# Modelos de regresión para estimar el uso de Bicing en una estación

Fernando Guirao
fernando.guirao@estudiantat.upc.edu

Nicolas Llorens
nicolas.llorens@estudiantat.upc.edu


Aprendizaje Automático - Q1 23/24

Fecha

# Índice

# 1. Introducción

En este informe se expone el trabajo realizado para la práctica de Aprendizaje Automático del Q1 23/24. Se describe el problema escogido y la recopilación de los datos para modelar el mismo. Además, se muestra la aplicación de diferentes modelos de aprendizaje automático para realizar predicciones sobre una variable objetivo. Finalmente, se realizan comparativas y se extraen conclusiones.

El objetivo es hacer estimaciones sobre el uso de Bicing, el servicio de bicicletas públicas de Barcelona, y así lograr detectar picos de demanda en determinados momentos del día.

## 1.1. ¿Qué es Bicing?

**Bicing** es un servicio de bicicletas públicas inaugurado a inicios de 2007, gestionado por la empresa municipal Barcelona de Serveis Municipals (BSM), que cuenta con más de cien mil abonados. Dispone de bicicletas mecánicas y eléctricas que pueden ser recogidas y ancladas en las diferentes estaciones que hay repartidas por la ciudad.

## 1.2. Motivación

Cada día miles de personas utilizan Bicing para desplazarse al trabajo, a la universidad, etc. Muchos estudiantes del Campus Nord usan este servicio para ir y volver de la universidad. No obstante, la elevada demanda hace que a veces las estaciones aledañas queden vacías obligando a algunos a escoger transportes alternativos.

El servicio cuenta con remolcadores que reponen bicicletas en las estaciones que se quedan vacías a ciertas horas del día. Pero en determinadas ocasiones, esto puede no ser suficiente.

Nuestra idea inicial es realizar predicciones sobre la demanda por cada estación en cada determinada hora del día de cada mes. Esta información puede resultar valiosa para que, de forma preventiva, se lleven a cabo las reposiciones en el mejor momento posible ante horas donde se espere una alta demanda.

En este proyecto trabajaremos sobre los datos de la estación que se encuentra en la **Calle Jordi Girona, 29** con identificador **422**, ya que es la estación que se encuentra justo debajo del campus. Sin embargo, la idea es que todo el trabajo realizado pueda aplicarse a cualquier estación.

Una consideración importante es que trataremos con datos relativos al uso de bicicletas **eléctricas**. Esto se debe a que el uso de bicicletas mecánicas es significativamente menor al de las eléctricas. Las bicicletas eléctricas requieren mucho menos esfuerzo y moverse con ellas es más rápido. En definitiva, cada tipo de bicicleta presenta patrones de uso lo suficientemente distintos como para ser tratados a parte. Por ejemplo, el uso de bicicletas mecánicas decae con mucha más facilidad ante episodios de calor. Por lo tanto, en adelante, nos referiremos

a las bicicletas eléctricas simplemente como bicicletas.

# 2. Obtención de los datos y creación del dataset

Cuando pensamos en demanda (o uso) y en cómo cuantificarla, pensamos en diferentes planteamientos que añadían una complejidad innecesaria. Finalmente decidimos considerar algo tan intuitivo como es el número de bicicletas que salen de la estación. Y concretamente, buscamos obtener el **número de salidas por hora**, ya que es lo que buscaremos predecir.

## 2.1. Primeras aproximaciones

Nuestra forma de encarar esta primera fase del proyecto puede no haber sido la más ortodoxa, ya que en lugar de buscar directamente datasets sobre los que aplicar modelos y realizar predicciones, hemos optado por pensar en un problema muy concreto, lo que nos creó la necesidad de tener un dataset muy concreto que quizás había que construir.

Y, en efecto, tras una serie de búsquedas, no logramos encontrar un conjunto de datos que se ajustase del todo al problema planteado. No obstante, sí que encontramos datos que podían servir como base para construir el dataset. El primer dataset con el que trabajamos se encuentra en el catálogo de datasets de **Open Data BCN**. Este dataset recoge información sobre, entre otras cosas, el número de bicicletas disponible de cada una de las estaciones de Bicing cada pocos minutos. Con esto, lo que podíamos hacer era filtrar los registros por la estación número 422, agruparlos por horas e ir viendo las salidas que se iban produciendo. Es decir, si estamos calculando las salidas que se producen el 1 de enero de 2021 a las 11:00, tendríamos que mirar el número de bicicletas del primer registro, luego calcular la diferencia con el siguiente registro y acumularla en una variable, y así sucesivamente.

El problema que había con esta forma de construir nuestro dataset, era que claramente podíamos estar dejando de capturar muchas salidas. Pongamos un ejemplo: nuevamente, queremos calcular el número de salidas que se producen el 1 de enero de 2021 a las 11:00. Estamos mirando los registros (por ejemplo) de las 11:03, y el de las 11:07, y en ambos registros el número de bicicletas es de 5. Con la actual forma de calcular el número de salidas diríamos que en ese intervalo no se ha producido ninguna. Sin embargo, puede haber ocurrido que hayan llegado 3 bicicletas pero también hayan salido 3, que nos estaríamos perdiendo.

Otro problema, es que el dataset de Open Data BCN recopila el número de bicicletas a cada momento, y las variaciones que pueden producirse de un momento a otro pueden venir dadas por reposiciones o retiradas de un remolcador, y no por salidas o entradas de usuarios.

Por lo tanto, llegados a este punto, si queríamos ser fieles a la realidad con el número de salidas que se producían por hora, necesitábamos obtener el *log* de salidas que se producían en la estación.

## 2.2. Datos definitivos

Decidimos entonces ponernos en contacto con Barcelona de Serveis Municipals (BSM), quienes muy amablemente nos proporcionaron los datos que necesitábamos para seguir adelante.

En este punto habíamos logrado construir un dataset que indicaba, por cada hora, de cada día, de cada mes del año, el número de bicicletas que habían salido de la estación 422. Cabe mencionar que la construcción del dataset se realiza mediante una serie de scripts en Python a partir de los datos originales. Estos scripts están disponibles en el repositorio de este proyecto. Sin embargo, los datos originales no se encuentran en el repositorio debido al gran tamaño que tienen.

Nuestro conjunto de datos contiene un registro por hora **entre el 1 de enero de 2022 y el 30 de noviembre de 2023** (ambos incluidos), es decir, un total de **16775 registros**. No utilizamos datos anteriores debido a que en la fase de pruebas de este proyecto nos dimos cuenta de que en los datos de 2021, en parte debido a las restricciones sanitarias que todavía existían, clases online, etc., el número de salidas en media era mucho menor, por lo tanto eran datos que añadían ruido y confundían a los modelos. Quizás añadir variables relacionadas con las medias móviles podría haber ayudado a que esos datos contribuyan en vez de añadir ruido, pero es algo que no pudimos probar.

## 2.3. Variables obtenidas

Tenemos el número de salidas por cada hora en el periodo de tiempo mencionado anteriormente, pero naturalmente necesitamos variables que nos ayuden a predecir la variable objetivo.

No es difícil pensar en variables que puedan afectar al uso de Bicing. En primer lugar pensamos en las variables meteorológicas. Es evidente que la lluvia o las fuertes temperaturas (y más variables meteorológicas) desincentivan el uso de las bicicletas. Por otro lado tenemos las variables temporales: según la hora del día el número de bicicletas que se cogen cambia mucho, o según el día de la semana, o el mes, o si se trata de un día festivo o no lectivo en la universidad.

### 2.3.1. Variable objetivo y tipo de problema

Queda claro que la variable objetivo es el número de salidas que se producen en la estación. En esta sección la definimos formalmente con el nombre de `exits`. Al ser una variable numérica deberíamos pensar que estamos ante un problema de regresión, pero esta variable es discreta, y en concreto es una variable de conteo. Este hace que este problema pueda considerarse de regresión de conteo. Pero es que además, nuestro dataset es una serie temporal, con determinadas periodicidades, no son observaciones aleatorias. Todo esto se tendrá en cuenta a la hora de entrenar los modelos y validarlos.

### 2.3.2. Variables temporales

Con variables temporales nos referimos a aquellas que nos indicanla hora, día, mes y año del registro. También si ese registro pertenece a un día festivo, laborable, o no lectivo (que no tiene por qué ser festivo). Por lo tanto, en cuanto a este tipo de variables, contamos con:

- `hour`
- `day`
- `month`
- `year`
- `week_day`
- `workiing_day`
- `class_day`

Aclaramos que la variable `working_day` es binaria, es 1 si el día al que pertenece el registro es festivo municipal, autonómico o nacional, 0 en caso contrario. La variable `class_day` tiene 3 valores: 1 si es día de clase, 2 si es día de exámenes, y 3 si no es día ni de clase ni de exámenes (que puede ser o no festivo).

### 2.3.3. Variables meteorológicas

Para obtener las variables meteorológicas cada hora utilizamos la API de Open-Meteo. Las variables que utilizamos son:

- `temperature`
- `humidity`
- `a_temperature`
- `precipitation`
- `rain`
- `wind_speed`

## 2.4. Variables Generadas

### 2.4.1. Variables *lagged*

Dado que nuestro conjunto de datos es una serie temporal, decidimos añadir este tipo de variables, que recogen información del registro anterior. De este modo facilitamos que se capturen tendencias y patrones temporales.

Las variables de este tipo añadidas son `lag_1`, que indica las salidas que se produjeron en la hora anterior, y `demand_satisfied_lag`, que indica si en la hora anterior la estación quedó con 0 bicicletas en algún momento. Esta última variable se genera a partir del dataset de Open Data BCN.

### 2.4.2. Variable `initial_bikes`

Por último, gracias también a los datos de Open Data BCN, añadimos una variable que indica el número de bicicletas que hay disponibles al comienzo de cada hora.

# 3. Trabajos previos

Durante los últimos años se han realizado y publicado diversos proyectos y artículos que usan datos de Bicing, además de muchos otros proyectos y artículos enfocados a servicios públicos de bicicletas de otras ciudades. Vamos a centrarnos en comentar aquellos estudios que han utilizado datos de Bicing centrándose en diferentes aspectos como la disponibilidad de bicicletas, el uso del sistema y factores que influyen en este uso. Aquí van algunos de ellos:

- «**Bicing Stats**» (Trabajo de fin de grado): Mencionamos primero este TFG realizado en la UPC por Alejandro Carol Campreciós. En este proyecto, lo que se hace es un estudio sobre la viabilidad de una aplicación capaz de predecir la disponibilidad de bicicletas. Logra obtener buenos resultados utilizando el algoritmo de Random Forest.
- «**Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system**»: Este artículo analiza datos de movilidad humana en un área urbana utilizando la cantidad de bicicletas disponibles en las estaciones del programa de bicicletas comunitarias Bicing en Barcelona. El enfoque está en entender y predecir tendencias en el uso de bicicletas basándose en datos de disponibilidad
- «**Riding the wave: Predicting the use of the bike-sharing system in Barcelona before and during COVID-19**»: Esta investigación desglosa los factores clave para predecir el uso de Bicing, descubriendo evidencia de un aumento en la infraestructura construida relacionada con bicicletas (por ejemplo, urbanismo táctico),

distancia de viaje y niveles de ingreso de los barrios como los predictores más relevantes. Concretamente se busca predecir el número de viajes por día que se realizan por barrio.

- «**Predicting Occupancy Trends in Barcelona's Bicycle Service Stations Using Open Data**». En este paper realizado por investigadores de la UPF muestra el estudio de algoritmos como Random Forest para predecir la disponibilidad de bicicletas en momentos muy concretos a dos días vista.

- «**Bike sharing usage prediction with deep learning: a survey**» Este estudio no utiliza datos de Bicing sino de otros servicios de bicicletas públicas pero hemos querido mencionarlo ya que también busca predecir uso. Categoriza diferentes tipos de problemas de predicción de uso de sistemas de bicicletas compartidas y resume los modelos de predicción de aprendizaje profundo más recientes utilizados en estudios revisados, así como aplicaciones basadas en predicciones.

Lo que vemos es que la mayoría de estudios se enfocan en realizar predicciones sobre la disponibilidad, y no sobre el uso como es nuestro caso. Sí que es cierto que el estudio de «Riding the wave» realiza predicciones sobre el uso pero aplicándolo a barrios enteros y por días. El último estudio que mencionamos habla de distintas formas de enfocar el problema de predicción de uso y soluciones mediante algoritmos de deep learning.

En definitiva, pensamos que nuestro proyecto tiene cabida ya que nos enfocamos en realizar predicciones sobre el uso por horas en una estación determinada. Sí que es interesante ver que nuestro problema podría escalarse mucho más utilizando datos de más estaciones y explorando soluciones que involucren algoritmos de deep learning. Sin embargo todo ello se sale del *scope* de este proyecto, pero podría servir como base de un proyecto de mayor talla como puede ser un trabajo de fin de grado.

