

**Universidad Nacional del Centro de la Provincia de Buenos Aires**  
**Facultad de Ciencias Exactas**

Ingeniería de Sistemas



**Tesis de Grado**

Asistente inteligente para estacionamiento en espacios urbanos

**Autor**

Luciano Tangorra

**Directores**

Dr. Juan M. Rodriguez

Dr. Marcelo G. Armentano

## **Agradecimientos**

Quisiera agradecer en primer lugar a mis directores, Juan Manuel y Marcelo, por su constante guía y apoyo a lo largo de este trabajo. También quiero agradecer a todos los integrantes del ISISTAN, por brindarme valiosos consejos y hacer de este trabajo una experiencia aún más enriquecedora y a todos los docentes y participantes de la Facultad de Ciencias Exactas por estar presentes en todo momento y hacer de esta etapa un momento inolvidable.

Quisiera agradecer a mi familia, por todo el esfuerzo realizado para que pudiera llevar adelante mis estudios y, particularmente, por toda la ayuda y cariño que siempre me brindaron. A todos los amigos que hice durante esta etapa de mi vida y con quienes compartimos este proceso. A mis amigos de toda la vida, que siempre están ahí para escucharme. Quisiera agradecer especialmente a Milagros, por el apoyo que me brindo en esta última etapa de la carrera y por ser siempre una persona positiva.

Finalmente, agradezco al Consejo Interuniversitario Nacional (CIN) por el apoyo financiero para este trabajo.

# Índice

<b>Capítulo 1: Introducción</b>	1
1.1. Motivación	1
1.2. SUMO y el tráfico en la ciudad de Tandil	2
1.3. Objetivos	4
1.4. Organización del trabajo	4
<b>Capítulo 2: Marco teórico</b>	6
2.1. Métricas	6
2.1.1. Clasificación	6
2.1.2. Regresión	9
2.2. Machine learning	10
2.2.1. Perceptrón	11
2.2.2. Funciones de activación	12
2.2.3. Optimizadores	15
2.2.4. Embeddings	16
2.2.5. Funciones de pérdida	17
2.2.6. Batch	18
2.3. Modelos de machine learning	20
2.3.1. Árboles de decisión	20
2.3.2. Random forest (Bosques aleatorios)	21
2.3.3. Regresión lineal y regresión logística	22
2.3.4. Aprendizaje profundo (Deep Learning)	24
2.3.4.1. Red neuronal profunda	24
2.3.4.2. Red neuronal recurrente (RNN)	25
<b>Capítulo 3: Trabajos relacionados</b>	27
3.1. Sensores especializados	27
3.2. Sensores móviles	28
3.3. Carga de datos manual	31
3.4. Conclusión	32
<b>Capítulo 4: Dataset</b>	34
4.1. Obtención de datos	34
4.1.1. Dataset 2018	34
4.1.2. Dataset web scraping	36
4.1.3. Dataset 2019	40
4.2. Análisis de datos	42
4.2.1. Problemas iniciales	42
4.2.2. Identificadores	43
4.2.3. Diagramas de caja	44

4.2.4. Máximos por cuadra	45
4.2.5. Cantidad de operaciones	47
4.2.5.1. Por cuadra	47
4.2.5.2. Por día de semana	48
4.2.5.3. Por hora	49
4.2.6. Momentos de mayor ocupación	49
4.2.7. Estadísticas por mes	50
4.2.8. Estacionamientos por mes	52
4.2.9. Estacionamientos por día de semana	53
4.2.10. Mapas de calor	56
4.2.11. Conclusión	59
<b>Capítulo 5: Predictores</b>	60
5.1. Problemas iniciales	60
5.2. Pre-procesamiento de los datos	61
5.3. Regresores	63
5.3.1. Modelos	64
5.3.2. Comparación entre modelos de regresión	74
5.4. Clasificadores	75
5.4.1. Modelos	77
5.4.2. Comparación entre los modelos de clasificación	78
5.5. Complejización de los modelos de deep learning	79
5.6. Comparación entre modelos de regresión y clasificación	80
<b>Capítulo 6: Conclusión</b>	82
6.1. Implicancias	83
6.2. Limitaciones	83
6.3. Trabajos futuros	84
6.3.1. Aplicación para dispositivos móviles	84
6.3.2. Ampliación de datos	84
<b>Bibliografía</b>	85

# **Capítulo 1: Introducción**

Administrar los lugares para estacionar es uno de los grandes problemas en la movilidad urbana. El uso de horarios restringidos o el estacionamiento medido son las maneras en las cuales los municipios han manejado tradicionalmente este problema. Aún así, hallar lugar para estacionar es un problema creciente debido a que el parque automotor está en crecimiento constante [44][45].

En un principio este problema no era tan grave ya que la cantidad de vehículos era relativamente poca, siendo sencillo conseguir lugar para estacionar. Mientras fue avanzando el tiempo, más gente comenzó a utilizar este medio de transporte y comenzaron a aparecer problemas a la hora de encontrar lugar libre. Esto se convirtió en una tarea a realizar por los conductores, ya que deben tener en cuenta el tiempo que conlleva encontrar lugar en la ciudad.

Además de que los conductores deben ocupar su tiempo en buscar lugar para estacionar, se generan al menos tres efectos negativos colaterales:

- 1) contaminación atmosférica, la cual perjudica al medioambiente,
- 2) contaminación sonora, que perjudica y molesta a los ciudadanos, y
- 3) embotellamientos, los cuales afectan el tráfico y generan descontento en los conductores.

El desarrollo de nuevas tecnologías de Internet de las cosas, dispositivos móviles y parquímetros inteligentes están agregando nuevas herramientas para proveer más servicios a los ciudadanos y llevar un control más detallado del uso de los lugares de estacionamiento.

La mayoría de los trabajos existentes que abordan el problema del estacionamiento analizan el problema en el contexto de ciudades grandes, utilizando diversos dispositivos costosos, como en [3], o permitiendo al usuario participar de la obtención de datos que luego serán utilizados para brindar posibles soluciones, en el caso de [5][6][7][8][9]. Esto es aún más complicado de implementar en ciudades medianas o pequeñas ya que, normalmente, existe una limitante de presupuesto, además de una cantidad de usuarios relativamente escasa, la cual podrá apenas brindar una pequeña cantidad de datos.

En este trabajo se propone el desarrollo de una solución para estacionamiento inteligente basada principalmente en datos obtenidos de los parquímetros de la ciudad de Tandil, ciudad en la cual todavía no se ha implementado ninguna solución a esta problemática.

## **1.1. Motivación**

Una porción significativa de las ciudades está destinada a espacio de estacionamiento [1], por ejemplo, en 18% de espacio en la ciudad de Nueva York está destinado al estacionamiento, en Londres el 16%, aunque puede llegar a cerca del 80% en ciudades como Los Ángeles o Melbourne. Las soluciones de estacionamiento inteligente, o Smart Parking, proponen integrar

diversas tecnologías para mejorar la administración de dichos espacios. Sin embargo, el estacionamiento inteligente es un área multidisciplinaria, ya que integra diversos actores con diferentes objetivos [2]. Por un lado, se encuentra el ciudadano/conductor, que espera poder conseguir un lugar de estacionamiento lo más cercano posible a su destino. Además, puede tener intereses económicos, como estacionar en zonas sin estacionamiento medido. Por otro lado, el municipio que puede estar interesado en reducir el tiempo que los ciudadanos estacionan en zonas de alto tráfico para que más ciudadanos puedan acceder a esos puntos de estacionamiento, reducir el tráfico y la contaminación generada por conductores buscando lugar para estacionar, detectar patrones anómalos de estacionamiento (por ejemplo, más autos estacionados que espacios disponibles) que pueden indicar conductas ilegales como estacionamiento en doble fila y/o lugares prohibidos para una mejor asignación de los inspectores de tránsito. También, puede haber intereses privados, como estacionamientos privados o concesiones de estacionamiento medido que podrían brindar sus servicios o detectar lugares donde los conductores no fichan sus autos en los parquímetros.

Muchas de las soluciones actuales se basan en la instalación de sensores especializados. Algunos de estos sensores incluyen sensores infrarrojos, de ultrasonido, y magnéticos [3]. Sin embargo, según Evenepoel et. al. [4] el costo de instalar y operar estas redes de sensores puede ser prohibitivo. En particular, a medida que se incrementa el número de sensores a instalar, el costo de los sensores baja linealmente de 150 euros a unos 50 euros, pero el costo de operar los mismos crece de forma exponencial. Esto presenta una limitante a la hora de implementar estas soluciones en ciudades medianas o pequeñas, con presupuestos limitados. Para solucionar esta y otras limitantes de las infraestructuras de sensores, diversos trabajos han analizado la posibilidad de utilizar los dispositivos móviles de los conductores para la recolección de datos de sus sensores. Esta metodología de recolección de datos de sensores se conoce como CrowdSensing. Por ejemplo, en [5] se analiza la posibilidad de utilizar los magnetómetros, sensores de campos magnéticos, de los dispositivos móviles de los conductores para mapear qué lugares de estacionamiento se encuentran libres. PhonePark [7][8] propone utilizar el GPS y acelerómetro de los teléfonos móviles para detectar el comportamiento de los conductores, en particular dónde estacionan y cuándo liberan el espacio, de manera de generar en tiempo real informes de disponibilidad de estacionamiento. Otros ejemplos son [9][6], en estos casos es el usuario el encargado de informar manualmente el estado de un lugar de estacionamiento. En estos casos, las aplicaciones están pensadas para playas de estacionamiento y a los usuarios se le asigna un nivel de confianza. En este sentido, el incentivo para los usuarios para participar es que la aplicación facilita encontrar lugar para estacionar.

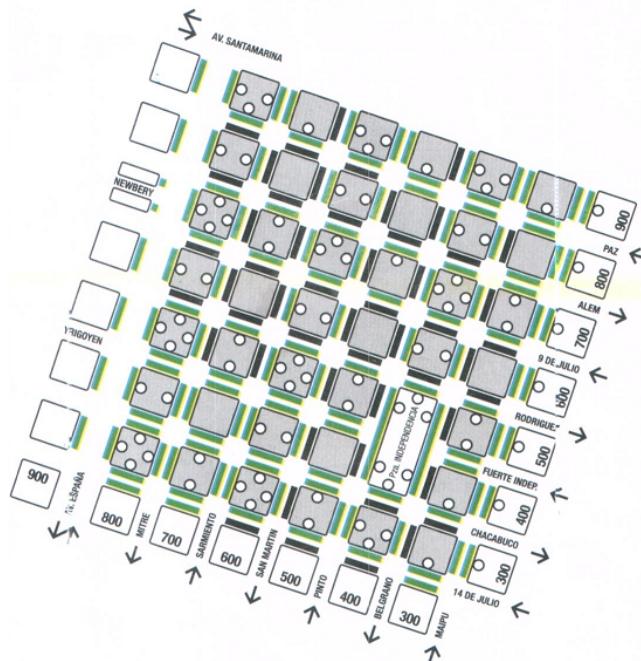
## **1.2. SUMO y el tráfico en la ciudad de Tandil**

El Sistema Único de Movilidad Ordenada (SUMO) es un sistema que unifica el pago de estacionamiento medido, de los boletos de colectivo y de las infracciones. El sistema se

encuentra actualmente implementado sólo en la ciudad de Tandil. Para el caso del estacionamiento medido, se puede abonar de dos formas:

1. Una, en dónde través de la aplicación se deberá ingresar la cuadra en la que se estacionó el vehículo junto con la patente del mismo.
2. La otra, a través de los parquímetros de la ciudad, ubicados en las principales cuadras del centro de la ciudad (Figura 1.1), en la cual se ingresará con la tarjeta personal y se marcará como un ingreso o una salida según corresponde.

El horario de estacionamiento medido es de 10:00 a 20:00 para los días de semana y de 10:00 a 14:00 para los sábados. Los domingos y feriados el estacionamiento medido es libre. Aún así, se cuentan los ingresos y las salidas en horarios fuera del dispuesto, aunque sin realizar ningún cobro al usuario. La Figura 3.1 presenta un plano donde se destacan, en color verde, las cuadras donde el estacionamiento está permitido, en negro donde está prohibido y con círculos blancos la ubicación de los parquímetros ubicados en la ciudad. Además, hay un parquímetro ubicado en la terminal de la ciudad, fuera de la zona céntrica, utilizado para abonar infracciones.



**Figura 1.1:** Plano del estacionamiento medido y los parquímetros habilitados en Tandil.

Este sistema centralizó la información sobre todos sus servicios, así como recorridos de colectivos y visualización de cuentas personales, en el sitio web <http://www.gpssumo.com/>. Aquí se podrán observar en detalle todos los parquímetros ubicados en la ciudad, junto con su dirección exacta.

Aún con este sistema de cobro implementado, el volumen de personas que buscan lugar para estacionar sigue siendo muy grande. Varios conductores prefieren estacionar en doble fila, lo cual está prohibido y perjudica el tráfico habitual, en vez de buscar lugar para estacionar, ya que esto suele ser una tarea demandante en lo referente al tiempo. Esto perjudica la experiencia de los ciudadanos al manejar esta zona de la ciudad, además de incrementar la posibilidad de accidentes automovilísticos.

El lugar donde se encuentra la mayor cantidad de espacio para estacionar, por ende el que más lugares libres suele tener, es la Plaza Independencia. A pesar de esto, la gente suele evitarlo ya que suele haber gente que ilegalmente cobra, aparte del cobro de los parquímetros, a las personas por estacionar el vehículo en ese lugar.

### **1.3. Objetivos**

La hipótesis principal del trabajo es que se puede extraer información sobre las acciones de los conductores a través de los parquímetros de la ciudad de Tandil para detectar cuándo estacionan o liberan un lugar de estacionamiento. Se buscarán patrones de estacionamiento de los conductores para mejorar las predicciones realizadas, además de realizar un análisis de los datos obtenidos para obtener valores que serán útiles a la hora de entrenar modelos de machine learning.

Se utilizarán los parquímetros la ciudad ya que, además de poseer un gran volumen de información de todos los usuarios que ingresan en el día a día, son sistemas que se encuentran actualmente instalados, por lo que el costo de implementación es nulo. Además, a comparación de utilizar sensores de dispositivos móviles, no se necesita ningún usuario en el sistema para poder realizar las predicciones, por lo que el volumen de conductores ingresados a nuestro sistema no será un problema.

El objetivo final de este trabajo es utilizar la información ofrecida por los parquímetros en tiempo real para generar predicciones de qué lugares de estacionamiento se encontrarán libres en un momento determinado. En particular el proyecto tiene dos objetivos específicos:

1. Recolectar y analizar datos de parquímetros de la ciudad de Tandil para determinar las acciones de los conductores y los hábitos de estacionamiento.
2. Desarrollar modelos de machine learning capaz de predecir lugares libres para estacionar.

