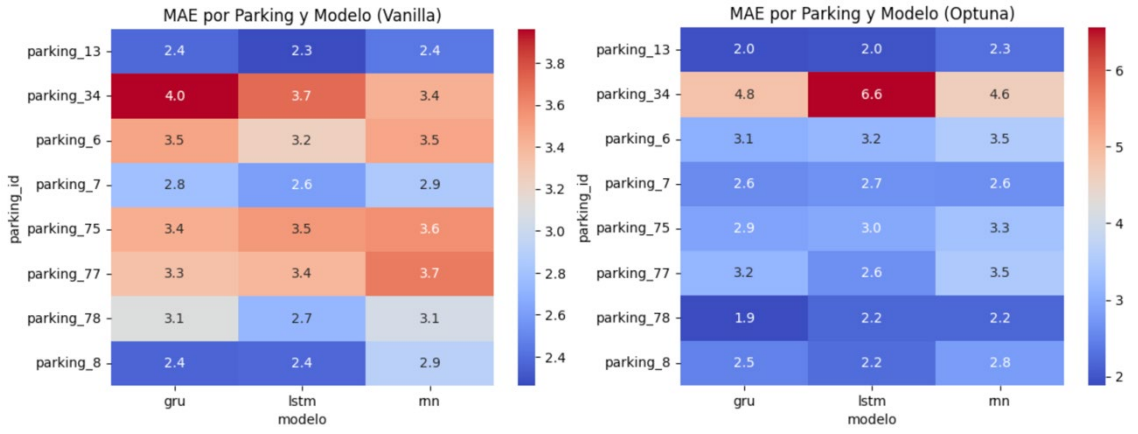


Figura 6: Error medio absoluto por enfoque, modelo y parking

Como puede apreciarse, el ajuste óptimo de hiperparámetros reduce el MAE en todos los modelos y parkings analizados. La diferencia no es muy grande, ya que no es un problema con patrones difíciles de detectar, y una red vanilla, es suficiente para captar las dependencias. Para la puesta en producción, habría que considerar si conviene reducir el error a costa de aumentar el coste computacional.

En la siguiente gráfica se observa como todos los modelos planteados capturan los comportamientos cíclicos con mayor o menor detalle para el parking con id 6 sobre la primera semana del conjunto de test (24 horas * 7 dias = 168 horas)


Figura 7: predicciones sobre 1º semana del conjunto de test para el parking con id 6

## 7. Explicabilidad de los algoritmos de IA

A la hora de aplicar modelos de Inteligencia Artificial (IA) y Machine Learning (ML) es muy importante aprender a interpretar los resultados, para ello existen métodos de interpretabilidad y explicabilidad.

En este caso se plantea utilizar *SHapley Additive exPlanations* (SHAP), un método basado de la teoría de juegos, que estima cuanto aporta cada variable a la predicción en un instante temporal determinado.

De forma intuitiva, SHAP genera todas las combinaciones de variables posibles para ver cuánto contribuye cada una de ellas a la predicción final. Es decir, en nuestro caso particular, genera todas las combinaciones de 1, 2,…, 24 instantes anteriores para ver cuáles son los que más contribuyen a la predicción final de un instante temporal determinado.

En la siguiente imagen se muestra la importancia de cada instante temporal anterior para el parking 6 en su versión vanilla, para el modelo LSTM. Podemos ver cómo los 10/12 instantes anteriores son los que más influyen en la predicción.

Esto contrasta con lo que se había observado en el análisis EDA, que sugería utilizar los 24 instantes anteriores para realizar las predicciones. La figura 8 sugiere reducir la ventana temporal a la mitad para ver si se mantiene la calidad de las predicciones reduciendo el coste computacional a la mitad.