# 4. Exploración preliminar del dataset

En esta sección, llevamos a cabo una exploración del dataset. Este estudio preliminar es fundamental para comprender la naturaleza y la estructura de los datos, identificar patrones y tendencias significativas, y determinar las variables más relevantes que influirán en la construcción y el rendimiento de nuestros modelos de aprendizaje automático.

Repasemos el aspecto final del dataset y sus variables:

| | hour | day | month | year | week_day | working_day | class_day | exits | temperature | a_temperature | humidity | precipitation | rain | wind_speed | initial_bikes | lag_1 | demand_satisfied_lag |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16347 | 3 | 13 | 11 | 2022 | 6 | 0 | 3 | 0 | 15.7 | 16.9 | 96.5 | 0.0 | 0.0 | 3.6 | 10.0 | 0.0 | 1.0 |
| 22879 | 7 | 12 | 8 | 2023 | 5 | 0 | 3 | 0 | 25.4 | 29.0 | 78.8 | 0.0 | 0.0 | 7.3 | 7.0 | 0.0 | 1.0 |
| 9727 | 7 | 10 | 2 | 2022 | 3 | 1 | 3 | 1 | 2.2 | -0.6 | 94.8 | 0.0 | 0.0 | 6.3 | 7.0 | 0.0 | 0.0 |
| 13225 | 1 | 6 | 7 | 2022 | 2 | 1 | 3 | 0 | 23.6 | 25.3 | 74.7 | 0.0 | 0.0 | 11.7 | 1.0 | 0.0 | 1.0 |
| 13594 | 10 | 21 | 7 | 2022 | 3 | 1 | 3 | 0 | 31.0 | 33.6 | 49.3 | 0.0 | 0.0 | 13.0 | 14.0 | 5.0 | 1.0 |

Figura 1: Aspecto final de nuestro dataset

| Variable | Descripción |
|---|---|
| `hour` | Hora del día (0-23) |
| `day` | Día del mes |
| `month` | Mes del año (1-12) |
| `year` | Año |
| `week_day` | Día de la semana (0-6) |
| `working_day` | Si es día laboral (0=no, 1=sí) |
| `class_day` | Si es día lectivo (1) de exámenes (2) o ninguna de las 2 (3) |
| `exits` | **Número de salidas (la variable objetivo)** |
| `temperature` | Temperatura en grados Celsius |
| `a_temperature` | Sensación térmica en grados Celsius |
| `humidity` | Humedad en porcentaje |
| `precipitation` | Precipitación en mm |
| `rain` | Si está lloviendo (0=no, 1=sí) |
| `wind_speed` | Velocidad del viento en km/h |
| `initial_bikes` | Bicis disponibles al comienzo de la hora |
| `lag_1` | Número de salidas que se produjeron en la hora anterior |
| `demand_satisfied_lag` | Si la estación siempre tuvo bicis disponibles en la hora anterior |

Cuadro 1: Descripción de las variables del conjunto de datos

## 4.1. Distribución de las variables

Comenzaremos visualizando la distribución de los datos. En primer lugar, usaremos histogramas para representar la distribución de las variables numéricas:

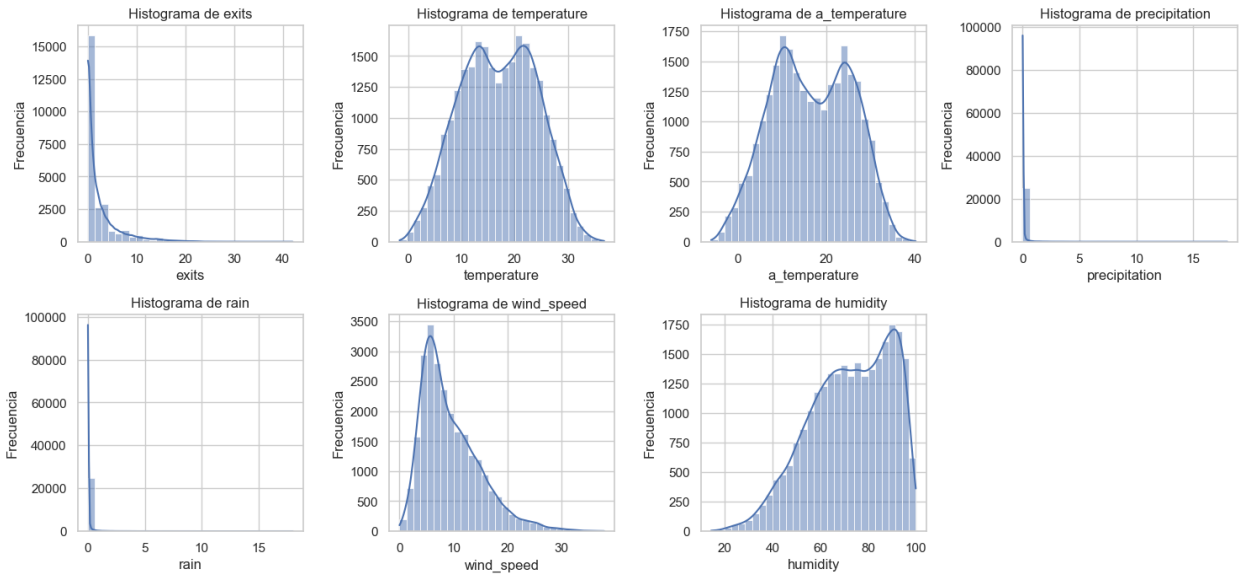


Figura 2: Histogramas de las variables numéricas

- Los histogramas de temperatura son evidentemente similares, aunque no del todo, por lo que por el momento mantendremos ambas variables.
- Algo parecido sucede con los histogramas de precipitación y lluvia, en este caso la similitud es mucho mayor ya que precipitaciones que no sean en forma de lluvia (nieve o granizo) se dan de forma excepcional. Cabe destacar que la forma que tienen es normal dado que la inmensa mayoría de registros marcan precipitación nula o muy cercana a 0. Los datos de precipitación que estamos acostumbrados a ver se dan en acumulaciones de más horas, o días enteros. Además también hay que sumar el factor de sequía que viene dándose los últimos años.
- Los histogramas de viento y humedad presentan valores esperados para una ciudad costera como es Barcelona.
- Por último, vemos que el número de salidas en la mayoría de ocasiones es nulo o casi nulo. Esto tiene sentido porque la actividad en la parada se concentra en momentos de hora punta. No significa que haya poca actividad en la parada, sino que esta se concentra en momentos concretos.

A continuación procedemos a visualizar la distribución de las variables categóricas (las más relevantes) mediante gráficos de barras:

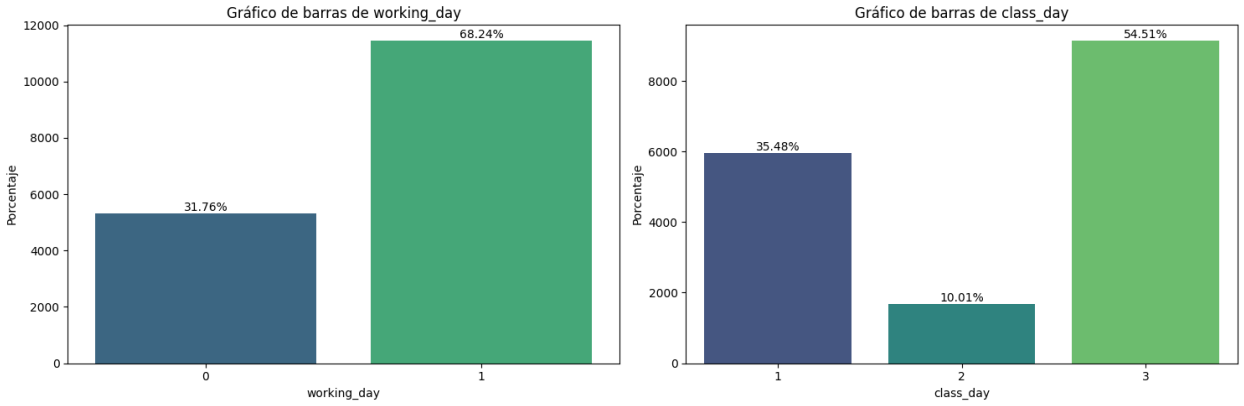


Figura 3: Gráficos de barras de las variables categóricas

Es interesante ver en este caso también los porcentajes.

- La mayoría de registros son de días laborables, pero los estudiantes parecen tener más tiempo en el que no tienen clases ni exámenes.

## 4.2. Relación de la variable objetivo con el resto de variables

En esta sección, primero utilizaremos gráficos de dispersión para ilustrar las relaciones de la variable objetivo con las demás variables numéricas:

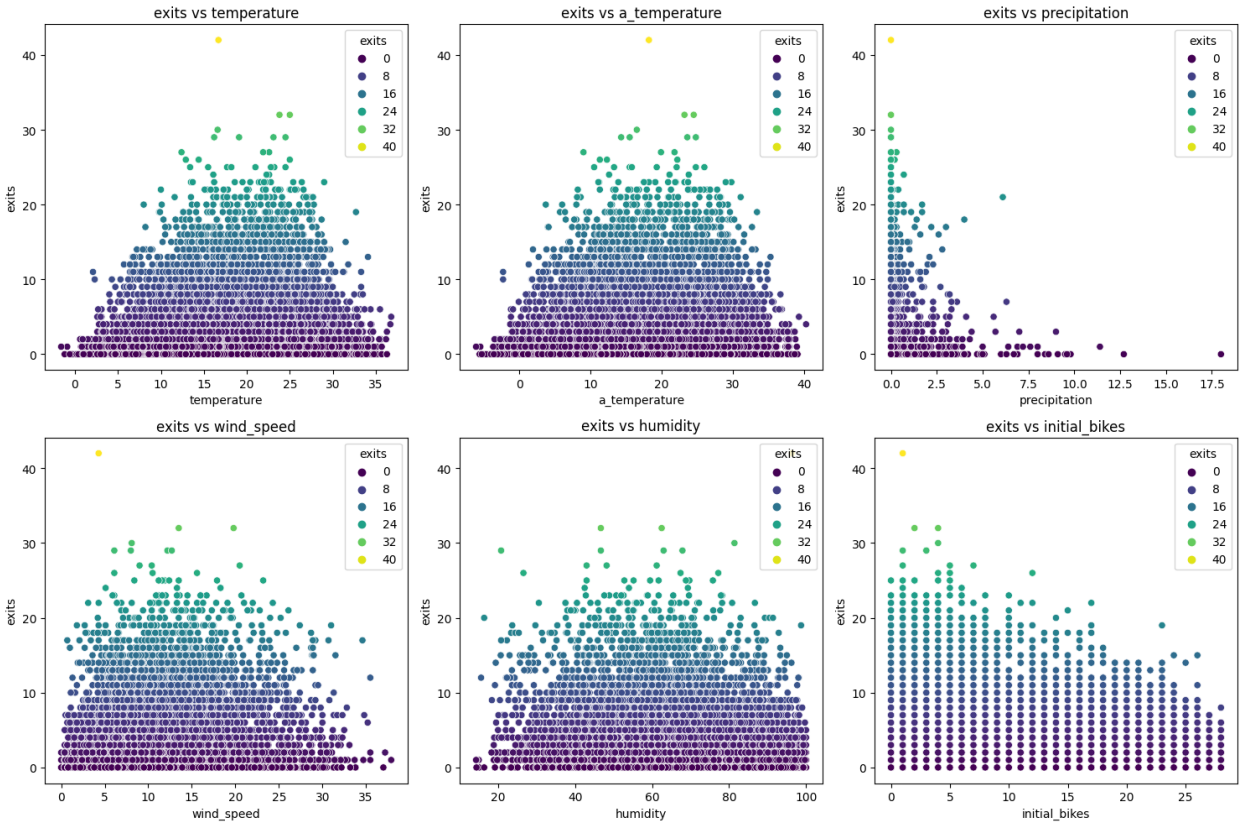


Figura 4: Gráficos de dispersión entre las variables numéricas y la variable objetivo

- En cuanto a temperaturas, vemos que la mayoría de salidas se realizan cuando estas son moderadas. En caso de temperaturas muy bajas o muy altas, el uso disminuye claramente.
- Como es de esperar, las condiciones de lluvia también desincentivan el uso de la bicicleta.
- En cuanto al viento, puede llegarse a apreciar que las fuertes rachas pueden ser un factor disuasorio.
- Los momentos de alta humedad, cuando es más fácil sudar, parecen indicar una ligera disminución en el uso de bicicletas.
- El número de bicicletas inicial muestra, como es de esperar, una relación positiva con el número de salidas.