### **1.4. Organización del trabajo**

En esta sección se proveen detalles acerca de cómo se estructura el presente informe, ofreciendo una descripción introductoria de las temáticas abordadas en cada uno de los capítulos.

En el capítulo 1 se corresponde con la introducción al trabajo de grado. En él se encuentran la motivación que llevó a la realización de este trabajo, una descripción de la situación actual en la ciudad de Tandil y los objetivos perseguidos.

En el capítulo 2 se explica detalladamente los aspectos teóricos que se emplearán a lo largo del trabajo. No se proveerán detalles relacionados al problema a resolver, sino que se explicará la teoría detrás de lo realizado. Se verán las herramientas y conceptos de machine learning necesarias para comprender lo realizado.

Luego, en el capítulo 3, se hará una breve introducción a los trabajos relacionados. Se plantean sus ventajas y desventajas, además de compararlas con la solución propuesta en este trabajo.

En el siguiente capítulo (capítulo 4), se explicará en detalle cómo fueron obtenidos los datasets utilizados, además de detallar los análisis realizados y qué información brindaron. Se hizo especial énfasis en buscar patrones y en obtener el mejor conjunto de atributos para utilizar al entrenar los modelos.

Con respecto al desarrollo de los modelos de predicción, en el capítulo 5 se discutirán los problemas encontrados al comenzar la implementación, el pre procesamiento realizado y sus ventajas. Además, se explicarán los tipos de modelos realizados junto a sus resultados, finalizando con una comparación y conclusión de los resultados obtenidos.

En el capítulo 6 se podrá observar la conclusión final del trabajo, resaltando los aspectos más importantes y las implicancias y limitaciones encontradas. También se especifican aquellas cuestiones cuyo alcance estuvo por fuera de este trabajo, las cuales sirven para el desarrollo de trabajos futuros, posibilitando la ampliación del actual trabajo.

## **Capítulo 2: Marco teórico**

En este capítulo se presentan y describen las técnicas de aprendizaje de máquina necesarias para generar los modelos del comportamiento de los conductores, desde un simple modelo de árbol de decisión, hasta redes neuronales recurrentes. Se explicará desde el funcionamiento de una simple neurona, logrando que el lector posea el conocimiento necesario para poder comprender las decisiones tomadas a lo largo de todo el trabajo. Además, se detallarán algunas de las métricas existentes, las cuales permiten medir que tan bien predice un modelo.

### **2.1. Métricas**

Las métricas son cálculos matemáticos que permiten medir la performance de un modelo. Estos cálculos utilizan tanto los valores predichos como los reales y brindan un resultado, el cual ofrecerá cierta información sobre el modelo entrenado. Existen métricas tanto para problemas de clasificación (predecir entre clases) como de regresión (predecir valores numéricos). Se debe evaluar la mejor opción en cada problema según los resultados que se deseen del modelo entrenado. A continuación se describirán las métricas utilizadas en este trabajo.

#### **2.1.1. Clasificación**

Para poder comprender los cálculos realizados por cada métrica, es conveniente comprender qué es y cómo funciona una matriz de confusión [15]. Una matriz de confusión es una tabla que describe el rendimiento de un modelo, donde se conocen los valores a predecir. Estos últimos son las posibles clases de un problema, por ejemplo si se desea conocer qué candidato votó la gente, las clases serían todos los posibles candidatos. Se le llama matriz de confusión ya que permite detectar fácilmente dónde el sistema se confunde de clases.

La matriz se divide en cuatro diferentes categorías, dependiendo si el valor predicho fue acertado o no. Estas categorías son:

- Positivos Verdaderos (TP): se predijo correctamente la clase como positiva.
- Negativos Verdaderos (TN): se predijo correctamente la clase como negativa.
- Positivos Falsos (FP): se predijo falsamente la clase como positiva.
- Negativos Falsos (FN): se predijo falsamente la clase como negativa.

		Positivo	Negativo	
		Positivo	FN	TP + FN
Valor actual	Positivo	TP		
	Negativo	FP	TN	FP + TN

**Figura 2.1:** abstracción de una matriz de confusión.

Aunque en este ejemplo se representan dos resultados (para dos posibles valores de salida), la matriz de confusión puede ser escalada a múltiples salidas, donde un positivo sería una clase en específico y se compararía contra las negativas, es decir, el resto de las clases. Esto se realizaría por cada clase del modelo.

- **Accuracy (Exactitud)**

Una de las métricas más utilizadas es el accuracy, la cual indica el porcentaje total de elementos clasificados correctamente. Tiene sus limitaciones, ya que puede predecir erróneamente si la cantidad de datos por clases se encuentra desbalanceada, es decir, si se poseen varios elementos de una clase y una mínima cantidad de otra [16]. La fórmula para calcular esta métrica es:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precisión**

Esta métrica representa la cantidad de positivos predichos que realmente son verdaderos. Busca resolver la pregunta: ¿Qué proporción de valores positivos predichos fue correcta? La fórmula para calcular este valor es:

$$\text{precision} = \frac{TP}{TP + FP}$$

La precisión es especialmente importante cuando los falsos positivos no son deseados [17]. Por ejemplo en el análisis de spam en mails, un falso positivo en este caso implicaría que el usuario podría perderse un mail importante.

#### - Recall (Sensibilidad)

Esta métrica representa la cantidad de positivos predichos sobre todos los verdaderos reales. Busca responder a la pregunta: ¿Qué proporción de positivos reales fue predicha correctamente? La fórmula para calcular este valor es:

$$\text{recall} = \frac{TP}{TP + FN}$$

El recall es especialmente importante cuando los falsos negativos no son deseados [17]. Por ejemplo, en el caso de análisis médico de COVID-19, un paciente infectado realiza el test y es predicho como negativo. Dependiendo el caso, el costo de estas equivocaciones puede ser extremadamente alto.

#### - F1-Score

Se emplea para la obtención de un valor único ponderado de la precisión y la sensibilidad. Este resultado es considerado como la media armónica que combina los valores de precision y recall. La fórmula para calcularlo es:

$$f1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Es aconsejable obtener y enfocarse en un solo valor para optimizar el modelo, por lo que esta métrica es una buena solución si se desea utilizar precision y recall de igual forma [17].

#### - Problemas con clases desbalanceadas

Como se detalló anteriormente, si el modelo tiene desbalance de clases, el accuracy no es una buena métrica cuando se quiere saber la calidad de un modelo de clasificación (hay clases mayoritarias y minoritarias). En estos casos, al utilizar dicha métrica, un modelo podría obtener un valor alto de accuracy prediciendo siempre la clase mayoritaria [16]. Supongamos el caso de que se tienen diez datos de infecciones, en donde nueve son negativos y uno es positivo. Si tomamos un modelo, el cual una vez entrenado predice siempre la clase mayoritaria, al utilizar accuracy fácilmente se podría obtener un resultado del 90%. Esto no es cierto, ya que nunca se podría predecir el valor positivo (para el caso de infecciones es muy importante).

Para solucionar este problema, es posible utilizar las demás métricas enunciadas anteriormente y calcular sus valores para cada clase.

Al tener desbalance de clases, se deberá conocer los pesos que se le debe asignar a cada clase para simular ese balance. Siguiendo el ejemplo anterior, a la clase positiva se le asignaría un peso de 9 y a la clase minoritaria un peso de 1, por lo que cada verdadero positivo contará como 9 falsos negativos.

Luego de determinar los pesos de cada clase, se calcularán los resultados de la métrica deseada y se aplicará una media ponderada. Es decir:

$$M_{total} = \frac{P_1 * M_1 + P_2 * M_2 + \dots + P_n * M_n}{P_1 + P_2 + \dots + P_n}$$

Donde  $P_i$  significa el peso asignado a la clase  $i$  y  $M$  es la métrica elegida.

### **2.1.2. Regresión**

A diferencia de los problemas de clasificación, en la regresión, al tener como salida un valor continuo, se utilizan métricas que ayuden a comprender que tan distantes se encuentran los datos predichos de los verdaderos.

- **Error absoluto medio (MAE)**

El error absoluto medio es una medida que hace referencia a la diferencia entre dos variables continuas. Se calcula como el promedio del valor absoluto de la diferencia entre el valor predicho y el valor real. Por ejemplo, un error de 10 es exactamente el doble de malo que un error de 5. Es especialmente útil cuando se desea corregir con la misma importancia tanto los pequeños como los grandes errores [18]. La fórmula para calcular este valor es:

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

- **Error cuadrático medio (MSE)**

El MSE representa la diferencia al cuadrado entre dos variables continuas. Se calcula similar al MAE, salvo que se utiliza el cuadrado de la diferencia entre los valores. En este caso, los errores grandes empeoran en gran medida este valor, ya que se encuentran elevados al cuadrado. Siguiendo con el ejemplo utilizado en MAE, un error de 10 resultaría en un MSE de 100, mientras que un error de 5 daría un MSE de apenas 25. Se observa que, en vez de ser el doble de grande el error, es cuatro veces peor. La fórmula para calcularlo es:

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

El principal problema de utilizar esta métrica es que, al elevar los valores al cuadrado, se obtienen resultados con sus respectivas unidades elevadas al cuadrado, por lo que se dificulta, en algunos casos, interpretarlos [18].

- **Raíz del error cuadrático medio (RMSE)**

El RMSE es similar al MSE, salvo que se le aplica la raíz al resultado. Esto permite interpretar de forma sencilla los resultados [18]. Al igual que el MSE, los errores grandes tienen mayor relevancia que los errores pequeños, afectando así el resultado de la métrica. La fórmula para calcular estos valores es:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}$$

- **Coeficiente de determinación ( $R^2$ )**

El coeficiente de determinación, también conocido como R cuadrado, representa la calidad del modelo para replicar los resultados. Se utiliza cuando se desea obtener un valor entre menos infinito y 1, es decir, obtener la calidad predictiva del modelo en valores similares a porcentajes (de 0 a 1) [19]. En el caso de que el valor resultante sea negativo, significa que el modelo implementado predice peor que el modelo base utilizado. La fórmula para calcular este valor es la siguiente:

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)}$$

donde el baseline representa un modelo que predice el promedio del conjunto de datos.

## 2.2. Machine learning

El aprendizaje automático es un subcampo de las ciencias de la computación, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Se dice que un agente (o modelo) aprende cuando su desempeño mejora a partir de la experiencia, es decir, de los datos históricos. Los modelos resultantes deben ser capaces de generalizar comportamientos o patrones sobre los datos, para luego predecir correctamente sobre datos nunca antes vistos. Estos modelos, a su vez, se dividen, principalmente, en dos categorías: supervisados y no supervisados.

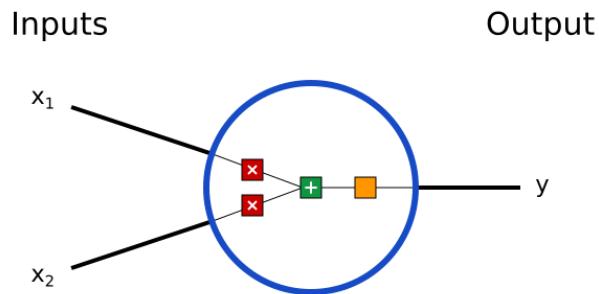
Los primeros establecen una relación entre las entradas y las salidas deseadas del sistema. La base del conocimiento de este tipo de sistemas está formada por ejemplos etiquetados, es decir, se conocen las entradas y las salidas de cada dato.

Los modelos no supervisados en cambio, se llevan a cabo sobre un conjunto de datos formado únicamente por las entradas al sistema, sin tener información sobre las categorías (o salidas) de estos datos.

En este trabajo se utilizaron varios tipos de modelos supervisados ya que, como se verá más adelante, podremos formar los datos de entrada y de salida de la red a partir del dataset obtenido.

## 2.2.1. Perceptrón

El perceptrón es una estructura que trata de imitar el funcionamiento de una neurona. Esta estructura es la unidad básica de las redes neuronales. En resumen, un perceptrón toma valores de entrada, realiza alguna operación matemática sobre ellos y produce una salida. A continuación se podrá observar una neurona de dos entradas y una salida.



**Figura 2.2:** Representación interna de un perceptrón, en donde ingresan dos valores de entrada y genera un valor de salida.

Los procesos por los que pasa un perceptrón son los siguientes [20]:

- En primer lugar, se multiplica cada valor de entrada por un peso.

$$x_1 \rightarrow x_1^* w_1$$

$$x_2 \rightarrow x_2^* w_2$$

- Luego, se suman todos los valores resultantes, junto con un valor de bias ( $b$ ), en caso de ser requerido.

$$(x_1^* w_1) + (x_2^* w_2) + b$$

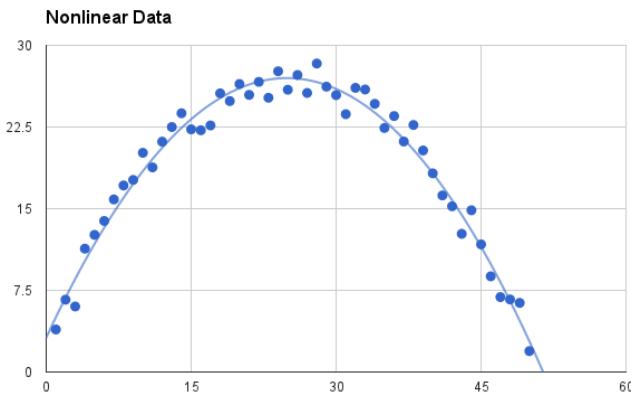
- Al final, se aplica una función de activación (detallada más adelante) y se obtiene un valor de salida según corresponda a la función designada.

$$y = f((x_1 * w_1) + (x_2 * w_2) + b)$$

## 2.2.2. Funciones de activación

El objetivo de las funciones de activación es definir la salida de un nodo, pudiendo modificar el valor obtenido para limitarlo entre ciertos rangos, o para mejorar la velocidad de entrenamiento de la red. En la figura 2.2, puede observarse que este valor representa el cálculo realizado antes de obtener la salida  $y$ .

Existen diversas funciones de activación y cada una brinda herramientas para resolver diversos problemas con deep learning. Pueden ser utilizadas, por ejemplo, en la última capa de un modelo de clasificación binario para mantener los valores en rangos entre 0 y 1 (valores de las clases) o en las capas ocultas de una red. Las funciones no lineales, por su parte, permiten encontrar patrones no lineales entre los datos, como se aprecia en el ejemplo de la figura 2.3.