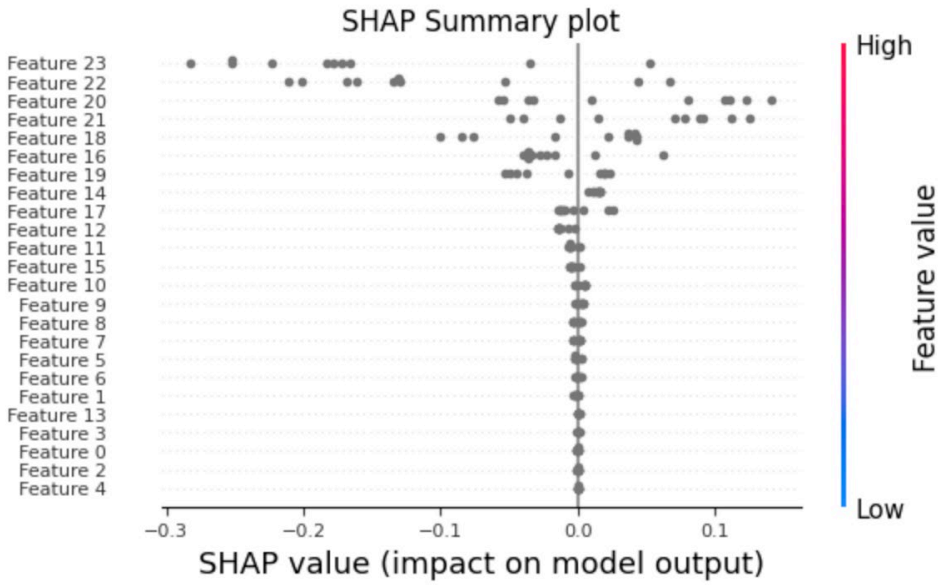


Figura 8: valores SHAP para el parking 6, modelo LSTM, versión “vanilla”

## 8. Conclusiones

El trabajo realizado ha permitido diseñar e implementar un sistema de predicción de disponibilidad de plazas de aparcamiento en los parkings públicos de la ciudad de Valencia, basado en técnicas avanzadas de aprendizaje automático.

A lo largo del proyecto se ha llevado a cabo un exhaustivo proceso de adquisición, limpieza y depuración de datos, identificando y resolviendo múltiples limitaciones que podrían haber comprometido la calidad de las predicciones. Este trabajo preliminar ha resultado fundamental para garantizar la robustez del sistema desarrollado.

Se ha evaluado el rendimiento de diferentes arquitecturas de redes neuronales recurrentes (RNN, LSTM, GRU), tanto en su versión básica como en configuraciones optimizadas mediante ajuste de hiperparámetros. Los resultados obtenidos muestran que todas las arquitecturas son capaces de capturar los patrones cíclicos de ocupación de los parkings con gran precisión. Si bien el ajuste de hiperparámetros mejora ligeramente el rendimiento, una configuración sencilla ya proporciona resultados satisfactorios, lo que permite equilibrar precisión y coste computacional.

Además, la incorporación de técnicas de explicabilidad, como SHAP, ha permitido entender mejor el comportamiento de los modelos y la influencia de las distintas variables en las predicciones, aportando un valor añadido a la solución final.

En definitiva, el sistema desarrollado constituye una base sólida para futuras aplicaciones de movilidad inteligente en la ciudad de Valencia. Como líneas de trabajo futuro se plantea la evolución hacia un sistema MLOps automatizado que permita integrar nuevos datos en tiempo real, optimizar continuamente los modelos y facilitar su despliegue en entornos productivos.

## 9. Trabajo futuro

Con respecto al trabajo futuro, se puede desarrollar un sistema de MLOps, que orqueste el despliegue, la monitorización continua y la gestión de versiones del modelo seleccionado. Se planea construir pipelines automáticos de entrenamiento y validación, reduciendo la intervención manual y acelerando el ciclo de mejoras.

También se pueden integrar datos adicionales (meteorología, eventos sociales…) para ver si contribuyen a la calidad de los resultados.

## 10. Aplicación de modelos de predicción a otros datos de la ciudad.

Las técnicas y metodologías desarrolladas en este trabajo son fácilmente extensibles a otros ámbitos relacionados con la movilidad y los servicios urbanos de la ciudad.

Por ejemplo, podrían aplicarse al análisis y predicción de flujos de tráfico en la red viaria, al uso de bicicletas públicas (Valenbisi), a la ocupación de plazas de carga y descarga, o incluso a la gestión dinámica de las rutas de recogida de residuos. La combinación de estos modelos predictivos con datos abiertos y sistemas de visualización en tiempo real podría contribuir significativamente a la optimización de los servicios municipales y a la mejora de la sostenibilidad urbana.