Estas tendencias reflejan comportamientos razonables, ya que las personas tienden a evitar realizar actividades al aire libre o salidas en condiciones climáticas adversas. Pero es evidente que estos patrones **no son lineales** ni monotónicos y ya nos indican por lo tanto que los modelos lineales pueden no funcionar del todo bien. Pueden requerir modelos más complejos que puedan capturar la relación entre las salidas y las condiciones climáticas, como árboles de decisión, ensambles de modelos o redes neuronales, que no se basan en la suposición de relaciones lineales.

Nuevamente proseguimos con las variables categóricas:

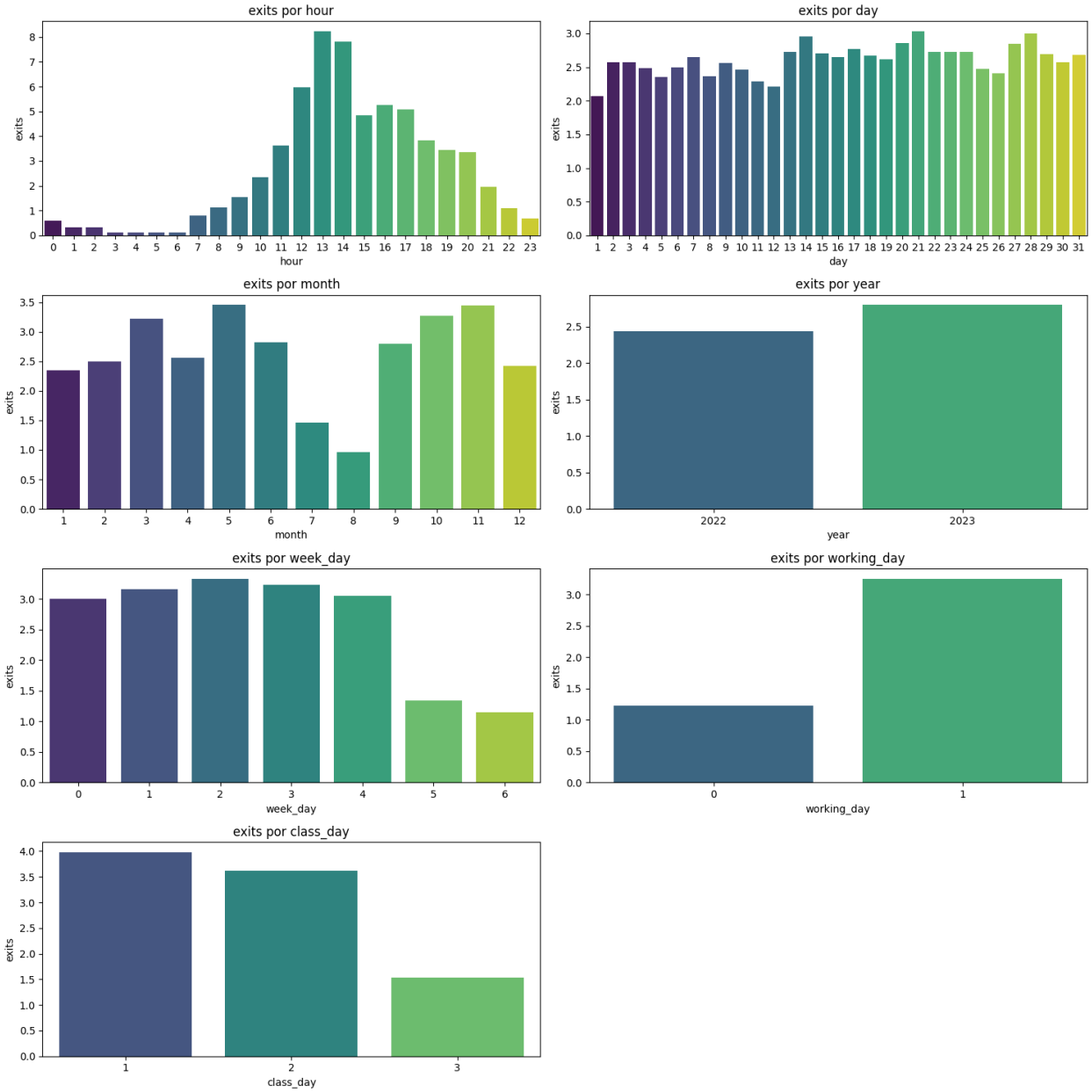


Figura 5: Gráficos comparativos entre las variables categóricas y la variable objetivo

Vemos claros patrones sobre el número de salidas.

- Claramente se distinguen las horas del día con mayor actividad de salidas.
- Los meses de verano registran números más bajos.
- Durante el fin de semana, también se ve reducido el número de bicicletas que se cogen.
- También podemos ver cómo en los días laborables se cogen más bicicletas en promedio, así como en los días de clase y exámenes.

### 4.3. Tratamiento de valores faltantes

El dataset de Open Data BCN tiene períodos de tiempo en los que no se reporta nada debido a fallos técnicos. Como consecuencia, nuestro dataset tiene valores faltantes en las variables construidas a partir del dataset de Open Data BCN `demand_satisfied_lag` y `initial_bikes`. Debido a que estamos tratando con una serie temporal de datos y tenemos variables que utilizan información de registros adyacentes, no podemos borrar los registros con valores perdidos. Es por ello que debemos imputar esos valores.

Para no hacerlo de una forma *dummy* usaremos el imputador `KNNImputer` de `sklearn`, que reemplaza los valores faltantes utilizando los valores de los vecinos más cercanos. En este proceso omitimos la variable objetivo para evitar sesgos en los valores imputados y predecimos usando sólo los datos de entrenamiento (que definimos en la siguiente sección).

En total imputamos 653 valores perdidos en cada una de las variables mencionadas.

## 5. Protocolo de remuestreo

En esta sección definimos cuál será el esquema general a la hora de entrenar los modelos y medir su rendimiento.

### 5.1. División de los datos entre entrenamiento y test

Debido a que nuestro dataset es una serie temporal de datos, es imprescindible que los datos de entrenamiento contengan registros anteriores a los de los datos de test. Los conjuntos de entrenamiento y test no pueden seleccionarse aleatoriamente.

Es por ello que **utilizaremos el último mes de nuestro dataset (noviembre de 2023) como conjunto de pruebas**. Es decir, nuestros modelos se medirán prediciendo el número de salidas para cada hora de noviembre de 2023.

Una consideración importante a hacer es que, idealmente, para este problema, en una aplicación real este modelo debería reentrenarse en tiempo real con el objetivo de predecir

la próxima hora con la información más actualizada. Para simular esto, lo que podríamos hacer sería entrenar y validar el modelo para cada hora de noviembre de 2023 (o cualquier intervalo de tiempo), pero desafortunadamente no disponemos de recursos computacionales para ello.

Nuestros modelos quizás no realizarán las predicciones con la información más reciente a cada hora, pero sí que contarán con las variables *lagged* y el número inicial de bicicletas, que pese a no ser todo el histórico desde 2022, es información actualizada con la que contaríamos en la aplicación real.

## 5.2. Exploración de hiperparámetros

Para determinar unos hiperparámetros adecuados en aquellos modelos que hay que definirlos, hacemos validación cruzada utilizando `TimeSeriesSplit`, que es esencial cuando tratamos con series temporales. Con este método se divide el conjunto de entrenamiento en n partes (3 en nuestro caso), asegurándose de que la validación se realice siempre en datos que cronológicamente siguen a los datos de entrenamiento.

A la hora de ir probando los hiperparámetros, en lugar de probar todas las combinaciones posibles como en una búsqueda en cuadrícula estándar (`GridSearchCV`), utilizamos `BayesSearchCV`. La búsqueda Bayesiana selecciona las combinaciones de manera más inteligente, basándose en los resultados anteriores para mejorar las estimaciones de los mejores hiperparámetros. Esto reduce el coste computacional de la validación cruzada, lo que nos permitirá agilizar el proceso de entrenamiento de los modelos.

```
# Configurar TimeSeriesSplit para la validación cruzada
tscv = TimeSeriesSplit(n_splits=3)

# Configurar BayesSearchCV
bs = BayesSearchCV(
    rf,
    param_grid,
    n_iter=30,
    cv=tscv,
    scoring=make_scorer(mean_squared_error, greater_is_better=False),
    n_jobs=-1,
    refit=True,
    random_state=0
)
```

### 5.2.1. Métricas de rendimiento

Para medir el rendimiento de los modelos nos fijaremos principalmente en estas métricas:

- **Coeficiente de determinación** ($R^2$): mide qué tan bien explica el modelo la variabilidad de los datos reales.
- **RMSE (Root Mean Squared Error)**: mide la desviación promedio de las predicciones del modelo de los valores reales.

Complementaremos y comprenderemos estas métricas con gráficos que ilustren las predicciones contra los valores reales.

# 6. Aplicación de modelos lineales y/o cuadráticos

Empezaremos con 3 modelos clásicos: Regresión lineal, *K-Nearest-Neighbours* ySVM con Kernel polinómico.

## 6.1. Regresión lineal

La regresión lineal es el punto de partida más común para problemas de regresión debido a su simplicidad y facilidad de interpretación. Es útil para entender la influencia directa de cada predictor individual en la variable objetivo.

```
# Creamos el modelo
linear_reg = LinearRegression()

# Lo ajustamos con los datos de entrenamiento
linear_reg.fit(X_train_lr, y_train_lr)

# Realizamos las predicciones
y_pred_lr = linear_reg.predict(X_test_lr)

# Evaluamos en el conjunto de test
r2_score_lr = linear_reg.score(X_test_lr, y_test_lr)
mse = mean_squared_error(y_test_lr, y_pred_lr)
```

Obtenemos estas métricas:

| Model | R2 Score | RMSE |
|---|---|---|
| Linear Regression | 0.539578 | 3.301920 |

Como vemos, obtenemos un R2 de 53.96 % esto se interpreta como variabilidad en la variable dependiente que es explicada por el modelo. El RMSE es algo alto: 3.30 lo que quiere decir que de media predice ±3.30 salidas respecto al valor real de salidas. Para verlo mejor veremos dos gráficas.

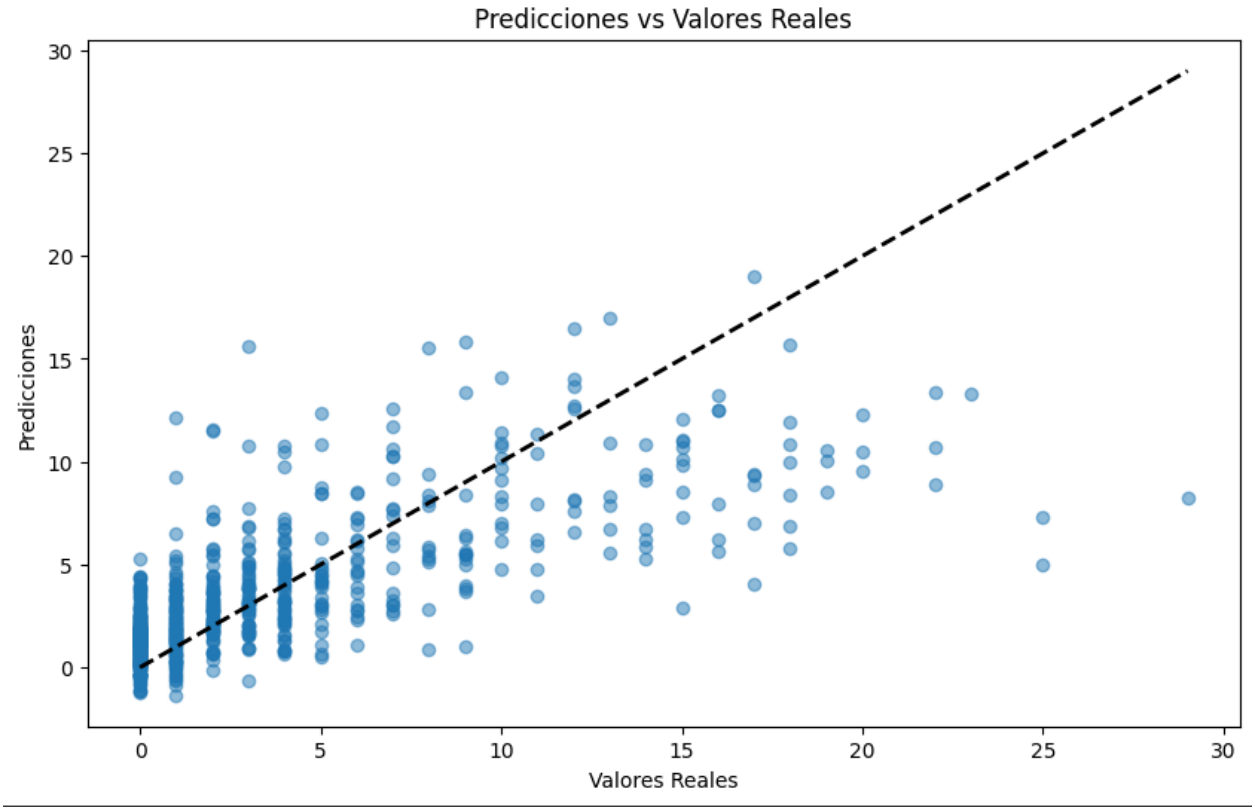


Figura 6: Gráfica de los valores reales vs los valores predichos por el modelo

Esta gráfica nuestra que el modelo mide claramente peor a medida que los valores reales a predecir son mas grande, lo que sugiere que hay factores adicionales o relaciones no capturadas por el modelo que podrían mejorar el ajuste.