**Figura 2.3:** representación de un problema con datos no lineales.

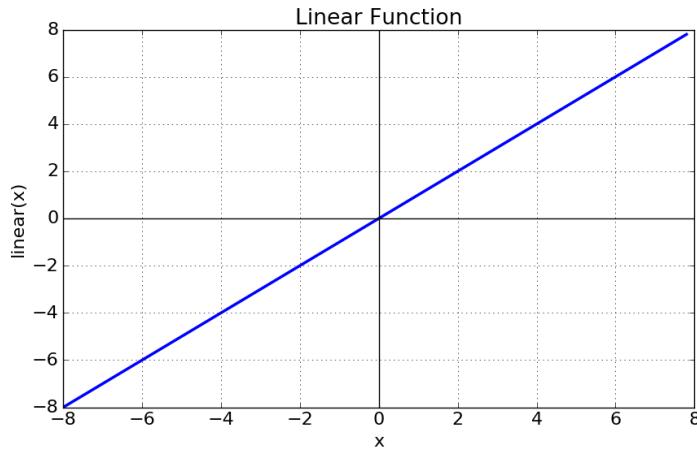
A continuación se describen algunas funciones de activación.

- **Linear**

La salida de este tipo de función es proporcional a la entrada, es decir, el rango de salida no está restringido. Para calcular esta función se aplica:

$$F(x) = x$$

Al ser una función lineal podemos observar que su derivada es constante, por lo que no es posible realizar propagación hacia atrás, ya que el gradiente no tiene relación con  $x$  (será siempre constante). Este problema se disipa si es utilizada luego de otras capas con funciones de activación no lineales. Por ejemplo, en los modelos de regresión se emplea como última capa para predecir un valor continuo de salida.



**Figura 2.4:** gráfica genérica de una función lineal.

### - Rectificador (ReLU)

El rectificador (rectified linear unit en inglés) es una función de activación no lineal que restringe el valor de salida entre 0 y más infinito. Se define como:

$$f(x) = \max(0, x)$$

La principal ventaja de este tipo de activación, además de ser no lineal y permitir la propagación hacia atrás, es que las salidas con valores negativos valdrán cero. Esto reducirá el total de neuronas a utilizar durante el entrenamiento, por lo que los cálculos se realizan aún más rápido que con otras funciones. A su vez, al ser un cálculo matemático sencillo, su complejidad temporal es muy baja [14].

La desventaja de esta función está ligada a su ventaja, ya que los valores obtenidos como cero tendrán, a su vez, gradiente cero, por lo que no se ajustarán sus pesos. Este problema puede causar que algunas neuronas dejen de responder y una parte de la red neuronal pase a un estado pasivo.



**Figura 2.5:** gráfica genérica de una función ReLU.

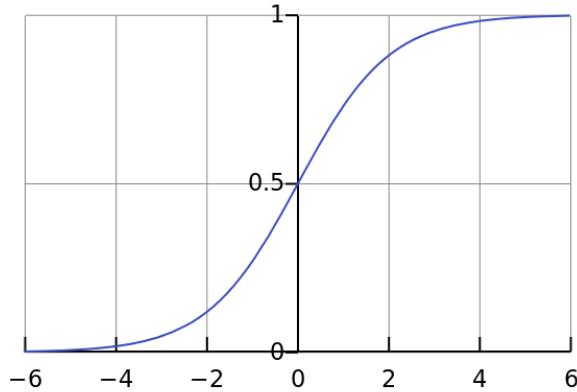
### - Función sigmoide

La función sigmoide es una función de activación no lineal, restringida en el rango entre 0 y 1, calculada de la siguiente forma:

$$f(x) = \frac{1}{1+e^{-x}}$$

Este tipo de función tiende a ubicar los valores de  $y$  al final de la curva, en la parte pseudo-horizontal, siendo ésta una de las principales razones de su utilización. Si se requiere predecir entre dos clases, la función sigmoide hará una clara distinción entre las predicciones ya que los valores resultantes de las clases se encontrarán entre 0 y 1 [14]. Esto no implica que el predictor obtendrá buenos resultados, sino que se observará claramente si la predicción resultó en la clase 0 o en la clase 1.

La principal desventaja es su costo computacional. A su vez, surge la problemática del gradiente que desaparece (vanishing gradient). Surge cuando los valores de  $x$  son demasiado bajos, por lo que el gradiente es casi nulo, evitando así que se modifiquen los pesos del modelo. Esto conlleva a que el modelo no aprenda.



**Figura 2.6:** gráfica genérica de una función sigmoide.

### - Softmax

La función softmax es una generalización de la función logística [14], utilizada en la última capa de un modelo de multi clasificación. Aporta la misma utilidad que la función sigmoide, pero es aplicada a clasificación multi-clase. Los valores resultantes de todas las neuronas suman 1 (100% de probabilidad), por lo que la clase predicha será la que posea mayor valor. Se calcula de la siguiente forma:

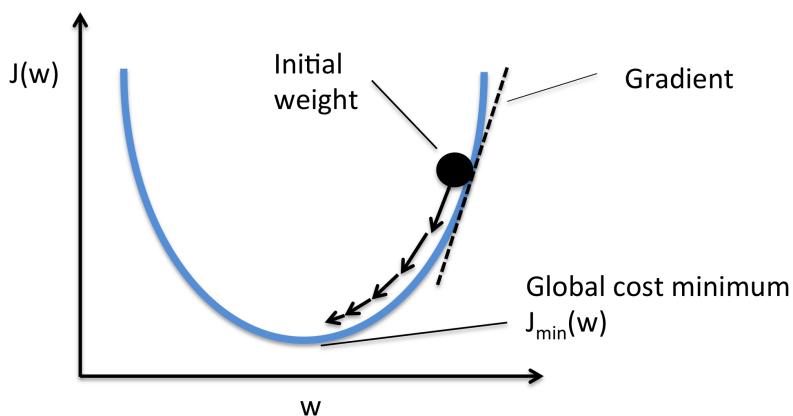
$$f(x) = \frac{e^x}{\sum_{i=0}^n e^{x_i}}$$

donde  $i$  es la cantidad total de posibles salidas del modelo. La ventaja de esta función es que agrega al modelo el conocimiento de que la suma de las probabilidades de todas las etiquetas es igual a 1.

### 2.2.3. Optimizadores

Los algoritmos de optimización minimizan (o maximizan) una función objetivo. En el caso de deep learning, esta función se corresponde con la de pérdida y se busca minimizar [21]. La función de un optimizador se basa en mejorar los valores de los parámetros entrenables para reducir el error cometido por la red. Para ello se modifican los parámetros (pesos), restando el valor de cada uno por el gradiente de un conjunto de datos (batch) multiplicado por un valor conocido como tasa de aprendizaje (learning rate).

Podemos observar en la figura 2.7 como el optimizador comienza realizando “pasos” grandes al principio, ya que se encuentra lejos de los valores óptimos y mientras más se acerca comienza a realizar movimientos cada vez más pequeños.



**Figura 2.7:** representación de aprendizaje de un modelo. La distancia y el movimiento realizado podrá variar según el optimizador utilizado.

A continuación se describen los optimizadores más utilizados.

- **Descenso por el gradiente estocástico (SGD)**

El algoritmo más utilizado y simple de optimización es el descenso por el gradiente estocástico, también conocido como SGD (Stochastic Gradient Descent) [21]. Esta técnica utiliza el mismo valor de tasa de aprendizaje (learning rate) para todos los parámetros en cada iteración. Esto

no significa que los pesos son modificados por el mismo valor en cada iteración, ya que mientras más se acerca el modelo al valor óptimo, menor será el gradiente, por lo que las modificaciones serán cada vez menores. La fórmula para calcular estos valores es:

$$w = w - \alpha * \Delta w$$

Donde  $w$  son los pesos del modelo,  $\alpha$  es la tasa de aprendizaje escogida y  $\Delta w$  el gradiente de los pesos.

#### - **Estimación adaptativa de momentos (Adam)**

La estimación adaptativa de momentos (adaptive moment estimator en inglés) es un método en el cual la tasa de aprendizaje se adapta a cada parámetro y los gradientes calculados anteriormente se utilizan para acelerar el entrenamiento de la red [22]. Se explicará el funcionamiento de este optimizador sin entrar en detalles en su matemática, ya que es compleja y se perdería el objetivo de este apartado.

Una tasa de aprendizaje que se adapta a cada parámetro hace referencia a que realiza actualizaciones más grandes (mayor tasa de aprendizaje) para valores muy dispersos y actualizaciones más pequeñas (menos tasa de aprendizaje) para valores menos dispersos.

A diferencia del optimizador SGD, que modifica su dirección según se indique por el gradiente, Adam toma en cuenta los valores de gradiente obtenidos anteriormente y, en caso de obtener un valor de gradiente diferente, lo tomará en cuenta y esto modificará su “velocidad”.

Una analogía sencilla para comprender el funcionamiento de este algoritmo es pensar en una bola bajando por el costado de una parábola, como la de la figura 2.7. La bola en cuestión es “pesada”, por lo que no modifica su dirección fácilmente. Si la bola comienza en el mismo punto que en el de la imagen, ésta continuará su recorrido hasta pasar el vértice. Luego, modificará su dirección para dirigirse nuevamente en la dirección del mínimo, aunque esta vez con menor velocidad. Esto se repetirá hasta que la bola logra colocarse en vértice.

En resumen, la idea de este algoritmo es modificar la “velocidad” de la bola según los nuevos gradientes. Si la bola se dirige con gran velocidad hacia un lado (muchos gradientes que le indican una dirección), aunque se le indique que se mueva en otra dirección, solo verá modificada su velocidad.

#### **2.2.4. Embeddings**

A partir de 2013, Tomas Mikolov et al. iniciaron una línea de investigación enfocada a la representación del texto [23, 24, 25]. Word embedding es el nombre de un conjunto de lenguajes de modelado y técnicas de aprendizaje en procesamiento del lenguaje natural (PLN) en donde las palabras o frases del vocabulario son vinculadas a vectores de números reales

[26]. Conceptualmente implica el encaje matemático de un espacio con una dimensión por palabra a un espacio vectorial continuo con menos dimensiones.

La idea detrás de word embedding es poder representar cada palabra (o valor categórico) como un vector de valores, los cuales representan las características de dicha palabra. Estas características pueden no estar dadas por el usuario, sino que pueden aprenderse utilizando, por ejemplo, modelos de Deep Learning.

En esta arquitecturas, cada palabra o valor se representa mediante un ID único, donde cada ID a su vez puede ser traducido en un vector de todos ceros, menos un uno en la posición de ID. Supongamos un vocabulario restringido  $VOC=\{P1,P2,P3,P4\}$ , entonces a la palabra P2 se le asigna el ID 2 y el vector  $V(P2)=(0,1,0,0)$ . Esta forma de codificación, se conoce como one-hot-encoding. El embeddings de las palabras es la matriz de pesos  $W$ , donde cada fila corresponde a una palabra y cada columna a una característica. Es decir, el embedding de la palabra P2 sería  $W[2,:]$ . Es importante notar que:

$$V(P2) \cdot W = (0, 1, 0, 0) \cdot \begin{bmatrix} W[1, 1] & \dots & W[1, n] \\ W[2, 1] & \dots & W[2, n] \\ W[3, 1] & \dots & W[3, n] \\ W[4, 1] & \dots & W[4, n] \end{bmatrix} = w[2,:]$$

### 2.2.5. Funciones de pérdida

Las funciones de pérdida (también conocidas como funciones objetivo y denotadas  $L$ ), al igual que las métricas, son cálculos matemáticos que permiten medir la performance de un modelo. El objetivo de entrenar un modelo es minimizar este valor, para lo cual se buscan los parámetros que resulten en predicciones cada vez más cercanas al valor real, es decir:

$$\underset{\theta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n L(y_i, H(x_i, \theta)) \right\}$$

Al ser una función que debe ser minimizada, las funciones de pérdida que se deberán elegir deben ser funciones las cuales deban ser minimizadas para encontrar el valor que hace que el modelo prediga los resultados de forma óptima. Según el tipo de problema a tratar, ya sea de clasificación o de regresión, existen diferentes funciones de pérdida que se podrán utilizar.

En el caso de los problemas de regresión, las funciones que se suelen utilizar son las descritas en el apartado de métricas, exceptuando  $R^2$ , ya que esta función indica mejores resultados mientras más grande es su valor (problema de maximización). En lo relacionado a RMSE, se obtiene siempre un resultado menor o igual que utilizando MSE, aunque con un costo computacional mayor, ya que se tiene que aplicar la raíz del resultado. Por este motivo se suele

utilizar MSE como función de pérdida en su lugar, disminuyendo la complejidad temporal y acelerando el tiempo de entrenamiento.

En el caso de los problemas de clasificación, las funciones de pérdida utilizadas no suelen ser iguales a sus métricas, ya que estas últimas buscan ser maximizadas. La función de pérdida más utilizada en problemas de clasificación es la entropía cruzada categórica (categorical cross-entropy en inglés), la cual mide la información mutua entre el resultado del clasificador y el valor verdadero. Esta función crece mientras más divergen las predicciones de los valores reales y decrece mientras más se asemejan las predicciones a los valores reales. La fórmula para calcularla en el caso de clasificación binaria es:

$$CrossEntropyLoss = - (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Se puede observar que cuando el valor real de resultado es 1 ( $y_i = 1$ ), la segunda mitad de la función se hace 0 y, cuando el valor real de resultado es 0 ( $y_i = 0$ ), la primer mitad se hace 0.

En resumen, se aplica el logaritmo de la probabilidad de que la clase predicha sea la clase real. Un aspecto a tener en cuenta es que esta función penaliza en gran medida las predicciones con alto valor de probabilidad pero erradas.

Para problemas de multiclase se realiza el mismo cálculo, calculando el valor correspondiente por cada clase, donde tomaría el valor 0 con todas las otras clases, y luego realizando el promedio de dichos resultados.

## 2.2.6. Batch

Un batch es un conjunto de datos agrupados que se utilizan para entrenar un modelo. El tamaño de batch define la cantidad de datos que van a ser propagados por la red.

Por ejemplo, si se tienen 1050 datos de entrenamiento y un tamaño de batch (batch\_size) de 100, se tomarán los primeros 100 datos del conjunto de entrenamiento y se entrenará a la red, realizando la etapa de forward, calculando los gradientes y modificando los pesos. Luego se tomarán otros 100 datos diferentes del mismo conjunto de entrenamiento y se entrenará nuevamente a la red. Cuando queden los últimos 50 datos, se los utilizarán para entrenar a la red, ignorando el tamaño de batch y utilizando un menor número de datos.