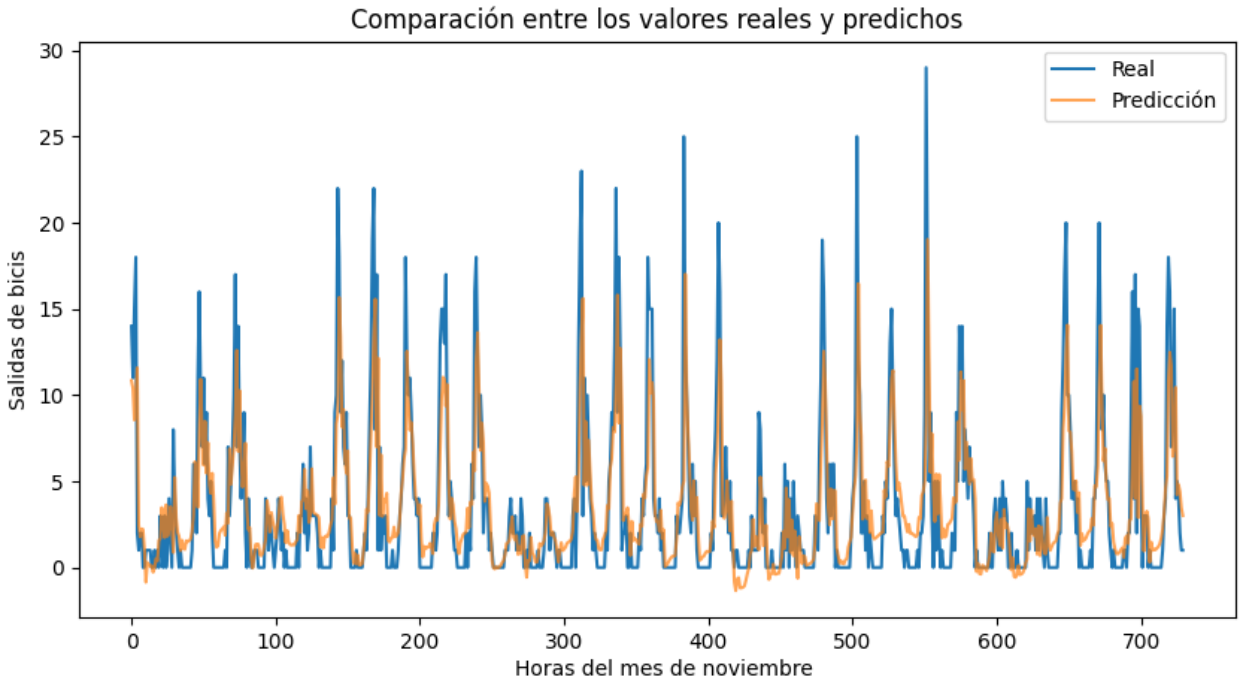


Figura 7: Gráfica de la predicción del mes de noviembre según el modelo

Aunque hay una captura general de la tendencia y la estacionalidad, las predicciones no coinciden con la magnitud de los picos y valles reales. Las predicciones parecen subestimar o sobreestimar en diferentes puntos, lo cual es típico de los modelos que no pueden capturar completamente la variabilidad en los datos.

## 6.2. k-Nearest Neighbors (KNN)

KNN puede capturar relaciones no lineales al basarse en la similitud de las instancias más cercanas. Es especialmente útil en situaciones donde los patrones de datos son intrínsecamente locales, como podría ser el caso en ciertos momentos del día o ciertas condiciones meteorológicas.

Antes de ajustar el modelo, debemos aplicar la normalización de las variables numéricas y one-hot encoding para las categóricas, como requiere este modelo. Estos son los hiperparámetros que exploraremos:

- **n_neighbors** (`3, 20`)**:** Afecta directamente la suavidad o la rugosidad de las predicciones.
- **weights** [**'uniform', 'distance'**]**:** Determina cómo se ponderan los votos de los vecinos en la predicción.

- **leaf_size [20, 30, 40]:** Afecta la velocidad de construcción y consulta del árbol.
- **p [1, 2]:** Define la métrica de distancia utilizada (por ejemplo, la distancia Euclidiana o la distancia de Manhattan).

Después de la búsqueda bayesiana obtenemos que estos son los mejores hiperparámetros para nuestro conjunto de datos dentro de nuestro param grid:
`('leaf_size', 40), ('n_neighbors', 12), ('p', 1), ('weights', 'distance')`

Obtenemos estas métricas:

| Model | R2 Score | RMSE |
|---|---|---|
| K-Nearest Neighbors | 0.752024 | 2.423222 |

Un $R^2$ del 75 %, notablemente mejor que la regresión lineal. El RMSE es mucho mas bajo.

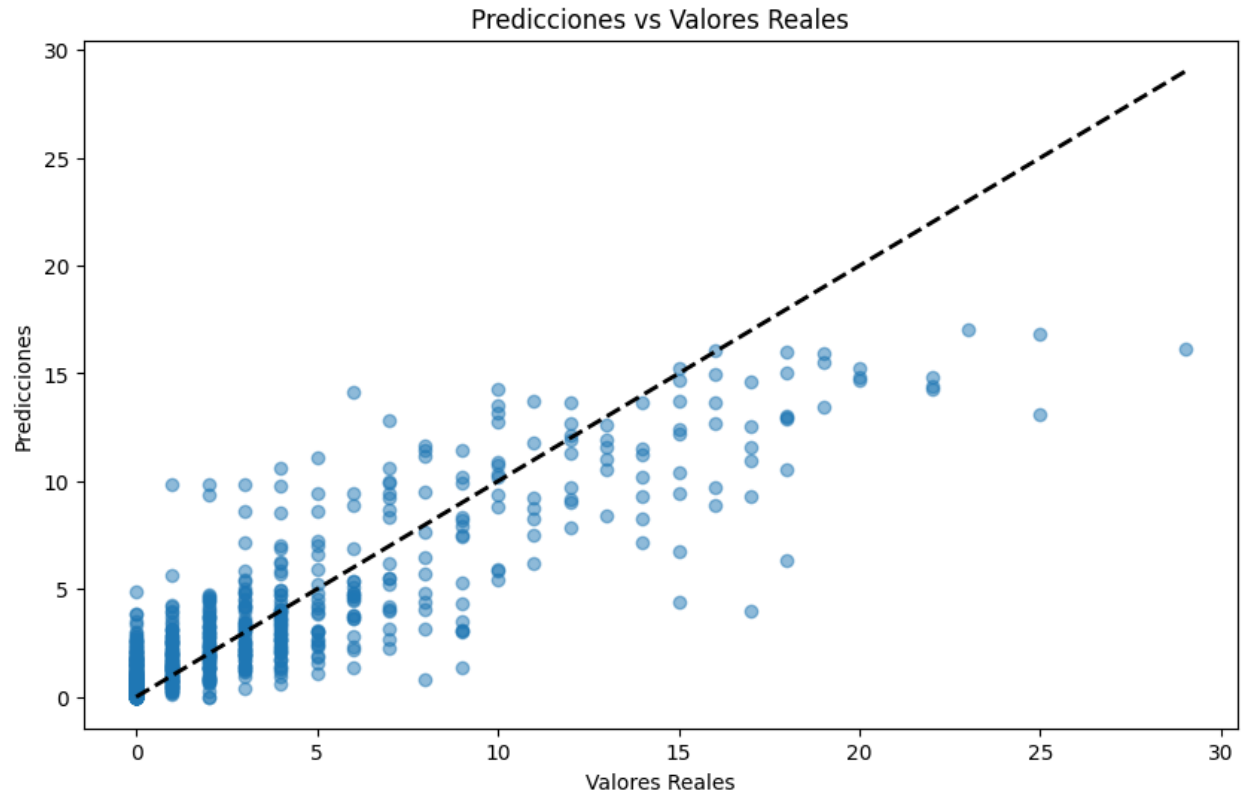


Figura 8: Gráfica de los valores reales vs los valores predichos por el modelo

Con esta gráfica podemos ver la diferencia respecto a la regresión sobretodo a partir de los valores reales altos, aun que también a mayor número de salidas a predecir mas se equivoca (prediciendo también por lo bajo).

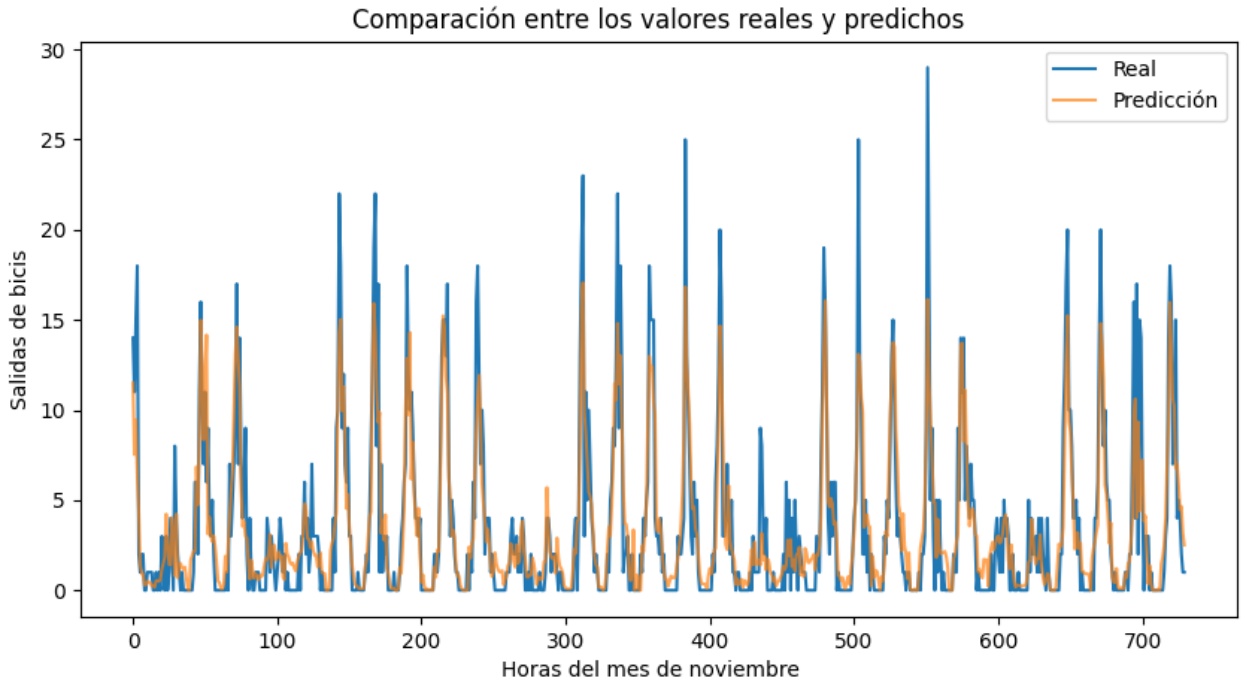


Figura 9: Gráfica de la predicción del mes de noviembre según el modelo

Mirando la imagen global del mes entero, vemos un poco de mejora en el acercamiento de las predicciones a esos picos de demanda.

## 6.3. Support Vector Machines (SVM) con kernel polinómico

Ajustamos un modelo SVM Poly. También, aplicaremos la normalización de las variables numéricas y one-hot encoding para las categóricas, como requiere este modelo. Exploraremos los siguientes hiperparámetros:

- **degree (2, 4):** Controla la complejidad del modelo. Valores más altos permiten considerar interacciones más complejas.
- **C [0.1, 0.5, 1.0, 5.0]:** controla la penalización por errores en la función de pérdida
- **gamma ['scale', 'auto', 1.0]:** Gamma afecta la influencia de un solo punto de entrenamiento. Un valor bajo significa una influencia alta y un valor alto significa una influencia baja.
- **max_iter [-1, 100, 200]:** Este parámetro establece el número máximo de iteraciones para la optimización del problema de SVM.

Los hiperparámetros seleccionados son:
`('degree', 2), ('C', 5.0),('gamma', 'auto'), ('max_iter', -1)])`

Obtenemos estas métricas:

| **Model** | **R2 Score** | **RMSE** |
|---|---|---|
| SVM with Polynomial Kernel | 0.627484 | 2.970030 |

Obtenemos un R2 score de aproximadamente el 62 % . El MSE es mas bajo que lr pero peor que KNN: 2,97.

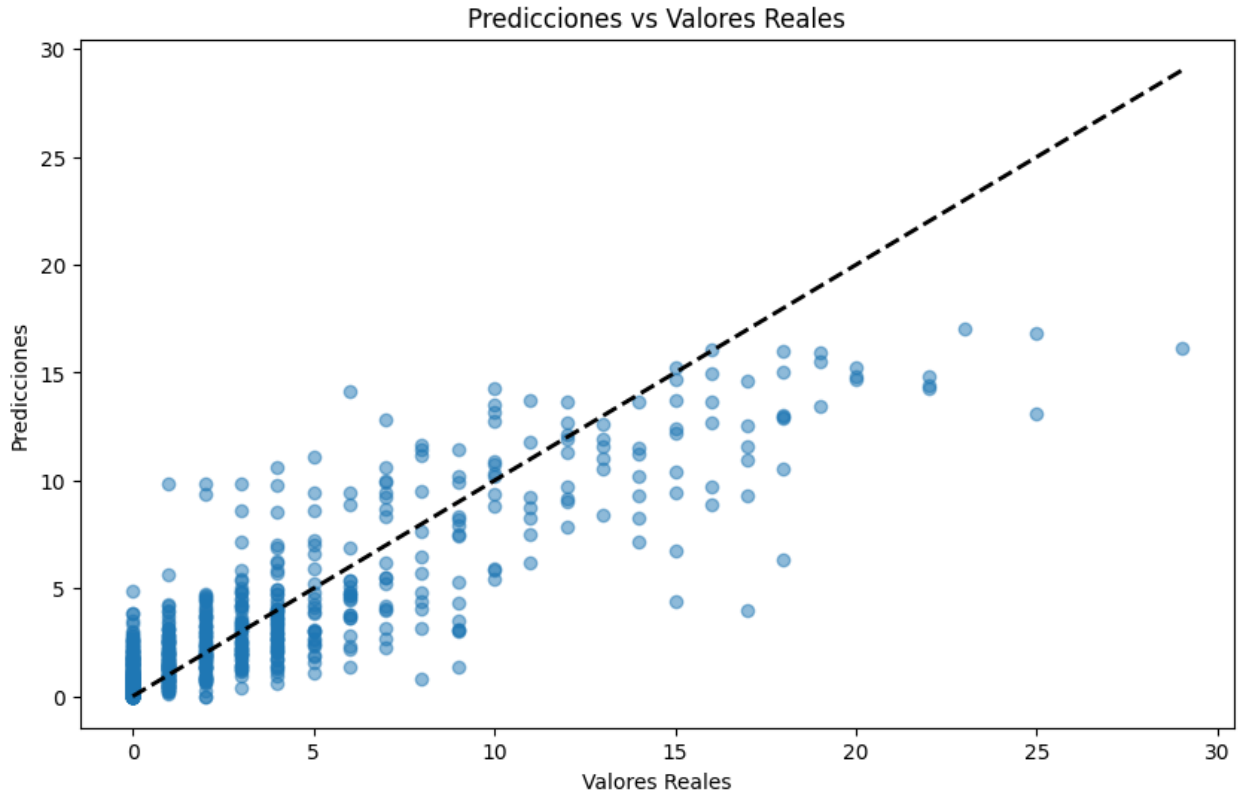


Figura 10: Gráfica de los valores reales vs los valores predichos por el modelo

Con esta gráfica podemos ver la diferencia respecto a la regresión sobretodo a partir de los valores reales altos, aunque también a mayor número de salidas a predecir, más se equivoca (subestimando).

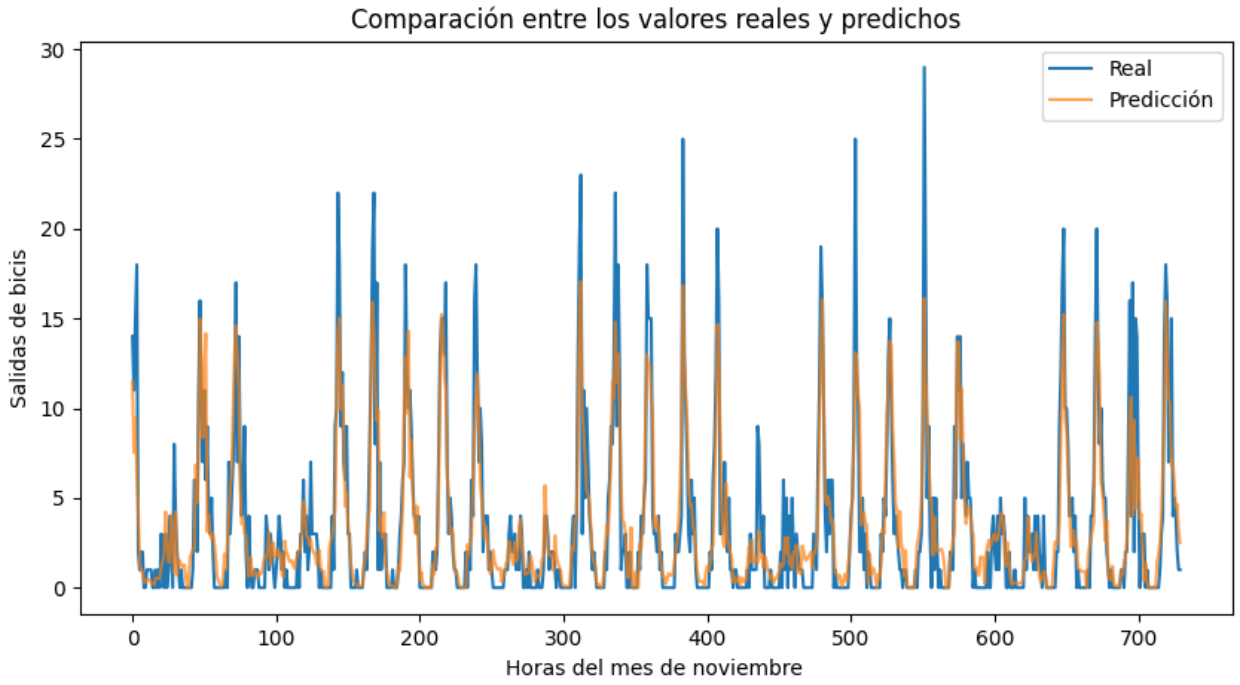


Figura 11: Gráfica de la predicción del mes de noviembre según el modelo

Mirando la imagen global del mes entero, vemos un poco de mejora en el acercamiento de las predicciones a esos picos de demanda.

# 7. Aplicación de modelos no lineales

los modelos no lineales son importantes para explorar y capturar patrones más complejos que no son aparentes a través de métodos lineales. Probaremos con **Random Forest**, **Gradient Boosting** y el **Perceptrón Multicapa**.

- **Random Forest:** Lo hemos escogido porque maneja bien conjunto de datos grandes. Al incorporar múltiples árboles de decisión, este modelo puede capturar patrones no lineales.
- **Gradient Boosting:** Es un modelo que destaca por su habilidad para mejorar gradualmente la precisión de este mediante la construcción secuencial de árboles de decisión. Esta característica puede ser especialmente valiosa ya que puede .ªprender"de los errores de predicciones anteriores.
- **MultiLayer Perceptron (MLP)**: Tiene capacidad para modelar relaciones no lineales complejas entre las variables (mas que RF y GB). Este modelo destacara si realmente hay patrones subyacentes en los datos mas complejos de lo que el resto de modelos pueden capturar.

## 7.1. Random Forest

Random Forest construye un conjunto de árboles de decisión y promedia sus predicciones para obtener un modelo robusto. Exploraremos los siguientes hiperparámetros:

- **n_estimators [50, 100, 150]:** Número de árboles en el bosque.
- **max_depth [5, 10, 15]:** La máxima profundidad de cada árbol en el bosque.
- **min_samples_split [2, 6, 10]:** Número mínimo de muestras requeridas para dividir un nodo interno.
- **min_samples_leaf [10, 20, 30]:** Número mínimo de muestras requeridas para estar en un nodo hoja.

Después de la búsqueda bayesiana, hemos obtenido que para nuestra serie de datos, los mejores hiperparámetros son:
`('max\_depth', None), ('min\_samples\_leaf', 4), ('min\_samples\_split', 10), ('n\_estim`

Obtenemos estas métricas:

| Model | R2 Score | RMSE |
|---|---|---|
| Random Forest | 0.7828 | 2.2674 |

Obtenemos un R2 score del 78 %.El RMSE 2,26.

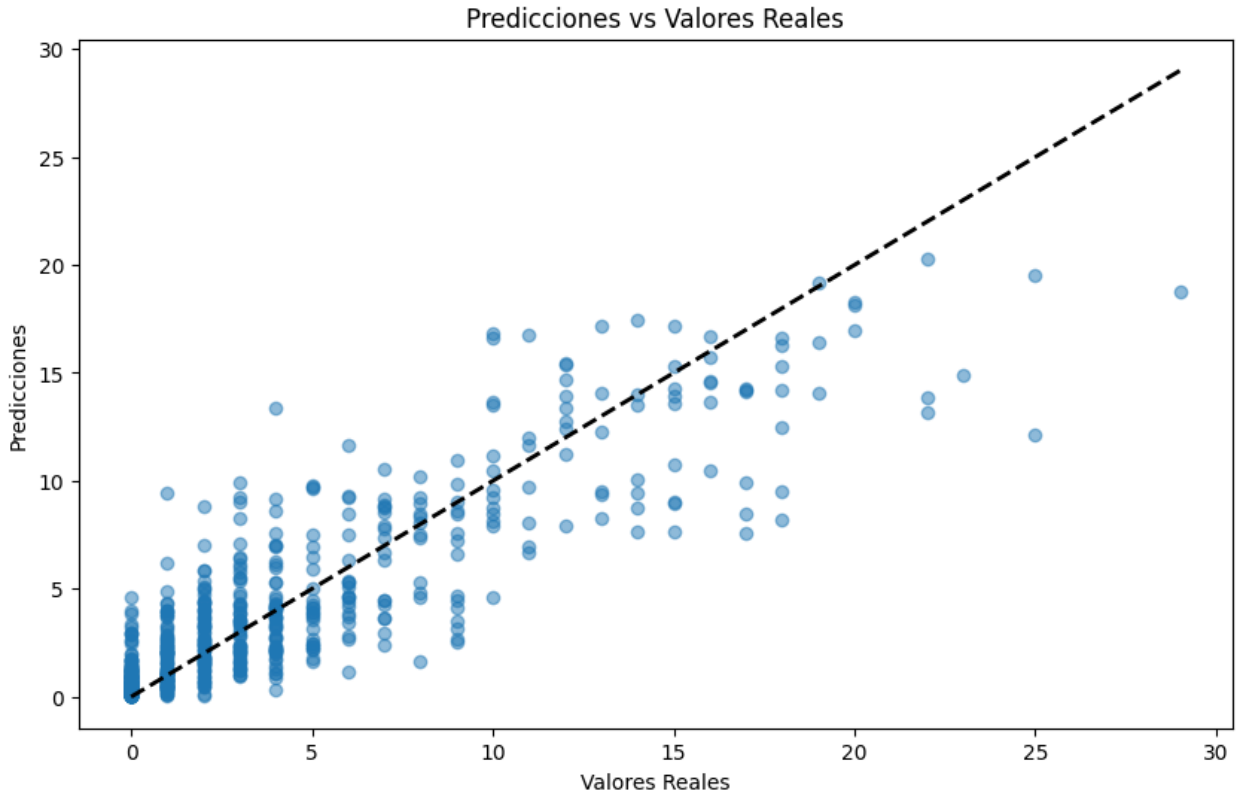


Figura 12: Gráfica de los valores reales vs los valores predichos por el modelo

Con metricas parecidas al KNN vemos también un comportamiento similar para los valores altos, ya a partir de las aproximadamente 13 salidas se pierde esa media de la dispersión en la linea.