Según el tamaño de batch elegido, la etapa de aprendizaje de parámetros se definirá como batch gradient descent, stochastic gradient descent o mini-batch gradient descent. A continuación se plantean las ventajas y desventajas de cada uno.

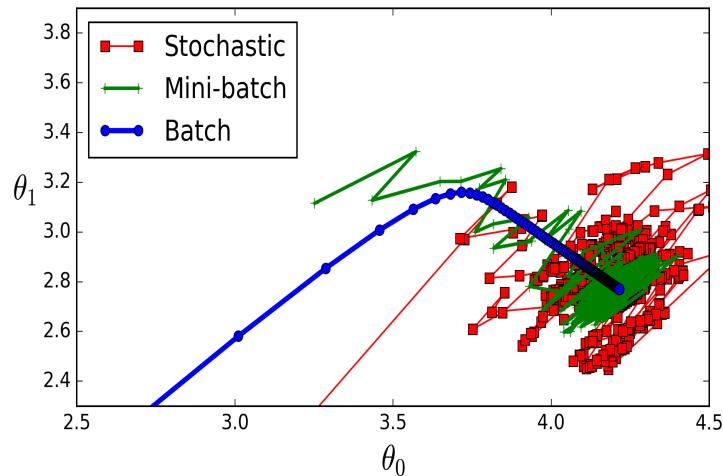
En el batch gradient descent [28] (tamaño del batch = tamaño del conjunto de entrenamiento), se computa el gradiente sobre todo el dataset, por lo que la trayectoria que se toma para llegar al mínimo global está dirigida por todos los datos. Realizar este tipo de entrenamiento lleva mucha memoria, ya que se deberá almacenar todo el batch en cuestión. Normalmente no se

suele usar esto ya que no se poseen los recursos para alojar todo el dataset en memoria, aunque es recomendable.

En el stochastic gradient descent (descenso por el gradiente estocástico, tamaño de batch = 1), al contrario, se modifican los parámetros por cada instancia del conjunto de entrenamiento. Al utilizar un único dato, la muestra posee mucho ruido y puede alejarse de los óptimos. La principal desventaja es su ineficiencia (ya que no se beneficia de la vectorización) y requerirá mucho tiempo encontrar una buena solución.

Por último, en mini-batch gradient descent [28] ( $1 <$  tamaño de batch  $<$  tamaño del conjunto de entrenamiento) se computa el gradiente sobre un subconjunto de datos de entrenamiento, por lo que se posee ruido, evitando caer en malos óptimos locales, aunque no es suficiente para evitar la rápida convergencia de la red. Al utilizar un subconjunto de datos, se requerirá menos memoria que en batch gradient descent, ya que se almacenará sólo el batch a utilizar. La principal desventaja que se tiene es que, mientras más pequeño sea el tamaño de batch, menos preciso es el gradiente estimado. Se recomienda utilizar números potencia de 2 por cómo funciona el hardware existente y, además, utilizar el tamaño máximo que soporte la GPU, ya que esto beneficiará enormemente el uso de recursos y la velocidad del modelo.

En la figura 2.8 se observa una comparación entre estos tipos de batch.



**Figura 2.8:** Comparación entre SGD, M-B GD, BGD buscando llegar al óptimo.

Se puede observar que si el tamaño de batch es grande asegura llegar al óptimo global, aunque de manera lenta, ya que calcular el gradiente sobre todo el conjunto de datos lleva mucho tiempo. En cambio, utilizar tamaños de batch reducidos demuestran llegar a una buena solución rápidamente. Esto se puede explicar intuitivamente partiendo de que con tamaño de batch reducido se permite al modelo aprender antes de ver todo el conjunto de datos de entrenamiento. La desventaja de usar tamaños pequeños en un modelo es que no garantizan obtener el mínimo global.

Hay que tener en cuenta que lo que se busca es minimizar la función de pérdida, por lo que un batch más grande asegura descender por el gradiente verdadero en lugar de estimarlo. El problema es que favorece el overfitting y perjudica la generalización del modelo sobre casos de prueba y reales. Por ende, hay que encontrar un tamaño de batch que aproveche la velocidad de cómputo de la GPU y que permita al modelo generalizar. Este tamaño se encuentra a prueba y error. No hay una fórmula predefinida que pueda ayudar a definir este valor.

## 2.3 Modelos de machine learning

### 2.3.1. Árboles de decisión

Este tipo de modelo usa un árbol de decisiones como modelo predictivo. Además de ser sencillo de implementar, posee la ventaja de que puede capturar relaciones no lineales entre los datos.

Si la variable de destino puede tomar un conjunto finito de valores se denominan árboles de clasificación. En estas estructuras de árbol, las hojas representan etiquetas de clase y las ramas representan las conjunciones de características que conducen a esas etiquetas de clase. En cambio, si la variable de destino puede tomar valores continuos, se los llama árboles de regresión [29]. Este tipo de modelos no soportan variables categóricas como valores de entrada, por lo que únicamente pueden ser utilizados con variables numéricas.

El objetivo es lograr crear el árbol óptimo a partir de los datos, el cual es construido de arriba hacia abajo (del nodo raíz a los nodos hoja). Para realizar esto, se parte de un nodo raíz y se selecciona el atributo más útil para crear los nuevos nodos hijo. Esto se realiza por cada nodo nuevo creado hasta llegar a la profundidad máxima o hasta que se llegue a una cantidad mínima de muestras (datos) en cada hoja.

Para seleccionar el atributo más útil para crear los nuevos nodos se utilizan diversas funciones, siendo las más conocidas y utilizadas la ganancia de información y la impureza de gini para árboles de clasificación [29] y, para los árboles de regresión, las funciones más utilizadas son MSE y MAE, explicadas en el apartado de métricas.

La impureza de Gini dice que si seleccionamos dos elementos al azar de una población (elementos que se poseen en un nodo) y son de la misma clase entonces la impureza del nodo será 0, es decir, es un nodo puro (todos los elementos pertenecen a la misma clase). La fórmula para calcular el grado de impureza de un nodo es:

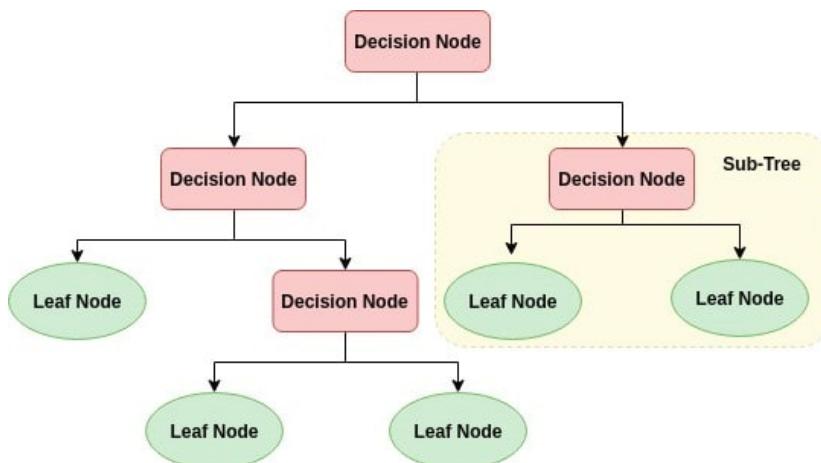
$$I_g(f) = 1 - \sum_{i=1}^m f_i^2$$

Donde  $m$  es la cantidad total de clases en el nodo y  $f_i$  la fracción de elementos etiquetados con valor  $i$  en el conjunto.

La ganancia de información, en cambio, se basa en el concepto de la entropía para seleccionar los atributos más útiles para crear los nuevos nodos, midiendo qué tan bien un atributo separa los datos de entrenamiento según su clase. La fórmula para calcular esta función es:

$$I_E(f) = - \sum_{i=1}^m f_i \log_2 f_i$$

Donde  $m$  es la cantidad total de clases en el nodo y  $f_i$  la fracción de elementos etiquetados con valor  $i$  en el conjunto.



**Figura 2.9:** ejemplo de árbol de decisión, constituido por nodos hoja (verde), nodos de decisión (rojo) y sub-árboles (amarillo).

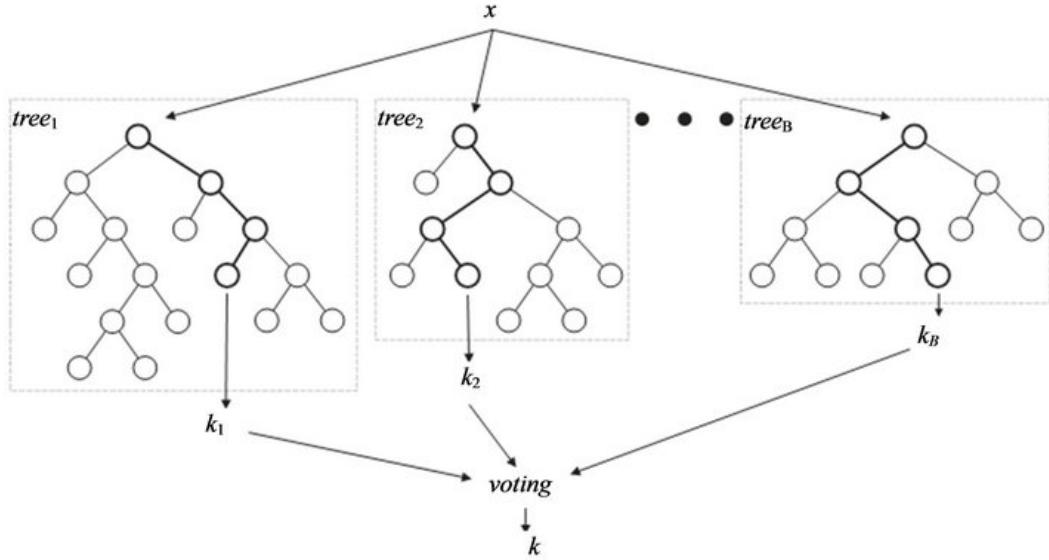
### 2.3.2. Random forest (Bosques aleatorios)

Esta técnica, tal cual describe su nombre, es un conjunto de árboles de decisión entrenados con diferentes conjuntos de entrenamiento. Cada árbol interno funciona como los descritos anteriormente. Para entrenar cada árbol se utiliza una técnica conocida como bagging [29], la cual busca utilizar un subconjunto de datos para entrenar cada uno, pudiendo encontrarse el mismo dato en varios árboles. Esto reduce la varianza en el modelo. Según sea un modelo de clasificación o regresión, la clase o valor resultante se calcula de diferente manera.

En el caso de ser un modelo de clasificación se elige la clase con mayor cantidad de votos entre los árboles creados, es decir, la clase que más veces haya sido predicha, mientras que en un modelo de regresión se calcula el promedio de los valores predichos.

Se puede observar en la figura 2.10 un modelo random forest para clasificación con  $n$  árboles. Al modelo se le ingresan los valores de entrenamiento, junto con sus valores de salida

correspondientes. Luego, el algoritmo dividirá estos datos de entrenamiento y entrenará un árbol por cada conjunto de dato creado. Una vez creados los árboles, el modelo ya se encuentra entrenado, por lo que podremos ingresar valores de prueba para conocer la precisión de nuestro dataset. Al ingresar los valores, estos son evaluados por cada árbol y, luego de que todos los árboles hayan terminado de evaluar, se elige la clase que haya sido predicha la mayor cantidad de veces. Esto es similar para los modelos de regresión, ya que en vez de votar se calcula el promedio de los valores continuos predichos.



**Figura 2.10:** ejemplo de modelo random forest, constituido diversos árboles de decisión, los cuales se construyen con un subconjunto del total de datos de entrenamiento.

### 2.3.3. Regresión lineal y regresión logística

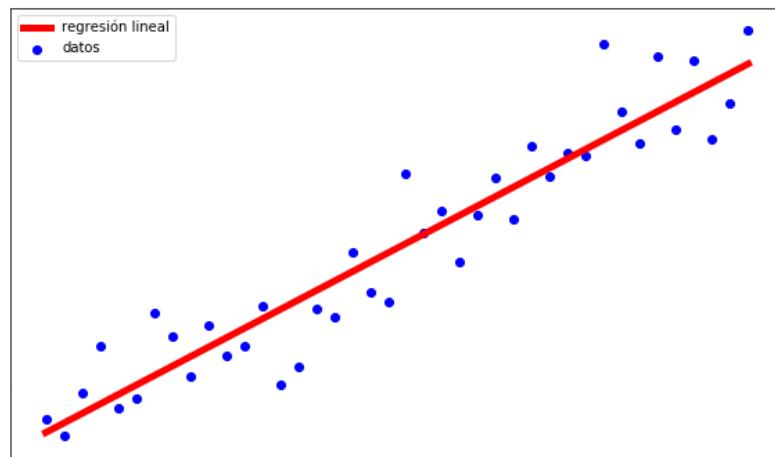
Un modelo de regresión lineal, al igual que una regresión logística, son modelos conformados únicamente por un perceptrón, el cual poseerá tantos valores de entrada como sea necesario. El objetivo de este tipo de modelo es encontrar la función lineal que minimice el error obtenido por la red. Este error está dado por alguna de las funciones vistas en el apartado de Funciones de pérdida.

El modelo recibe los datos de entrada, los cuáles deben ser únicamente numéricas [30], realiza los cálculos necesarios y retorna uno o varios valores (según el tamaño de batch). Luego, se calcula la función de pérdida con los datos predichos y los valores reales y se aplica el optimizador deseado. Esto se realiza la cantidad de veces deseadas (epochs de un modelo). De este modo, los parámetros del modelo (los pesos) son actualizados, ajustando la función lineal resultante y provocando que el error disminuya.

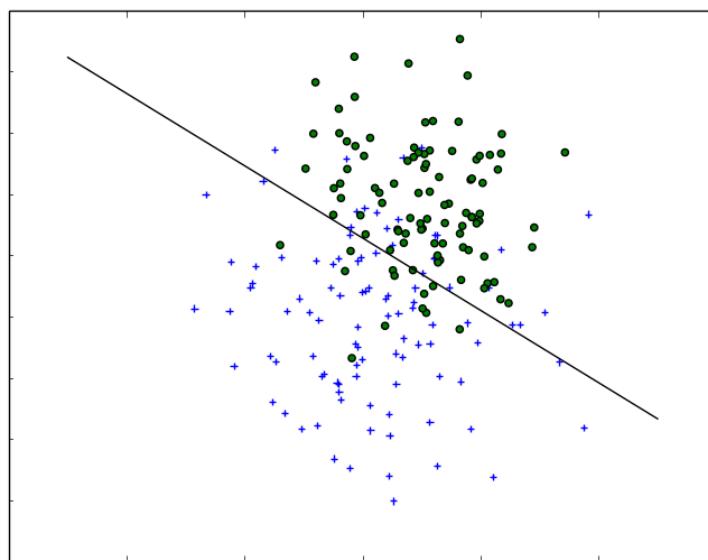
Hay que tener en cuenta que mientras más se ajuste este modelo y más se busque el menor error, hay posibilidad de que se pierda capacidad de generalización, por lo que las predicciones sobre un conjunto de diferente al de entrenamiento generen un error elevado [31]. A este

fenómeno se lo conoce como overfitting. Su contraparte es el underfitting, en la cual un modelo que no se adapta correctamente al conjunto de entrenamiento, por lo que tampoco dará buenos resultados en el conjunto de prueba.