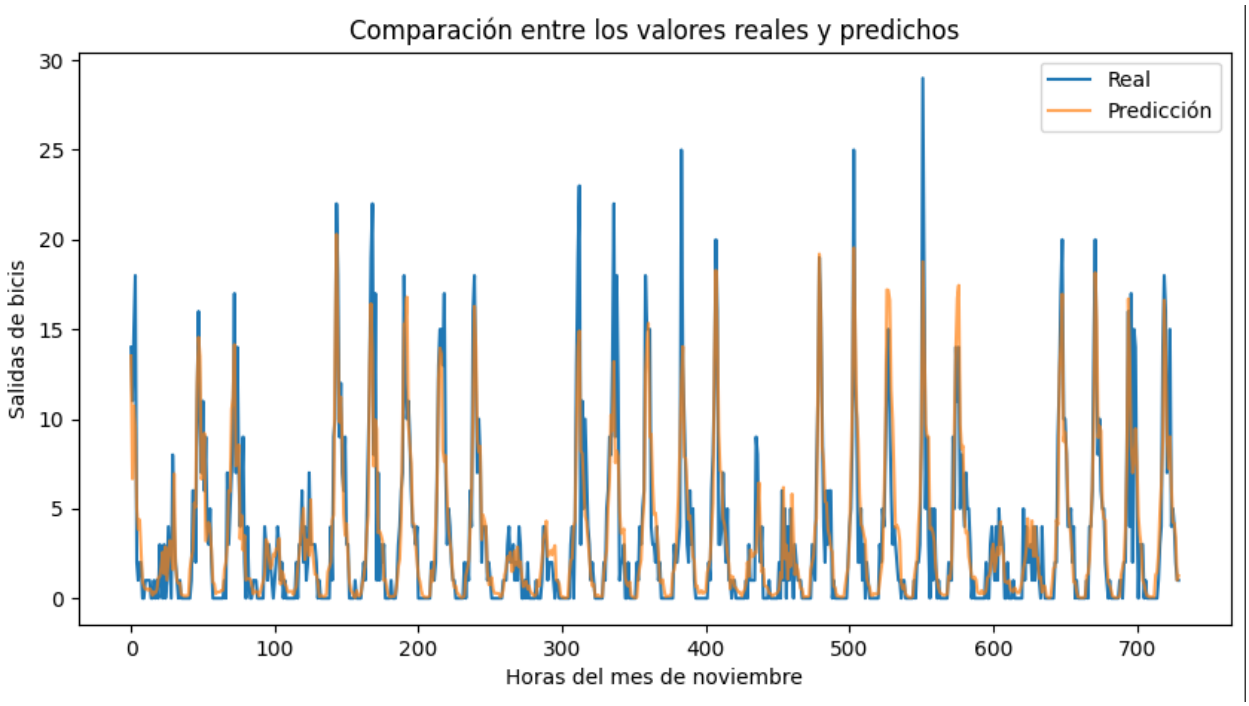


Figura 13: Gráfica de la predicción del mes de noviembre según el modelo

En el mes entero vemos lo que hablabamos en el gráfico anterior, este modelo aun teniendo buenas metricas se queda corto en los picos de máxima demanda.

## 7.2. Gradient Boosting

Gradient Boosting construye un modelo aditivo de manera secuencial, optimizando una función de pérdida. Exploraremos los siguientes hiperparámetros:

- **n_estimators [50, 100, 150]:** Es el número de árboles en el conjunto.
- **max_depth [5, 10, 15]:** Profundidad máxima de cada árbol. Un valor más alto permite que los árboles capturen patrones más complejos, pero también aumenta el riesgo de sobreajuste.
- **min_samples_split [2, 6, 10]:** Número mínimo de muestras requeridas para dividir un nodo interno.
- **min_samples_leaf [10, 20, 30]:** Número mínimo de muestras requeridas para estar en un nodo hoja.
- **learning_rate [0.05, 0.1, 0.2]:** Tasa de aprendizaje que controla la contribución de cada árbol al modelo.

Después de la busqueda bayesiana hemos obtenido que para nuestra serie de datos los mejores hiperparametros son:
`('learning\_rate', 0.1), ('max\_depth', 5), ('min\_samples\_leaf', 30), ('min\_samples\_`

Obtenemos estas métricas:

| Model | R2 Score | RMSE |
|---|---|---|
| Gradient Boosting | 0.7872 | 2.2444 |

Obtenemos un R2 score del 79 %.El RMSE de 2,24. Metricas muy parecidas al RF.

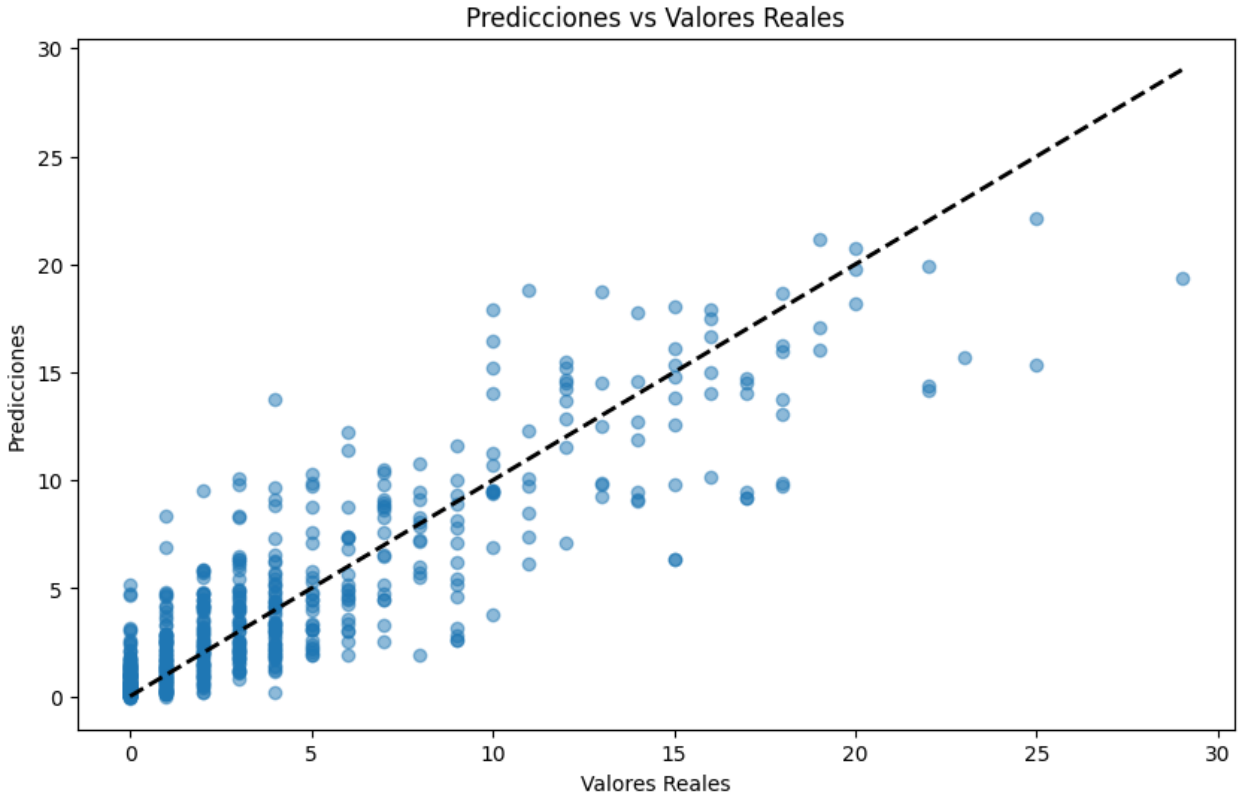


Figura 14: Gráfica de los valores reales vs los valores predichos por el modelo

Con esta gráfica podemos ver que el modelo entiende mucho mejor los valores mas altos, y se ajusta a ellos. Empieza a hacerlo muy al principio por lo que vemos que hay mas predicciones que debian ser alrededor de 0 que se alejan más.

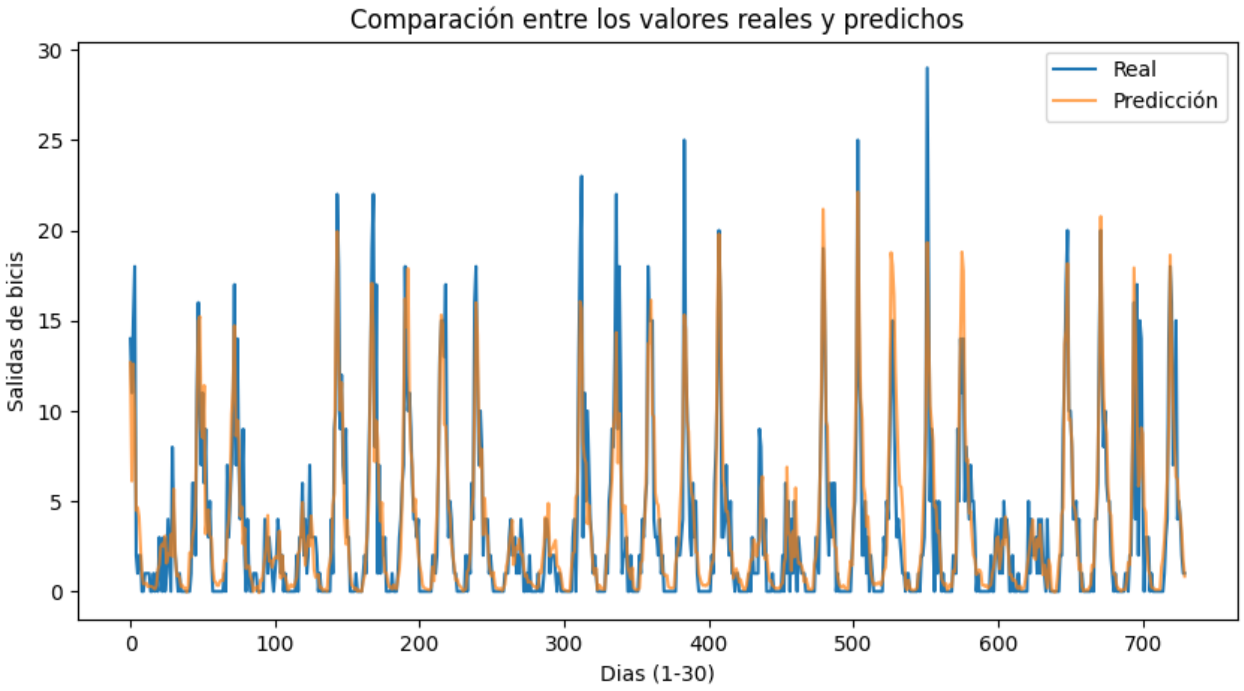


Figura 15: Gráfica de la predicción del mes de noviembre según el modelo

Mirando la imagen global del mes entero, observamos lo que decíamos: el modelo entiende mucho mejor los valores mas altos, y se ajusta a ellos. Cabe destacar que aun que en la gráfica anterior, por la dispersión en los valores bajos parece que el modelo vaya a sobre estimarlos, pero no lo hace (en mayor medida).

## 7.3. MultiLayer Perceptron (MLP)

MLP es un tipo de red neuronal que puede capturar relaciones complejas y no lineales a través de múltiples capas y neuronas. Exploramos los siguientes hiperparámetros:

- **hidden_layer_sizes [20, 50, 100]:** Número de neuronas en las capas ocultas.
- **activation ['relu', 'tanh']:** Función de activación.
- **solver ['lbfgs', 'adam']:** Solucionador utilizado para la optimización de pesos.
- **alpha [0.00005, 0.0001, 0.0002]:** Parámetro de regularización para controlar el sobreajuste.
- **learning_rate ['constant', 'adaptive']:** Tasa de aprendizaje. 'Constant' mantiene la misma tasa durante todo el entrenamiento, 'adaptive' ajusta la tasa de forma adaptativa.

- **max_iter [100, 200, 300]:** Número máximo de iteraciones para la convergencia del modelo.
- **shuffle [False]:** False mantiene la serialización de nuestras muestras en cada iteración.

Después de la búsqueda bayesiana hemos obtenido que para nuestra serie de datos los mejores hiperparametros son:
`('activation', 'tanh'), ('alpha', 0.0001), ('hidden\_layer\_sizes', 20), ('learning\_rat`

Obtenemos estas métricas:

| Model | R2 Score | RMSE |
|---|---|---|
| MultiLayer Perceptron (MLP) | 0.7936 | 2.1901 |

Obtenemos un R2 score del 79 %. El RMSE de 2,19. Son metricas casi identicas al RF.