La diferencia entre regresión lineal y regresión logística es que la primera brindará valores continuos (la capa de salida del modelo utiliza una función de activación lineal), siendo útil para cuando se debe resolver un problema de regresión. En este caso se buscará la función lineal que mejor se adapte a los valores (figura 2.11). En cambio, los modelos de regresión logística buscarán una función lineal que separe las clases que las comprenden (figura 2.12). Aquí, en lugar de obtener valores continuos, obtendremos la probabilidad de que sea de una clase designada (la capa de salida utiliza una función de activación sigmoide, en caso de tener dos clases, o softmax, en caso de tener más de dos clases).



**Figura 2.11:** ejemplo de modelo de regresión lineal. En rojo se encuentra la función lineal creada por el modelo. En azul los datos a predecir.



**Figura 2.12:** ejemplo de modelo de regresión logística. En negro se encuentra la función lineal creada por el modelo, la cual busca separar las dos clases (azul y verde).

#### 2.3.4. Aprendizaje profundo (Deep Learning)

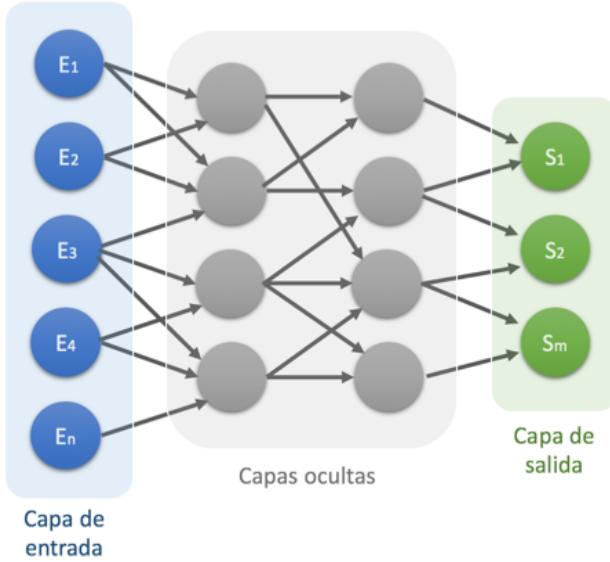
Aprendizaje profundo (deep learning en inglés) es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos. El aprendizaje profundo es una subrama del aprendizaje automático, la cual se basa en diversas capas de neuronas, las cuales utilizan la salida de las capas anteriores como entrada. Estos modelos aprenden en múltiples niveles. Las características de más alto nivel (detalles en los datos) se derivan de las características de más bajo nivel (información general de los datos).

##### 2.3.4.1. Red neuronal profunda

Las redes neuronales profundas son un tipo de modelos de aprendizaje profundo. Estos se caracterizan por estar constituidos por una entrada, una “caja negra” y una salida. Esta caja negra está constituida por capas, donde cada capa es un conjunto de neuronas tal cual se las vio anteriormente, por lo tanto, a cada neurona le llega una o más entradas, se realiza el cómputo necesario y al resultado se le aplica una función de activación designada [32], la cual funciona como entrada a la capa siguiente (o como salida de la red en caso de ser la última). Es importante tener en cuenta que la última capa debe poseer una función de activación según el problema que se busque resolver. Las más comunes son las mismas que se utilizan en ambos tipos de regresión, lineal (lineal) y logística (sigmoide o softmax).

Se ve una clara similitud entre los modelos de regresión lineal y logística y los de redes neuronales [32]. Esto se debe a que estos últimos poseen una estructura similar, con el agregado de nuevas capas de neuronas entre las de entrada y salida.

Este tipo de modelos, al poseer mayor cantidad de neuronas, puede aprender patrones más complejos que un simple modelo de regresión lineal (o logística), aunque puede ser posible que demore aún más tiempo de entrenamiento, ya que la cantidad de cálculos y modificaciones en los pesos es mayor mientras más cantidad de nodos (o neuronas) haya.

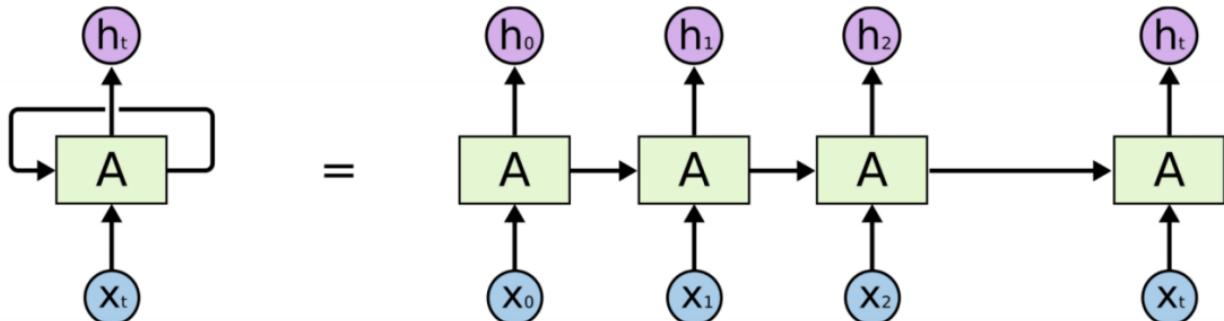


**Figura 2.13:** ejemplo de red neuronal profunda. En azul se encuentran los features de entrada, luego las capas ocultas y por último la capa de salida.

#### 2.3.4.2. Red neuronal recurrente (RNN)

Hasta el momento, todos los modelos son entrenados con instancias independientes una de la otra, es decir, ninguna característica influye directamente sobre otra. En cambio, las redes neuronales recurrentes permiten almacenar información del “pasado”, permitiendo que la red posea memoria de los sucesos.

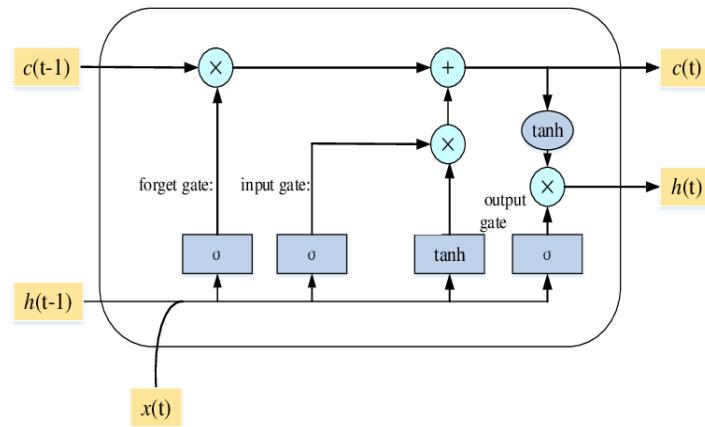
Este tipo de redes posee unidades cuya salida vuelve a introducirse en la celda, formando ciclos (figura 2.14). Dado que la salida de una neurona recurrente en un instante de tiempo determinado es una función de entradas de los instantes de tiempo anteriores, se podría decir que una neurona recurrente tiene en cierta forma memoria.



**Figura 2.14.** Ejemplo de celda recurrente, en donde la activación de una instancia tiene tantos ciclos como features tenga, lo cual provoca que el entrenamiento de las mismas sea sumamente costoso.

Para ilustrar el concepto de “memoria” de una RNN, imaginemos que tenemos un modelo como los vistos anteriormente, al cual le pasamos la palabra “neurona” como entrada y esta red procesa la palabra carácter a carácter. En el momento en que alcanza el carácter “r”, ya se ha olvidado de “n”, “e” y “u”, lo que hace que sea casi imposible para la red neuronal predecir qué letra vendrá después. Pero en cambio, una RNN permite recordar precisamente esto.

Un tipo especial de celda recurrente, creada por Hochreiter et. al., son las Long-Short Term Memory (LSTM) [33] (figura 2.15), las cuales básicamente amplían su memoria para aprender de experiencias importantes que han pasado hace mucho tiempo. Las LSTM permiten a las RNN recordar sus entradas durante un largo período de tiempo. Esta memoria se puede ver como una “celda” bloqueada, donde “bloqueada” significa que la célula decide si almacenar o eliminar información dentro (abriendo la puerta o no para almacenar), en función de la importancia que asigna a la información que está recibiendo.



**Figura 2.15.** Celda LSTM. Se tienen dos salidas en este caso, la activación correspondiente y el valor de memoria, ingresando ambos al próximo ciclo.

## **Capítulo 3: Trabajos relacionados**

Existen numerosas aplicaciones que buscan solucionar el problema de estacionamiento en espacios urbanos. Las mismas ofrecen diferentes enfoques tanto del problema como de su solución.

Un gran porcentaje de las propuestas se basan en la instalación de sensores en los espacios de estacionamiento que se buscan medir. Estos sensores representan un alto costo, ya que una vez instalados, se deberá proveer el correspondiente mantenimiento. Además, estas soluciones pueden ser implementadas únicamente en estacionamientos privados, en donde las plazas para estacionar ya se encuentran divididas por vehículo.

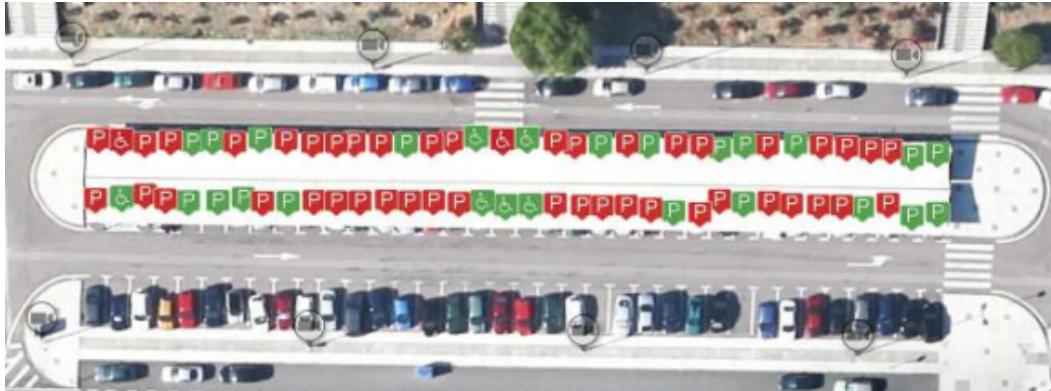
Como opción alternativa para reducir el costo inicial de estas aplicaciones, se utilizan los sensores propios de los dispositivos móviles de los usuarios. En este contexto se busca conocer la disponibilidad de lugares en base a la información proporcionada por el dispositivo móvil de cada usuario. Si se posee un limitado número de usuarios, la capacidad de estas aplicaciones para brindar información correcta se ve reducida. Esta característica se ve reflejada en un enfoque distinto, en el cual se busca que el usuario cargue la información correspondiente sobre la ocupación de un lugar.

A continuación se detallan las soluciones propuestas para resolver el problema de encontrar lugares de estacionamiento libre y los enfoques implementados en Argentina.

### **3.1. Sensores especializados**

Una solución para encontrar lugar de estacionamiento libre se basa en utilizar sensores especializados [3]. De esta forma, la información brindada al usuario sobre la disponibilidad de los lugares libres será correcta y precisa. Además, este tipo de sistemas es escalable, ya que cada sensor ofrece información de manera independiente, por lo que agregar nuevos sensores únicamente amplía la información disponible de los lugares libres.

Este enfoque es utilizado, principalmente, en las playas de estacionamiento, las cuales poseen sus lugares para estacionar delimitados. De este modo, se podrán instalar los sensores correspondientes en cada espacio. En la figura 3.1 se puede observar la respuesta brindada por el sistema [3]. Por este medio, puede conocerse la precisión con la que brinda información este tipo de enfoque, en el que podemos conocer con exactitud la disponibilidad de cada lugar para estacionar.



**Figura 3.1:** Ocupación en tiempo para cada plaza de estacionamiento utilizando la implementación propuesta [3].

La principal desventaja de esta propuesta es el costo de instalación y mantenimiento de los sensores instalados. A medida que incrementa el número de sensores a instalar, el costo disminuye. A pesar de esto el costo de mantenimiento crece exponencialmente [4]. Por esto, implementar esta solución en ciudades pequeñas o medianas representa una limitante, al igual que implementarlo en todos los lugares disponibles en ciudades grandes.

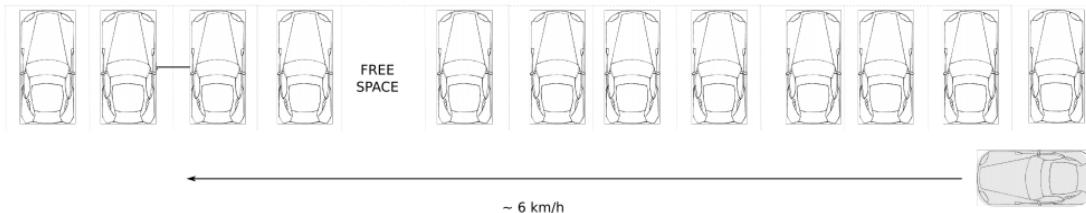
Una solución planteada con un enfoque similar fue implementada en la ciudad de Buenos Aires, en la cual se utiliza tanto sensores especializados como cámaras de las playas de estacionamiento para obtener lugares libres (CommuThings SmartParking) [43]. En esta propuesta, se busca solucionar la limitante de la cantidad de sensores necesarios utilizando cámaras en los lugares que las posean, siendo una solución ligeramente menos costosa, aunque igualmente representa un gran costo para ciudades pequeñas y medianas.

### 3.2. Sensores móviles

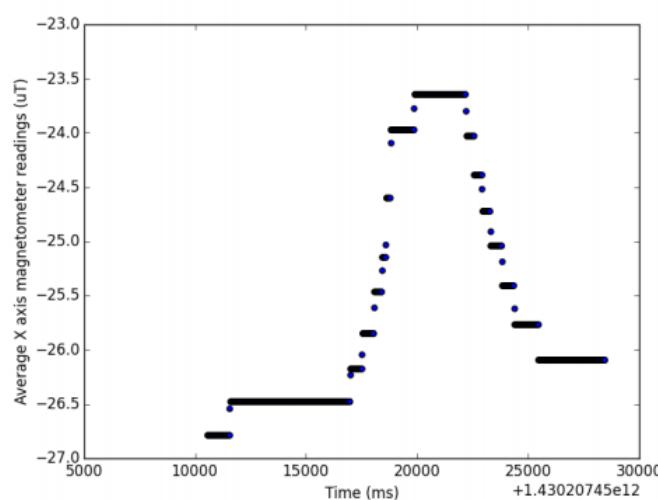
En un enfoque más económico se utilizan los sensores de los dispositivos móviles. El objetivo de utilizar estos sensores, aunque sean menos precisos que unos especializados, es reducir el costo de implementación de la solución, ya que solo basta con tener usuarios que brinden información al sistema para poder ofrecer información. Por este medio, se busca conocer los lugares libres y ocupados utilizando únicamente la información brindada por los usuarios del sistema, por lo que es posible utilizar esta aplicación tanto en playas de estacionamiento como en lugares de estacionamiento en espacios urbanos. La principal desventaja que posee este enfoque es la cantidad de usuarios que utilizan la aplicación y, además, no todos poseen dispositivos móviles con gran cantidad de sensores. Al ser los usuarios la única fuente de información, en caso de poseer pocos no se podrán ofrecer soluciones certeras y veraces al usuario, ya que estas no se conocen. En caso de poseer gran volumen de usuarios, se podrá conocer la ocupación real la mayoría del tiempo, resultando en buenos resultados brindados de un modo económico de implementar.

Para ser más precisos, se ejemplifican dos enfoques, en los cuales utilizan diferentes sensores para brindar resultados.

En una de las soluciones se utiliza el magnetómetro de los dispositivos móviles de los usuarios [5]. En este caso se busca que los usuarios colaboren con la búsqueda de lugares libres de estacionamiento, ya que la información recogida es utilizada para conocer si existe lugares libres adyacentes al vehículo del usuario. De esta manera, se busca armar un mapa no solo de los lugares ocupados y liberados por los usuarios de la aplicación, sino también con la información que estos brindan de su alrededor, por lo que la cantidad de usuarios necesarios para tener una cantidad de datos suficientes se reduce. Este enfoque, al obtener continuamente datos del magnetómetro del dispositivo, puede llegar a consumir una gran parte de la batería del dispositivo móvil en caso de no cerrarla correctamente, ya que todo el tiempo se envía información a los servidores de la aplicación. En la figura 3.2 se puede observar un escenario en donde un vehículo en movimiento, el cual posee en su interior un dispositivo móvil con la aplicación corriendo, recolecta datos continuamente. Este escenario se planteó como prueba, demostrando que los resultados obtenidos por el magnetómetro brindan información acerca de la disponibilidad de los lugares de estacionamiento (figura 3.4).

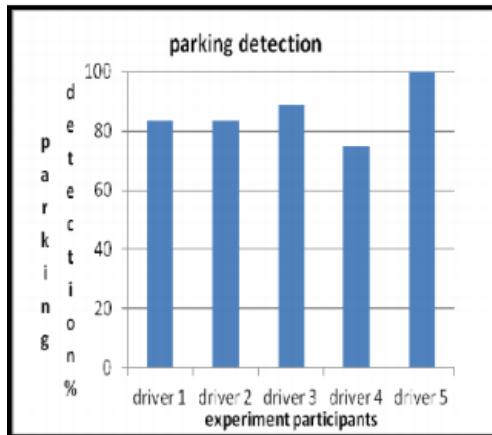


**Figura 3.2:** Escenario de prueba utilizando un vehículo en movimiento [5].

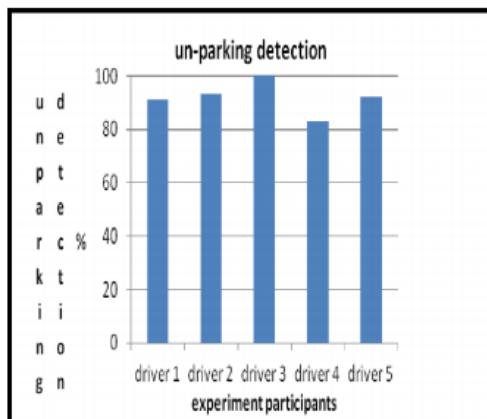


**Figura 3.4:** eje x del magnetómetro utilizado en un caso de prueba [5].

Mientras que el caso anterior utiliza únicamente el magnetómetro para encontrar solución al problema, PhonePark [7][8] busca utilizar sensores más comunes en los dispositivos móviles más antiguos (por lo que más gente los tendría disponibles), tales como el acelerómetro y el GPS, a la vez que puede utilizar el Bluetooth en caso de estar vinculado al vehículo. En este enfoque se busca encontrar el momento en el que el usuario se encuentra en movimiento, frena y comienza a caminar. Una vez que se detecta esta situación, el sistema guardará como lugar ocupado las coordenadas de dónde frenó el vehículo. Este sistema puede tener inconvenientes en caso de que la detección de movimiento, frenar y caminar no sea correcta (figura 3.5, figura 3.6). Al utilizar gran cantidad de datos y crear nuevos en cada etapa, cada uno de estos pasos puede sufrir y arrastrar errores, resultando en datos inválidos. Como se puede observar en la figura 3.5 y figura 3.6, esto sucede pocas veces, obteniendo datos correctos en un promedio del 85% de los casos, por lo que este enfoque puede brindar buenos resultados si se posee una gran cantidad de usuarios registrados.



**Figura 3.5:** precisión (accuracy) obtenida en la detección de estacionamiento de un usuario [7].



**Figura 3.6:** precisión (accuracy) obtenida en la detección de liberación de espacio de estacionamiento de un usuario [7].

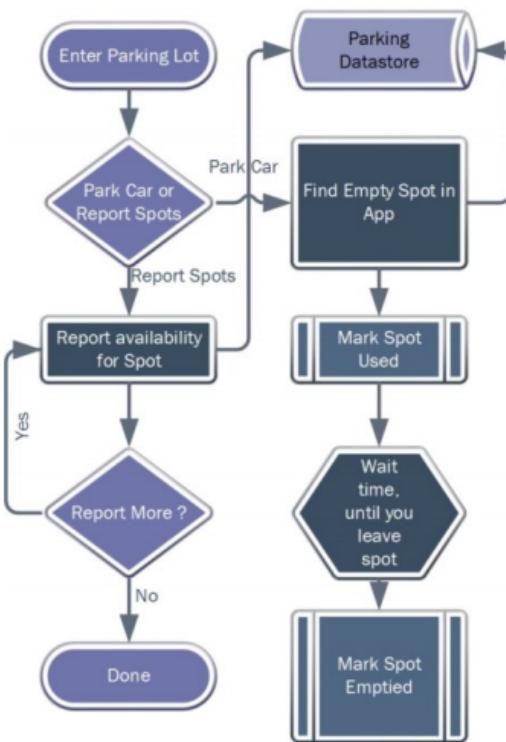
Este enfoque, al no requerir disponibilidad de ningún dispositivo físico salvo los dispositivos móviles de los usuarios, puede ser utilizado en cualquier parte del mundo, por lo que no se

implementa este tipo de solución únicamente para argentina. Las aplicaciones explicadas anteriormente pueden utilizarse en el país sin ningún tipo de restricciones, aunque hay que tomar en cuenta la cantidad de usuarios que posee la aplicación dentro del mismo, ya que en eso se basan los resultados que podrá brindarnos a la hora de consultar por disponibilidad de lugares de estacionamiento.

### **3.3. Carga de datos manual**

Otras formas de resolver el problema de encontrar lugar para estacionar se basan en métodos más simples, como ParkJam, en los cuales los usuarios son los encargados de brindar información sobre los lugares de estacionamiento [6][9]. La ventaja de este tipo de soluciones es que los datos son, en caso de que no haya usuarios intentando dañar al sistema, siempre correctos y exactos, por lo que se podrá saber en tiempo real la ocupación del lugar. La desventaja, además de que gran cantidad de usuarios deben utilizar la aplicación para conocer el estado de cada lugar para estacionar, es el incentivo que se les da a los usuarios para participar activamente de la aplicación. El beneficio que recibe el usuario por utilizar la aplicación y brindar información es conocer los lugares libres para estacionar, lo que no siempre lleva al usuario a querer realizar tales acciones, ya que se suele buscar una solución que realice esta tarea de forma automática. En caso de que la aplicación no llegue a tal volumen de gente, es difícil conseguir que los usuarios se mantengan activos en el uso de la misma, ya que el usuario brindaría información de los lugares libres pero no recibiría la misma de forma completa y correcta.

En la figura 3.7 puede observarse el flujo de información y de funcionalidad de la aplicación [9]. Se verifica que las operaciones realizadas por la aplicación son sencillas, ya que le brinda al usuario los lugares libres disponibles y este, una vez estacionado, marca el lugar como ocupado hasta que lo libera, en donde selecciona que el lugar se desocupó.



**Figura 3.7:** gráfico descriptivo del flujo de operaciones [9].

Al igual que en las propuestas que utilizan sensores de dispositivos móviles, no se implementó una solución de este tipo en Argentina, ya que pueden utilizarse en cualquier parte del mundo sin inconvenientes. Similar a lo dicho anteriormente, hay que tomar en cuenta la cantidad de usuarios que utilizan estas aplicaciones, ya que esto determina los resultados brindados a la hora de buscar lugar libre para estacionar.

### 3.4. Conclusión

Como pudo observarse en las secciones anteriores, la mayoría de los enfoques poseen dos problemas principales: el costo de implementación y la cantidad de usuarios dispuestos a usar la aplicación para brindar datos. En este tipo de problemas una buena solución busca reducir al mínimo la cantidad de usuarios necesarios para funcionar, además de tener un costo de implementación mínimo, ya que esto resulta privativo en el caso de ciudades pequeñas o medianas y, en ciudad grandes, en caso de querer instalar la solución en toda la ciudad.

Para el caso de la ciudad de Tandil, la cual se encuentra en constante crecimiento, la solución que incluye la instalación de sensores no es redituable, ya que no hay gran cantidad de plazas de estacionamiento en la ciudad y, las que se encuentran disponibles, no suelen llegar a su capacidad máxima de vehículos estacionados. En cambio, las soluciones que utilizan sensores de dispositivos móviles tiene como principal desventaja la cantidad de usuarios que utilizan la aplicación, como se vió anteriormente.

Por estos motivos, se buscó otra manera de solucionar el problema de encontrar estacionamiento libre, en la cual se utilizan los parquímetros ya instalados, que los usuarios utilizan una vez que estacionan su vehículo, de la ciudad de Tandil. Este enfoque tiene como principal ventaja la disponibilidad de datos, ya que se obtienen cada vez que un usuario marca como ingresado o egresado su vehículo en un parquímetro. Además, no se depende de los usuarios para mejorar las predicciones, por lo que brindará los mismos resultados al tener cualquier cantidad de usuarios. Por otro lado, el costo de implementarlo es prácticamente nulo, ya que los parquímetros fueron instalados con el principal objetivo de funcionar como lugar de cobro por estacionamiento medido, por los que esto no representa un gasto adicional.

## **Capítulo 4: Dataset**

En este trabajo se realizaron diversos análisis para obtener patrones de comportamiento. Se construyeron tres datasets, ya que se vió necesario obtener mayor volumen de datos y más actualizados. Aún así, se realiza el análisis sobre los tres para comprobar las diferencias y similitudes en los patrones de estacionamiento por cada conjunto. Los tres datasets contienen datos del sistema SUMO en diferentes períodos de tiempo, lo cual es útil ya que brindaran información sobre si se mantienen los mismos patrones de estacionamiento a través del tiempo o se vieron modificados por algún motivo.

Primeramente, se explicará cómo fueron conseguidos estos datasets y qué información brindan. Luego, se detalla el análisis realizado sobre los mismos y los patrones de estacionamiento observado, además de describir las anomalías encontradas y los problemas hallados durante este análisis. También se busca encontrar relaciones en base a los días de la semana, a los meses y a cada hora del día, por lo que se podrán visualizar gráficas y datos correspondientes a diversos patrones que siguen estos datos.

Para facilitar el manejo de los datos se decidió utilizar la librería pandas, la cual fue diseñada para manipular grandes volúmenes de datos de manera simple y ordenada en estructuras propias llamadas DataFrame. Además, cuenta con herramientas para leer y escribir datos en memoria, sobre las que acepta formatos de archivos variados (csv, excel, etc.) [11].

### **4.1. Obtención de datos**

#### **4.1.1. Dataset 2018**

Para la primera aproximación al análisis de datos se utilizó un dataset con datos relativamente antiguos ya que fechan desde el 01/01/2018 al 14/06/2018. Se decidió comenzar con estos datos ya que fueron los primeros en encontrarse a disposición. Este dataset consta de 704.751 datos, organizado por filas, las cuales poseen: fecha y hora en la que se efectuó la operación, un identificador de cuadra, la operación (Entrada o Salida), patente del vehículo y el número de la tarjeta SUMO utilizada. Se puede observar en la tabla 4.1 la representación de los datos utilizando un DataFrame de pandas.

<b>id_cuadra</b>	<b>operacion</b>	<b>patente</b>	<b>tarjeta</b>	<b>fecha</b>	<b>hora</b>
82	Entrada	1	10037637	2018-01-01	07:11:00
40	Entrada	2	10059452	2018-01-01	11:26:00
61	Entrada	3	10032179	2018-01-01	13:09:00
34	Entrada	4	10026609	2018-01-01	16:53:00
66	Entrada	5	10073847	2018-01-01	17:25:00

**Tabla 4.1:** Ejemplo de filas pertenecientes al dataset 2018, las cuales contienen fecha, hora, id de cuadra, operación (Entrada/Salida), patente del vehículo y tarjeta utilizada. Los valores de patente no son mostrados con el fin de preservar la privacidad de los usuarios.

La primer problemática encontrada al visualizar los datos fue la incapacidad de conocer a qué cuadra de la ciudad de Tandil hacía referencia el identificador, por lo que fue imposible corroborar estos datos con la actividad de la ciudad en un día.

Aún así, se completó la información proporcionada generando la ocupación en cada momento sobre el conjunto total de datos. Primero se corroboró que los datos estuviesen ordenados por fecha y luego se inició la ocupación de cada calle en cero y se sumaba o restaba uno en caso de que la fila poseía una entrada o una salida respectivamente (tabla 4.2). Conocer la ocupación de cada calle en cada momento es fundamental para poder predecir los lugares libres para estacionar.