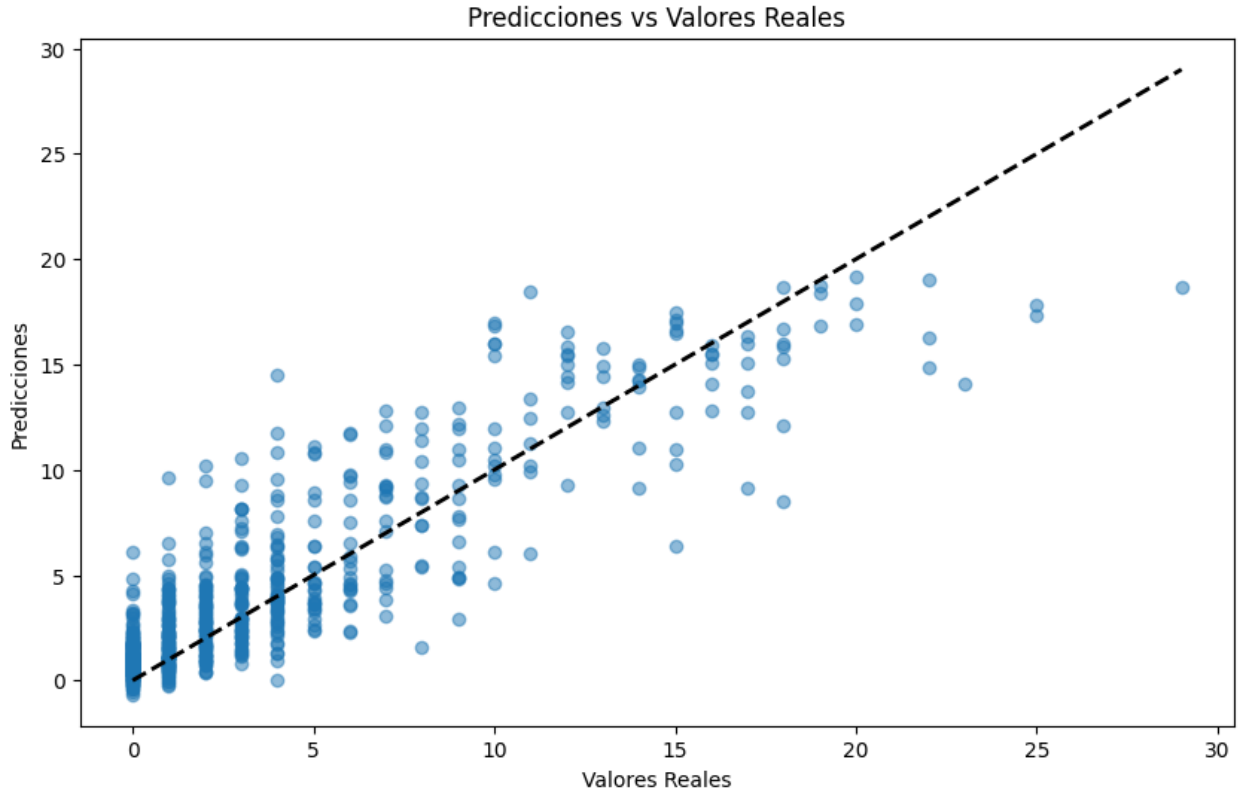


Figura 16: Gráfica de los valores reales vs los valores predichos por el modelo

Vemos que la dispersión de las predicciones es buena, significativamente mejor que para el RF, siguiendo la linea de 45 grados, la dispersión se mantiene (mas o menos) de media en la linea, lo que nos indica lo bien que esta prediciendo este modelo. Aun así no se llega a solucionar del todo los momentos de gran cantidad de salidas, aun que hay que ver como afecta finalmente a la predicción.

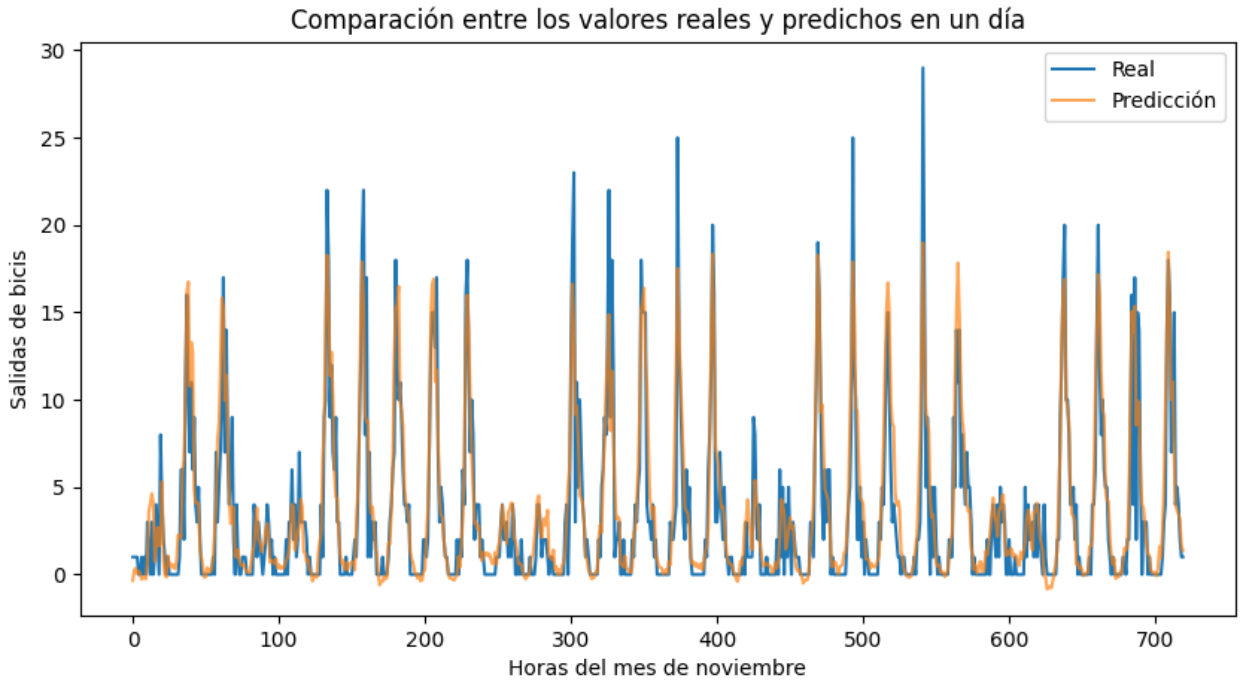


Figura 17: Gráfica de la predicción vs datos reales del mes de noviembre según el modelo

Para el mes entero vemos una precisión realmente buena, es conservador en los picos muy altos (mas de 20 salidas) pero aun así sigue teniendo una precisión general para los picos altos buena. Todo esto sin sacrificar casi la sensibilidad a las horas con 0 salidas.

# 8. Comparación de los modelos

La tabla de metricas final para predicciones del mes de noviembre de 2023 entero tiene finalmente esta forma:

| **Model** | **R2 Score** | **RMSE** |
|---|---|---|
| Linear Regression | 0.5395 | 3.3019 |
| K-Nearest Neighbors | 0.7520 | 2.4232 |
| SVM with Polynomial Kernel | 0.6274 | 2.9700 |
| Random Forest | 0.7828 | 2.2674 |
| Gradient Boosting | 0.7872 | 2.2444 |
| MultiLayer Perceptron (MLP) | 0.7936 | 2.1901 |

Como era de esperar, los modelos no lineales rinden notablemente mejor. No obstante, hay que destacar el papel de KNN, un algoritmo que además de ser competitivo, es simple, intuitivo y rápido.

| Model | R2 Score | RMSE |
|---|---|---|
| K-Nearest Neighbors | 0.6954 | 2.6080 |
| Random Forest | 0.7933 | 2.1481 |
| Gradient Boosting | 0.7856 | 2.1877 |
| MultiLayer Perceptron (MLP) | 0.7919 | 2.1555 |

## 8.1. Selección del mejor modelo

Descartando los modelos de regresión lineal y SVM polinómico, vamos a reentrenar los modelos usando como conjunto de test la última semana del dataset.

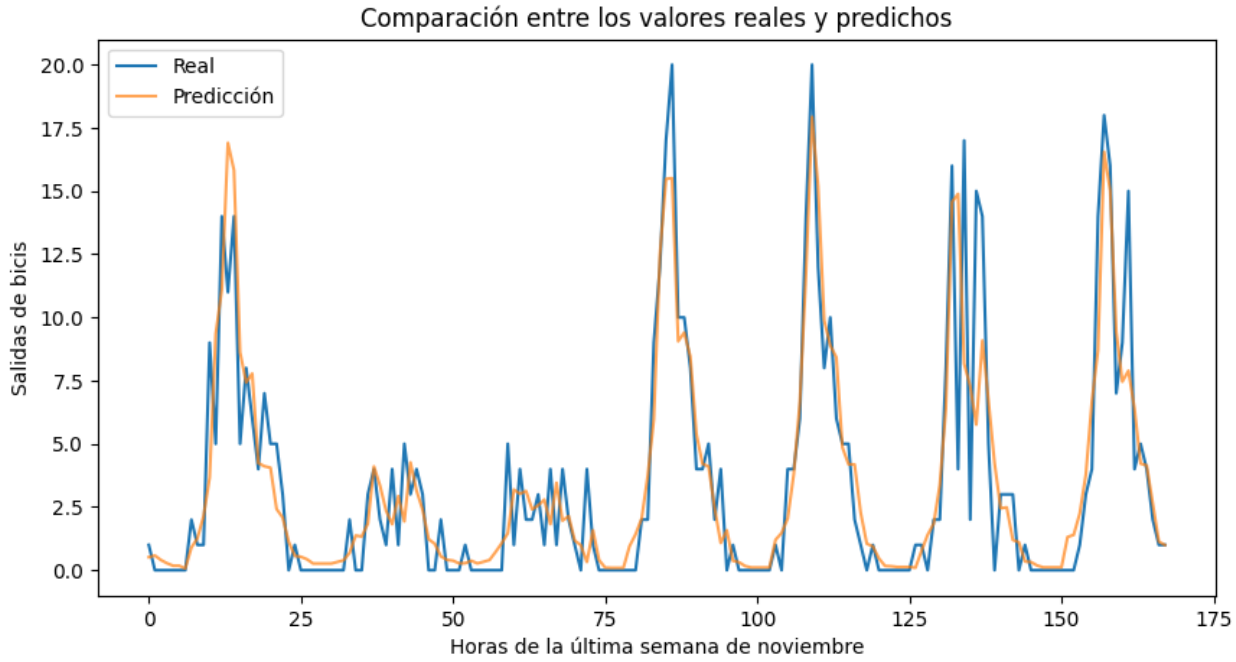


Figura 18: Predicciones de MLP vs. datos reales de la ultima semana de noviembre de 2023

El rendimiento de los modelos no lineales es bastante similar. Si los comparamos en cuanto a coste temporal, todo el proceso de entrenamiento de cada modelo no llega en ningún caso a los 15 minutos, y es algo que también depende de los hiperparámetros que queramos explorar. En cualquier caso es un coste perfectamente asumible para una aplicación real que prediga hora a hora.

En definitiva, podríamos escoger el **Perceptrón Multicapa (MLP)** como mejor modelo, ya que es el que minimiza el error cuadrático medio.

## 8.2. Modelo lineal Vs. no lineal (Análisis de interpretabilidad)

En esta sección vamos a comparar un modelo lineal (en este caso la regresión lineal) con un modelo no lineal (MLP) para ver con ejemplos concretos dónde y por qué es mejor el modelo no lineal.