Una solución a la problemática de no conocer las cuadras reales a las que se hace referencia en los datos fue obtener datos del sistema SUMO por medio de la web (se detalla en la sección 4.1.2), los cuales poseían un identificador de cuadra, la cuadra en cuestión y su localización por medio de coordenadas, entre otros atributos. Con esta información, confiando en que se utilizaron los mismos identificadores en ambos casos, se pudo relacionar las direcciones a las que hacían referencia los datos.

Al obtener estos datos web actualizados se optó por seguir trabajando con ellos para que el análisis reflejase los movimientos actuales de los habitantes de la ciudad.

<b>id_cuadra</b>	<b>operacion</b>	<b>patente</b>	<b>fecha</b>	<b>hora</b>	<b>direccion</b>	<b>latitud</b>	<b>longitud</b>	<b>ocupacion</b>
82	Entrada	6	2018-01-01	07:11:00	General Paz 647	-37.324173	-59.132848	1
82	Salida	6	2018-01-01	07:11:00	General Paz 647	-37.324173	-59.132848	0
82	Entrada	7	2018-01-02	09:06:00	General Paz 647	-37.324173	-59.132848	1
82	Entrada	8	2018-01-02	09:42:00	General Paz 647	-37.324173	-59.132848	2
82	Salida	9	2018-01-02	09:45:00	General Paz 647	-37.324173	-59.132848	1

**Tabla 4.2:** Ejemplo de filas pertenecientes al dataset 2018 una vez calculada la ocupación y añadidas dirección y coordenadas. Los valores de patente no son mostrados con el fin de preservar la privacidad de los usuarios.

#### 4.1.2. Dataset web scraping

El análisis con el dataset del 2018 tenía ciertas desventajas. La cantidad de datos era escaso y estos, a su vez, estaban desactualizados, como se verá más adelante. Por estos motivos surgieron dos posibilidades: solicitar los datos al personal del sistema SUMO de Tandil por medio de una carta o extraer datos por medio de una aplicación, la cual rastree tanto los vehículos como el momento en que estacionan.

Se optó por la primer solución, para la cual se envió una carta al municipio y al sistema SUMO solicitando los datos actualizados. Al no recibir respuesta, se consultó de manera personal sobre el pedido en las oficinas SUMO (9 de Julio 549), sobre el cual me informaron que no tenían noticias. Me facilitaron una planilla con los datos de la actividad de los parquímetros del momento.

La planilla contenía un link al sitio web (<http://gpssumo.com/parquimetros>) del sistema de los parquímetros, el cual contenía información actualizada. A partir de esto se planteó la idea de extraer la información en tiempo real del sistema, lo cual permitiría tener un conjunto de datos mayor y actualizado. Para obtenerlos se utilizaron herramientas de web scraping, las cuales permiten automatizar el proceso de extracción de grandes volúmenes de datos y almacenarlos en un formato sencillo de procesar, donde en este caso se utilizó csv. La idea de web scraping es simular la navegación de un humano en los sitios web, automatizando comportamientos como, por ejemplo, la obtención periódica de datos. Los pasos realizados se detallan a continuación.

	<b>Id</b>	<b>Dirección</b>	Zona Tarifa	Tensión Batería	Tensión Panel	Versión FM	Cantidad Aperturas-Cierres (Día)	Cantidad Reportes de Estado (Día)	Fecha Última Transacción	Hora Última Transacción	Ocupación Actual	Ocupación Máxima (Día)	Lugares Cuadra (BETA)
1	20066	14 de Julio 347	A	10.58	19.87	260	0	0	08-02-20	19:00:26	0	0	20
2	20009	Rodriguez 654	A	13.4	0.06	260	0	1	09-02-20	00:00:28	0	0	24
3	20071	14 de Julio 557	A	13.64	14.96	260	1	1	09-02-20	05:43:34	0	0	14
4	20025	Alem 658	A	13.43	13.98	260	0	17	09-02-20	16:00:06	0	0	18
5	20076	General Paz 547	A	14.54	19.78	260	2	18	09-02-20	16:00:06	0	0	17
6	20046	Mitre 348	A	12.98	14.24	260	0	17	09-02-20	16:00:07	0	0	22
7	20003	Rodriguez 552	A	13.71	13.76	260	0	17	09-02-20	16:00:09	0	0	28

**Figura 4.1:** Visualización de una porción del contenido del sitio web <http://gpssumo.com/parquimetros>.

Los datos mostrados se actualizan constantemente por cada transacción realizada en el sistema.

Si comparamos los datos del dataset 2018 con los de la tabla (figura 4.1) se puede apreciar la similitud en cuanto al formato de los datos. Los atributos de principal interés son: id (si se resta el valor 20.000 se encuentran los id ya conocidos), dirección (dato que no poseíamos), fecha, hora, ocupación actual y lugares cuadra (estimación que realiza el sistema SUMO a la ocupación máxima permitida por cada cuadra).

El primer problema que surgió al intentar extraer los datos fue que la tabla de la página se genera con código JavaScript, por lo que al obtener el HTML por medio de BeautifulSoup (librería Python para extraer información de archivos HTML y XML) se mostraba una tabla vacía. Se intentó con otras librerías pero el resultado no varió. Es por ello que se decidió ingresar al código fuente del sitio (F12 en Google Chrome)(figura 4.2), buscando otra alternativa.

```
...<html> == $0
  ▶ <head>...</head>
  ▼<body>
    ▶ <table id="table_park" cellspacing="0" cellpadding="0">...</table>
    ▶ <script>...</script>
  </body>
</html>
```

**Figura 4.2:** Código presentado al oprimir F12 en el sitio web <http://gpssumo.com/parquimetros>.

El apartado table mostraba información sobre la composición de la tabla, el orden que debían llevar y cómo debería estar distribuida. El apartado script contenía funciones y, en una de ellas, se visualizó una referencia a un sitio web diferente ([http://gpssumo.com/parquimetros/get\\_PA/+order](http://gpssumo.com/parquimetros/get_PA/+order))(figura 4.5).

```

<html>
  > <head>...</head>
  ><body>
    > <table id="table_park" cellspacing="0" cellpadding="0">...</table>
.. ><script> == $0
    function set_table_PA(order)
    {
        $.ajax(
            {
                type      : 'POST',
                url       :
                "http://gpssumo.com/parquimetros/get_PA/" + order,
                data      : "",
                success   : function(msg)
                {
                    var e = eval ('('+msg+')');

                    var str = "";
                    var par = true;
                    var clase = 'even';
                    var count = 0 ;
                    var tot_aperturas = 0 ; var tot_estados = 0 ; var
tot_ocupacion = 0 ; var tot_max = 0 ; var tot_lugares_cuadra = 0 ;

```

**Figura 4.3:** Ampliación del apartado script en el código presentado al apretar F12 en el sitio web <http://gpssumo.com/parquimetros>. Se realiza un pedido de datos a otro sitio web por medio de código JavaScript.

Surgió la necesidad de comprobar si la palabra order hacía referencia al orden que se deseaba tener en los datos de la nueva web ya que al ingresar al link del script se generaba un error en la base de datos. Se incluyó id\_cuadra, resultando en [http://gpssumo.com/parquimetros/get\\_PA/id\\_cuadra](http://gpssumo.com/parquimetros/get_PA/id_cuadra). Esta web contiene la información mostrada en la tabla del primer sitio encontrado (figura 4.1), aunque en formato HTML. Tener los datos de esta forma permite extraer la información necesaria. En la figura 4.4 pueden observarse los datos mostrados en el sitio web, y en la figura 4.5 se aprecia cómo son presentados los datos en el código fuente.

```
[{"codigo": "20001", "id_cuadra": "2", "latitud": "-37.327857", "longitud": "-59.136795", "direccion": "General Pinto 545", "zona_tarifa": "A", "tp": "13.48", "tb": "14.23", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:10:31", "trans_prk_dia": "141", "trans_estado_dia": "54", "ocupacion": "0", "ocupacion_max": "13", "lugares_cuadra": "27", "ultimo_nro_tran": "398904", "color": "green"}, {"codigo": "20002", "id_cuadra": "3", "latitud": "-37.327089", "longitud": "-59.138667", "direccion": "San MartAn 452", "zona_tarifa": "A", "tp": "13.08", "tb": "13.28", "version_fw": "260", "fecha_e": "2020-03-04 17:15:00", "fecha_a": "04-03-20", "hora_a": "17:15:51", "trans_prk_dia": "89", "trans_estado_dia": "54", "ocupacion": "4", "ocupacion_max": "7", "lugares_cuadra": "22", "ultimo_nro_tran": "265384", "color": "green"}, {"codigo": "20003", "id_cuadra": "4", "latitud": "-37.326948", "longitud": "-59.136471", "direccion": "Rodríguez 552", "zona_tarifa": "A", "tp": "13.56", "tb": "13.46", "version_fw": "260", "fecha_e": "2020-03-04 17:14:00", "fecha_a": "04-03-20", "hora_a": "17:14:30", "trans_prk_dia": "211", "trans_estado_dia": "54", "ocupacion": "7", "ocupacion_max": "17", "lugares_cuadra": "28", "ultimo_nro_tran": "347457", "color": "green"}, {"codigo": "20004", "id_cuadra": "5", "latitud": "-37.327848", "longitud": "-59.134263", "direccion": "Julio 441", "zona_tarifa": "A", "tp": "11.88", "tb": "12.46", "version_fw": "260", "fecha_e": "2020-03-04 17:14:00", "fecha_a": "04-03-20", "hora_a": "17:14:50", "trans_prk_dia": "166", "trans_estado_dia": "54", "ocupacion": "9", "ocupacion_max": "12", "lugares_cuadra": "26", "ultimo_nro_tran": "416768", "color": "green"}, {"codigo": "20005", "id_cuadra": "6", "latitud": "-37.326729", "longitud": "-59.133841", "direccion": "General Pinto 759", "zona_tarifa": "A", "tp": "18.39", "tb": "14.21", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:10:25", "trans_prk_dia": "172", "trans_estado_dia": "54", "ocupacion": "7", "ocupacion_max": "14", "lugares_cuadra": "30", "ultimo_nro_tran": "416244", "color": "green"}, {"codigo": "20006", "id_cuadra": "7", "latitud": "-37.325524", "longitud": "-59.134563", "direccion": "San MartAn 758", "zona_tarifa": "A", "tp": "15.2", "tb": "13.98", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:10:33", "trans_prk_dia": "81", "trans_estado_dia": "45", "ocupacion": "4", "ocupacion_max": "10", "lugares_cuadra": "28", "ultimo_nro_tran": "290840", "color": "green"}, {"codigo": "20007", "id_cuadra": "8", "latitud": "-37.324964", "longitud": "-59.136926", "direccion": "Sarmiento 641", "zona_tarifa": "A", "tp": "18.89", "tb": "14.77", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:11:57", "trans_prk_dia": "84", "trans_estado_dia": "58", "ocupacion": "9", "ocupacion_max": "15", "lugares_cuadra": "22", "ultimo_nro_tran": "325071", "color": "green"}, {"codigo": "20008", "id_cuadra": "9", "latitud": "-37.326575", "longitud": "-59.137315", "direccion": "San MartAn 560", "zona_tarifa": "A", "tp": "12.97", "tb": "13.73", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:10:34", "trans_prk_dia": "44", "trans_estado_dia": "54", "ocupacion": "2", "ocupacion_max": "8", "lugares_cuadra": "22", "ultimo_nro_tran": "301830", "color": "green"}, {"codigo": "20009", "id_cuadra": "10", "latitud": "-37.325723", "longitud": "-59.137192", "direccion": "Rodríguez 654", "zona_tarifa": "A", "tp": "13.38", "tb": "13.94", "version_fw": "260", "fecha_e": "2020-03-04 17:13:00", "fecha_a": "04-03-20", "hora_a": "17:14:11", "trans_prk_dia": "149", "trans_estado_dia": "54", "ocupacion": "4", "ocupacion_max": "13", "lugares_cuadra": "24", "ultimo_nro_tran": "352972", "color": "green"}, {"codigo": "20010", "id_cuadra": "11", "latitud": "-37.326071", "longitud": "-59.135994", "direccion": "San MartAn 658", "zona_tarifa": "A", "tp": "12.46", "tb": "13.29", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:15:43", "trans_prk_dia": "179", "trans_estado_dia": "54", "ocupacion": "9", "ocupacion_max": "16", "lugares_cuadra": "30", "ultimo_nro_tran": "465219", "color": "green"}, {"codigo": "20011", "id_cuadra": "12", "latitud": "-37.329495", "longitud": "-59.136805", "direccion": "Belengra 480", "zona_tarifa": "A", "tp": "12.43", "tb": "12.99", "version_fw": "260", "fecha_e": "2020-03-04 17:12:00", "fecha_a": "04-03-20", "hora_a": "17:12:35", "trans_prk_dia": "122", "trans_estado_dia": "54", "ocupacion": "4", "ocupacion_max": "14", "lugares_cuadra": "25", "ultimo_nro_tran": "285189", "color": "green"}, {"codigo": "20012", "id_cuadra": "13", "latitud": "-37.328875", "longitud": "-59.139382", "direccion": "General Pinto 343", "zona_tarifa": "A", "tp": "18.24", "tb": "14.74", "version_fw": "259", "fecha_e": "2020-03-04 17:15:00", "fecha_a": "04-03-20", "hora_a": "17:15:52", "trans_prk_dia": "110", "trans_estado_dia": "54", "ocupacion": "5", "ocupacion_max": "23", "lugares_cuadra": "33", "ultimo_nro_tran": "300257", "color": "green"}, {"codigo": "20013", "id_cuadra": "14", "latitud": "-37.328057", "longitud": "-59.135815", "direccion": "Rodríguez 464", "zona_tarifa": "A", "tp": "17.66", "tb": "15.13", "version_fw": "260", "fecha_e": "2020-03-04 17:10:00", "fecha_a": "04-03-20", "hora_a": "17:10:35", "trans_prk_dia": "100", "trans_estado_dia": "54", "ocupacion": "4", "ocupacion_max": "13", "lugares_cuadra": "30", "ultimo_nro_tran": "365562", "color": "green"}, {"codigo": "20014", "id_cuadra": "15", "latitud": "-37.327418", "longitud": "-59.135649", "direccion": "General Pinto 631", "zona_tarifa": "A", "tp": "12.34", "tb": "13.29", "version_fw": "260", "fecha_e": "2020-03-04 17:14:00", "fecha_a": "04-03-20", "hora_a": "17:14:16", "trans_prk_dia": "149", "trans_estado_dia": "54", "ocupacion": "8", "ocupacion_max": "10", "lugares_cuadra": "25", "ultimo_nro_tran": "363728", "color": "green"}, {"codigo": "20015", "id_cuadra": "16", "latitud": "-37.325004", "longitud": "-59.133195", "direccion": "San MartAn 852", "zona_tarifa": "A", "tp": "16.03", "tb": "14.94", "version_fw": "260", "fecha_e": "2020-03-04 17:12:00", "fecha_a": "04-03-20", "hora_a": "17:13:00", "trans_prk_dia": "128", "trans_estado_dia": "56", "ocupacion": "6", "ocupacion_max": "12", "ultimo_nro_tran": "329840", "color": "green"}]
```