Reentrenamos los modelos para predecir un día concreto. Tomamos el último día del dataset para que ambos modelos se entrenen con todo el histórico de datos desde 2022. Obtenemos los siguientes resultados.

| Model | R2 Score | RMSE |
|---|---|---|
| Linear Regression | 0.605522 | 3.5381 |
| MultiLayer Perceptron (MLP) | 0.8948 | 1.8265 |

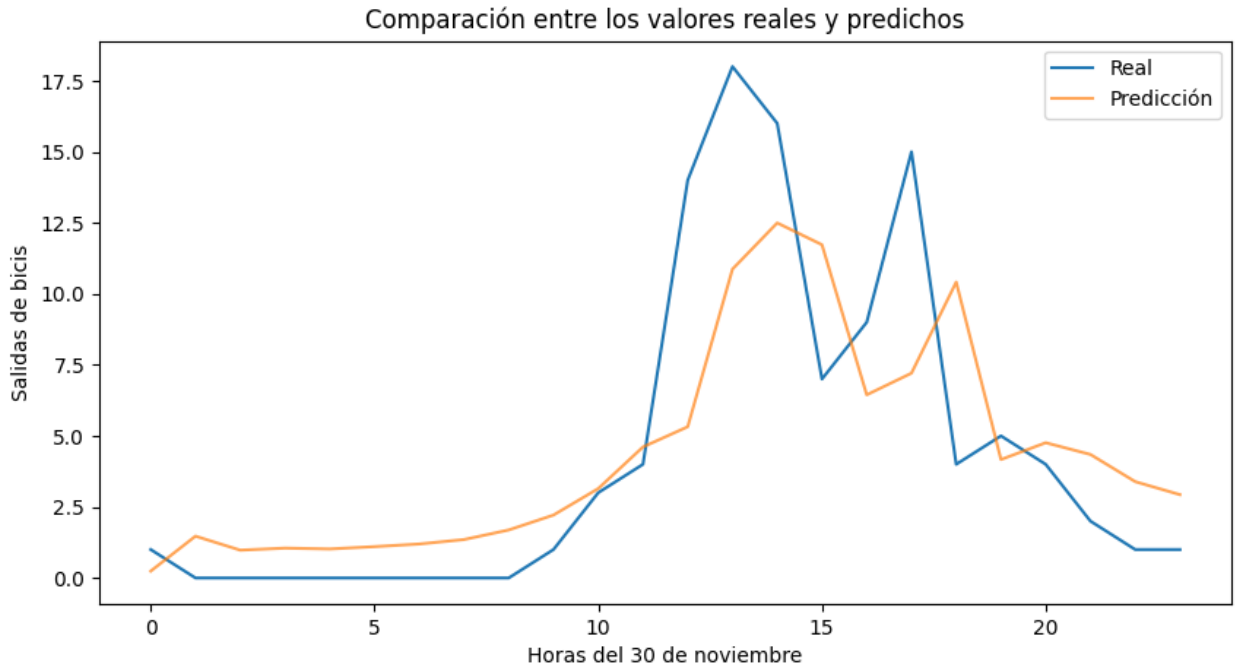


Figura 19: Gráfica de la predicción vs datos reales del dia 30 de noviembre de 2023 según el modelo (Regresión lineal)

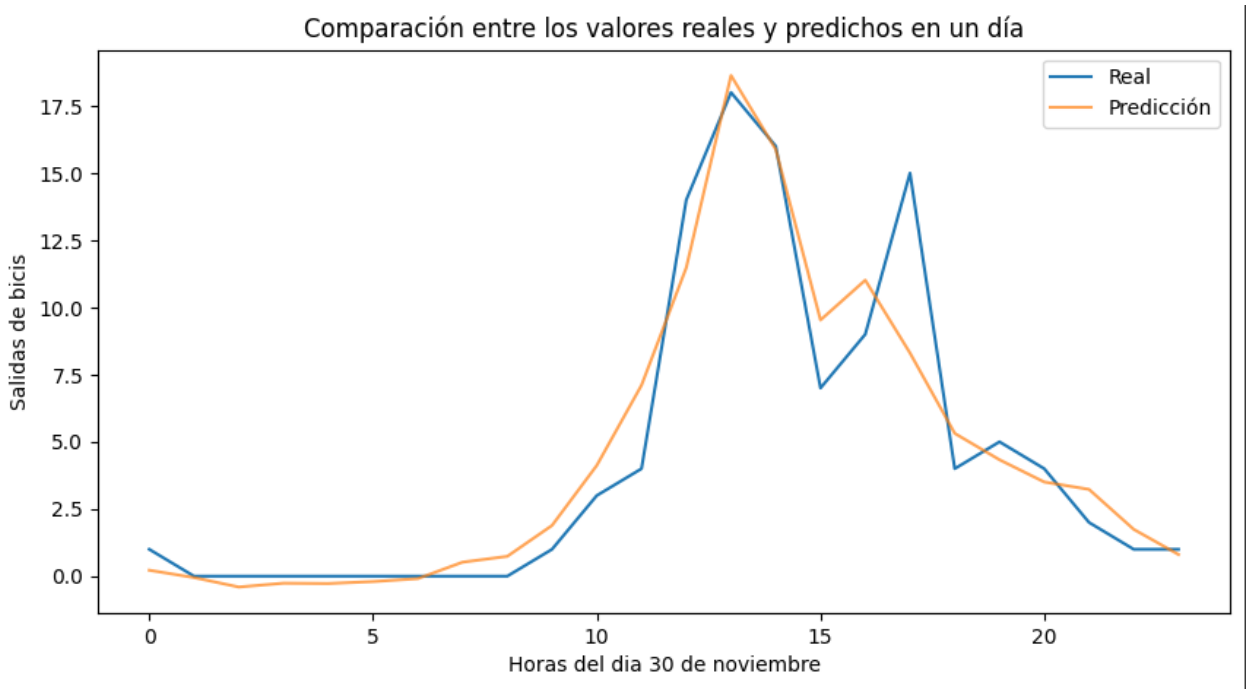


Figura 20: Gráfica de la predicción vs datos reales del dia 30 de noviembre de 2023 (MLP)

La predicción a un día, como era de esperar, es mucho mejor en el modelo MLP, sin embargo, la regresión lineal captura en cierto modo la tendencia de demanda. Algo interesante que vemos en el caso de la regresión linea, es que parece que las predicciones vayan çon retardo", y esto es algo que probablemente sea producido por las variables *lagged* y el número inicial de bicicletas. Estas variables tienen relaciones con la variable objetivo que la regresión lineal capta en cierta medida.

El perceptrón multicapa realiza unas predicciones realmente buenas. Es capaz de predecir el pico de demanda que se produce más o menos a la hora de comer (que coincide cuando la gente sale de clase), no obstante le cuesta más el rebote que se produce por la tarde, pero en general se ajusta bien a los datos reales.

Finalmente vamos a visualizar la importancia de permutación de las variables que utilizan ambos modelos para predecir. Esta técnica implica cambiar el orden de los valores de cada característica y medir el cambio en el rendimiento del modelo. Las características cuyo cambio en el orden afecta más el rendimiento se consideran más importantes.

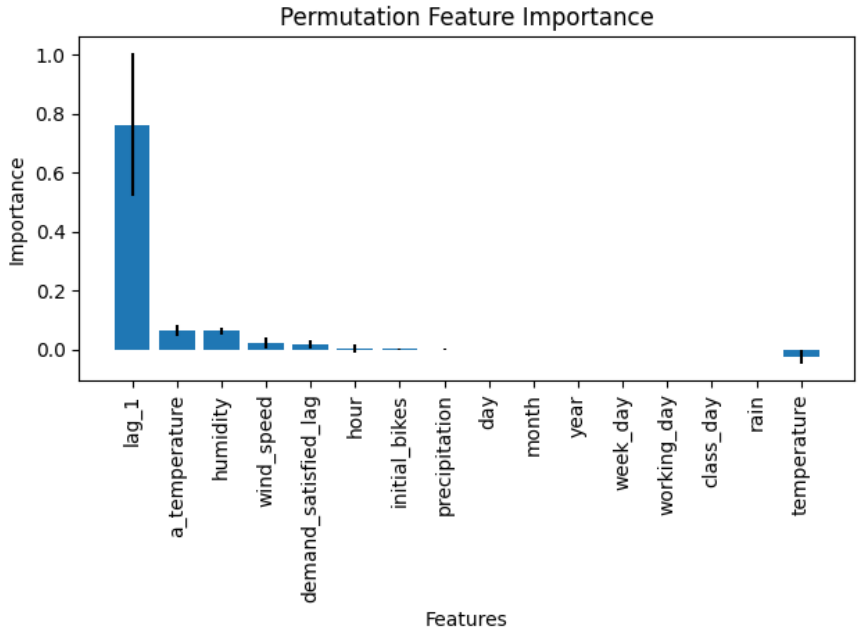


Figura 21: Importancia de permutación de las características para la regresión lineal

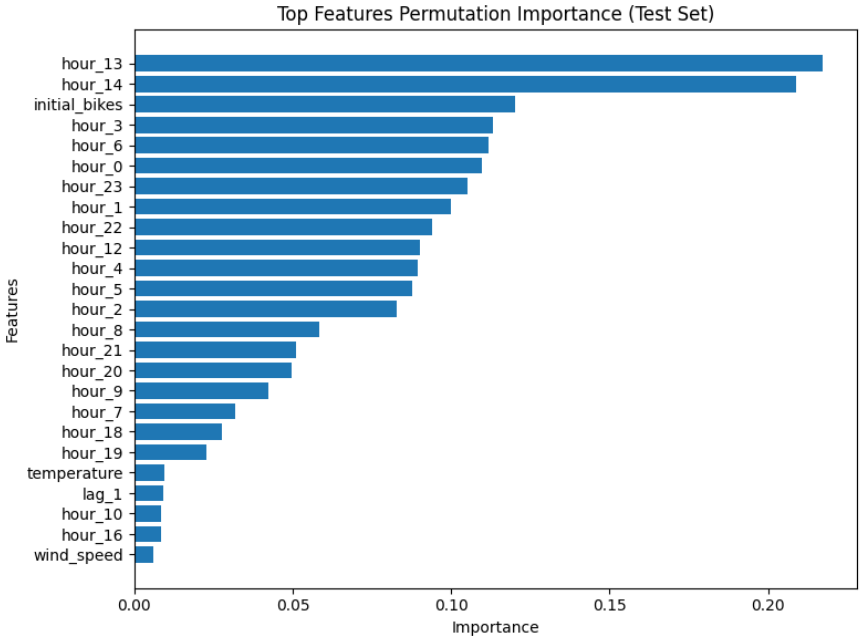


Figura 22: Importancia de permutación de las características para la regresión lineal

Vemos como, efectivamente, en el caso de la regresión lineal, la variable retardada que indica el número de las salidas de la hora anterior es con creces la más significativa. Esto es debido a que es la que guarda con la variable objetivo la relación que más se acerca a ser lineal.

Para el perceptrón multicapa, hay que tener en cuenta primero que usa variables que han sido binarizadas. No obstante podemos ver claramente como las variables temporales son cruciales para realizar las predicciones. Este es un ejemplo que confirma que el MLP ha sido capaz de captar relaciones lineales, ya que, como veíamos al principio del informe, las horas no están linealmente relacionadas con el número de salidas. Además, nótese también la importancia de características como el número inicial de bicicletas, y, en menor medida, algunas variables meteorológicas.

# 9. Conclusiones

Con este proyecto, tenemos que destacar en primer lugar el aprendizaje adquirido. Cuando pensamos en el problema a tratar no éramos conscientes de la magnitud que podía llegar a tener.

Desde luego, no ha sido simple. Durante el desarrollo de la práctica hemos ido comprendiendo que la naturaleza de nuestro problema es mucho más compleja de lo que pensábamos. Comprender que estamos ante un problema de análisis de una serie temporal, que además presenta patrones periódicos, fue lo que nos permitió modelar el problema de forma más adecuada. No estamos antes un problema cuyos datos son observaciones .ªleatoriasz eso añade un extra de complejidad. Durante el desarrollo hemos aprendido que existen modelos clásicos como ARIMA que también son interesantes para este problema, por no hablar de las medias móviles y muchos otros factores que pueden ser tenidos en cuenta.

En definitiva, son muchas las extensiones que se pueden hacer y las posibilidades por explorar. De hecho, nos hemos dado cuenta de que en el aprendizaje automático son infinitas, y no hay una solución óptima global para un problema. Debemos reconocer que lo que presentamos en este informe es una aproximación del problema. Podríamos haber explorado más opciones, pero o bien no son del alcance de este proyecto, o bien no hemos sido capaces de implementarlas.

Estamos muy satisfechos por haber logrado construir modelos que son, en mayor o menor medida, capaces de predecir las bicicletas que salen de la estación que tenemos debajo de la universidad. Consideramos además que este trabajo es una base que puede ser extensible a proyectos de mayor escala.

# Referencias

[1] UCLA: Statistical Consulting Group. *Regression Models for Count Data*. [En línea]. Disponible en: `https://stats.oarc.ucla.edu/stata/seminars/regression-models-with-count-data/`.

[2] Alejandro Carol Campreciós, *Bicing Stats*, Trabajo de Fin de Grado, Universidad Politécnica de Cataluña, 2016. [En línea]. Disponible en: https://upcommons.upc.edu/bitstream/handle/2117/90448/108297.pdf?sequence=1&isAllowed=y.

[3] Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, y Rafael Banchs. *Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system*. Pervasive and Mobile Computing, 6(4):455-466, 2010. Human Behavior in Ubiquitous Environments: Modeling of Human Mobility Patterns. ISSN 1574-1192. doi: https://doi.org/10.1016/j.pmcj.2010.07.002.

[4] Bustamante, X., Federo, R., & Fernández-I-Marin, X. (2022). *Riding the wave: Predicting the use of the bike-sharing system in Barcelona before and during COVID-19*. Sustainable Cities and Society, 83, 103929. https://doi.org/10.1016/j.scs.2022.103929

[5] Gabriel Martins Dias, Boris Bellalta, Simon Oechsner. (2015). *Predicting Occupancy Trends in Barcelona's Bicycle Service Stations Using Open Data*. [eprint arXiv:1505.03662].

[6] Jiang W. (2022). *Bike sharing usage prediction with deep learning: a survey*. Neural Computing & Applications, 34(18), 15369–15385. https://doi.org/10.1007/s00521-022-07380-5