**Figura 4.4:** Visualización de los datos en el sitio web

[http://gpssumo.com/parquimetros/get\\_PA/id\\_cuadra](http://gpssumo.com/parquimetros/get_PA/id_cuadra). Al igual que en el sitio presentado en la figura 4.1, los datos se actualizan constantemente.

```
<html>
  <head></head>
  ...<body> == $0
    "[
      {
        "codigo": "20001",
        "id_cuadra": "2",
        "latitud": "-37.327857",
        "longitud": "-59.136795",
        "direccion": "General Pinto 545",
        "zona_tarifa": "A",
        "tp": "19.35",
        "tb": "15.08",
        "version_fw": "260",
        "fecha_e": "2020-03-10 16:10:00",
        "fecha_a": ""
      },
      ...
      {
        "codigo": "20002",
        "id_cuadra": "3",
        "latitud": "-37.327089",
        "longitud": "-59.138667",
        "direccion": "San MartAn 452",
        "zona_tarifa": "A",
        "tp": "13.08",
        "tb": "13.28",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:15:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:15:51",
        "trans_prk_dia": "89",
        "trans_estado_dia": "54",
        "ocupacion": "4",
        "ocupacion_max": "7",
        "lugares_cuadra": "22",
        "ultimo_nro_tran": "265384",
        "color": "green"
      },
      ...
      {
        "codigo": "20003",
        "id_cuadra": "4",
        "latitud": "-37.326948",
        "longitud": "-59.136471",
        "direccion": "Rodríguez 552",
        "zona_tarifa": "A",
        "tp": "13.56",
        "tb": "13.46",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:14:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:14:30",
        "trans_prk_dia": "211",
        "trans_estado_dia": "54",
        "ocupacion": "7",
        "ocupacion_max": "17",
        "lugares_cuadra": "28",
        "ultimo_nro_tran": "347457",
        "color": "green"
      },
      ...
      {
        "codigo": "20004",
        "id_cuadra": "5",
        "latitud": "-37.327848",
        "longitud": "-59.134263",
        "direccion": "Julio 441",
        "zona_tarifa": "A",
        "tp": "11.88",
        "tb": "12.46",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:14:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:14:50",
        "trans_prk_dia": "166",
        "trans_estado_dia": "54",
        "ocupacion": "9",
        "ocupacion_max": "15",
        "lugares_cuadra": "22",
        "ultimo_nro_tran": "325071",
        "color": "green"
      },
      ...
      {
        "codigo": "20005",
        "id_cuadra": "6",
        "latitud": "-37.326729",
        "longitud": "-59.133841",
        "direccion": "General Pinto 759",
        "zona_tarifa": "A",
        "tp": "18.39",
        "tb": "14.21",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:10:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:10:25",
        "trans_prk_dia": "172",
        "trans_estado_dia": "54",
        "ocupacion": "7",
        "ocupacion_max": "14",
        "lugares_cuadra": "30",
        "ultimo_nro_tran": "416768",
        "color": "green"
      },
      ...
      {
        "codigo": "20006",
        "id_cuadra": "7",
        "latitud": "-37.325524",
        "longitud": "-59.134563",
        "direccion": "San MartAn 758",
        "zona_tarifa": "A",
        "tp": "15.2",
        "tb": "13.98",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:10:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:10:33",
        "trans_prk_dia": "81",
        "trans_estado_dia": "45",
        "ocupacion": "4",
        "ocupacion_max": "10",
        "lugares_cuadra": "28",
        "ultimo_nro_tran": "290840",
        "color": "green"
      },
      ...
      {
        "codigo": "20007",
        "id_cuadra": "8",
        "latitud": "-37.324964",
        "longitud": "-59.136926",
        "direccion": "Sarmiento 641",
        "zona_tarifa": "A",
        "tp": "18.89",
        "tb": "14.77",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:10:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:11:57",
        "trans_prk_dia": "84",
        "trans_estado_dia": "58",
        "ocupacion": "9",
        "ocupacion_max": "15",
        "lugares_cuadra": "22",
        "ultimo_nro_tran": "325071",
        "color": "green"
      },
      ...
      {
        "codigo": "20008",
        "id_cuadra": "9",
        "latitud": "-37.326575",
        "longitud": "-59.137315",
        "direccion": "San MartAn 560",
        "zona_tarifa": "A",
        "tp": "12.97",
        "tb": "13.73",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:10:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:10:34",
        "trans_prk_dia": "44",
        "trans_estado_dia": "54",
        "ocupacion": "2",
        "ocupacion_max": "8",
        "lugares_cuadra": "22",
        "ultimo_nro_tran": "301830",
        "color": "green"
      },
      ...
      {
        "codigo": "20009",
        "id_cuadra": "10",
        "latitud": "-37.325723",
        "longitud": "-59.137192",
        "direccion": "Rodríguez 654",
        "zona_tarifa": "A",
        "tp": "13.38",
        "tb": "13.94",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:13:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:14:11",
        "trans_prk_dia": "149",
        "trans_estado_dia": "54",
        "ocupacion": "4",
        "ocupacion_max": "13",
        "lugares_cuadra": "24",
        "ultimo_nro_tran": "285189",
        "color": "green"
      },
      ...
      {
        "codigo": "20010",
        "id_cuadra": "11",
        "latitud": "-37.326071",
        "longitud": "-59.135994",
        "direccion": "San MartAn 658",
        "zona_tarifa": "A",
        "tp": "12.46",
        "tb": "13.29",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:10:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:15:43",
        "trans_prk_dia": "179",
        "trans_estado_dia": "54",
        "ocupacion": "9",
        "ocupacion_max": "16",
        "lugares_cuadra": "30",
        "ultimo_nro_tran": "465219",
        "color": "green"
      },
      ...
      {
        "codigo": "20011",
        "id_cuadra": "12",
        "latitud": "-37.329495",
        "longitud": "-59.136805",
        "direccion": "Belengra 480",
        "zona_tarifa": "A",
        "tp": "12.43",
        "tb": "12.99",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:12:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:12:35",
        "trans_prk_dia": "122",
        "trans_estado_dia": "54",
        "ocupacion": "4",
        "ocupacion_max": "14",
        "lugares_cuadra": "25",
        "ultimo_nro_tran": "285189",
        "color": "green"
      },
      ...
      {
        "codigo": "20012",
        "id_cuadra": "13",
        "latitud": "-37.328875",
        "longitud": "-59.139382",
        "direccion": "General Pinto 343",
        "zona_tarifa": "A",
        "tp": "18.24",
        "tb": "14.74",
        "version_fw": "259",
        "fecha_e": "2020-03-04 17:15:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:15:52",
        "trans_prk_dia": "110",
        "trans_estado_dia": "54",
        "ocupacion": "5",
        "ocupacion_max": "23",
        "lugares_cuadra": "33",
        "ultimo_nro_tran": "300257",
        "color": "green"
      },
      ...
      {
        "codigo": "20013",
        "id_cuadra": "14",
        "latitud": "-37.328057",
        "longitud": "-59.135815",
        "direccion": "Rodríguez 464",
        "zona_tarifa": "A",
        "tp": "17.66",
        "tb": "15.13",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:10:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:10:35",
        "trans_prk_dia": "100",
        "trans_estado_dia": "54",
        "ocupacion": "4",
        "ocupacion_max": "13",
        "lugares_cuadra": "30",
        "ultimo_nro_tran": "365562",
        "color": "green"
      },
      ...
      {
        "codigo": "20014",
        "id_cuadra": "15",
        "latitud": "-37.327418",
        "longitud": "-59.135649",
        "direccion": "General Pinto 631",
        "zona_tarifa": "A",
        "tp": "12.34",
        "tb": "13.29",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:14:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:14:16",
        "trans_prk_dia": "149",
        "trans_estado_dia": "54",
        "ocupacion": "8",
        "ocupacion_max": "10",
        "lugares_cuadra": "25",
        "ultimo_nro_tran": "363728",
        "color": "green"
      },
      ...
      {
        "codigo": "20015",
        "id_cuadra": "16",
        "latitud": "-37.325004",
        "longitud": "-59.133195",
        "direccion": "San MartAn 852",
        "zona_tarifa": "A",
        "tp": "16.03",
        "tb": "14.94",
        "version_fw": "260",
        "fecha_e": "2020-03-04 17:12:00",
        "fecha_a": "04-03-20",
        "hora_a": "17:13:00",
        "trans_prk_dia": "128",
        "trans_estado_dia": "56",
        "ocupacion": "6",
        "ocupacion_max": "12",
        "ultimo_nro_tran": "329840",
        "color": "green"
      }
    ]"
```

**Figura 4.5:** Visualización del contenido del sitio web [http://gpssumo.com/parquimetros/get\\_PA/id\\_cuadra](http://gpssumo.com/parquimetros/get_PA/id_cuadra) al oprimir F12 y ver el código fuente. Se observan todos los datos en el HTML en lugar de hacer un pedido a una base de datos por JavaScript.

A partir de esto, se ejecutó de manera continua un script, el cual extrae la información del sitio a cada momento, en una computadora servidor. Como puede observarse en la tabla 4.3, los datos no poseen operación, por lo fue agregada para mantener la mayor cantidad de atributos en común entre todos los datasets. Este atributo fue calculado viendo si el siguiente valor de

cada cuadra aumenta o disminuye, en cuyo caso la operación era una entrada o una salida, respectivamente.

id_cuadra	direccion	fecha	tiempo	ocupacion	lugares_cuadra	ocupacion_max	operacion
2	General Pinta 545	11-09-19	17:15:31	6	27.0	17	NaN
2	General Pinta 545	11-09-19	17:20:25	7	27.0	17	Entrada
2	General Pinta 545	11-09-19	17:34:54	6	27.0	17	Salida
2	General Pinta 545	11-09-19	17:40:09	5	27.0	17	Salida
2	General Pinta 545	11-09-19	17:45:05	6	27.0	17	Entrada

**Tabla 4.3:** Ejemplo de filas pertenecientes al dataset web scraping, una vez agregada la operación a cada fila del DataFrame. Filas que poseen valores NaN son las primeras extraídas.

Aunque se obtuvieron nuevos datos por este medio, seguían siendo insuficientes para un análisis completo. Conseguir un tercio de datos como los del dataset del 2018 llevó alrededor de tres meses. A su vez, esta información contenía baches por problemas del servidor en el cual se ejecutaba el programa (se reiniciaba). Es por esta razón que se optó por otras alternativas mientras se continuaba con el análisis de los datos.

#### 4.1.3. Dataset 2019

La principal diferencia entre el dataset 2018 y el obtenido mediante la extracción de datos web es que los de este último eran datos actualizados, por lo que el problema de la cantidad de datos seguía vigente.

En enero de 2020 surgió la iniciativa de solicitud de información pública en la ciudad de Tandil, por la cual todos los pedidos de información pública debían ser obligatoriamente respondidos y, en la mayoría de los casos, afirmativos. Por este medio se realizó la solicitud de los datos completos correspondientes al año 2019.

Los datos fueron entregados en un archivo comprimido con doce carpetas, una por cada mes, las cuales contenían un .zip correspondiente a cada día del mes. Cada .zip incluía el archivo solicitado en formato xls.

El primer problema al intentar utilizar estos datos surgió en los archivos. Éstos contenían dos tablas, como puede observarse en la figura 4.6, por lo que al leerlas utilizando el método `read_excel` de pandas generaba una excepción. Se probó en diferentes frameworks, en los cuales se encontraron los mismos resultados.

Se optó por cambiar el formato de los archivos por medio de bash, cambiando la extensión de los archivos de xls a xlsx. De este modo aún se continuaban leyendo las dos tablas, aunque ahora almacenada como un solo DataFrame. Este DataFrame contenía todos los datos de la

primera tabla y, a continuación de su último dato, los nombres de las columnas de la siguiente tabla junto con sus respectivos datos, por lo que realizando un simple filtrado podemos obtener la sección de la tabla deseada.

Parquímetro	Transacción	Fecha Recepción	Fecha Emisión	Operación	Tarjeta	Patente	Tiempo Estacionamiento	Importe	Saldo	Zona
20016	275465	01-01-2019 [10:55:45]	01-01-2019 [10:55:00]	SALIDA ESTACIONAMIENTO	10013535	[REDACTED]	10:00	239,56	252,58 A (17.86 - 21.73 - 26.73)	
20060	247852	01-01-2019 [16:25:51]	01-01-2019 [16:25:00]	INGRESO ESTACIONAMIENTO	10137236	[REDACTED]	0:00	0	56,24 A (17.86 - 21.73 - 26.73)	
20060	247854	01-01-2019 [16:26:01]	01-01-2019 [16:25:00]	SALIDA ESTACIONAMIENTO	10137236	[REDACTED]	0:00	0	56,24 A (17.86 - 21.73 - 26.73)	
20060	247855	01-01-2019 [16:26:08]	01-01-2019 [16:25:00]	INGRESO ESTACIONAMIENTO	10137236	[REDACTED]	0:00	0	56,24 A (17.86 - 21.73 - 26.73)	
20060	247857	01-01-2019 [17:11:58]	01-01-2019 [17:11:00]	SALIDA ESTACIONAMIENTO	10137236	[REDACTED]	0:46	0	56,24 A (17.86 - 21.73 - 26.73)	
20076	36503	01-01-2019 [15:17:27]	01-01-2019 [15:17:27]	SALIDA ESTACIONAMIENTO APP	2494249452	[REDACTED]	76:42:00	66,14	3,21 A (17.86 - 21.73 - 26.73)	
20076	218668	01-01-2019 [23:37:04]	01-01-2019 [23:35:00]	INGRESO ESTACIONAMIENTO	10171252	[REDACTED]	0:00	0	119,93 A (17.86 - 21.73 - 26.73)	
20076	218669	01-01-2019 [23:37:10]	01-01-2019 [23:35:00]	SALIDA ESTACIONAMIENTO	10171252	[REDACTED]	0:00	0	119,93 A (17.86 - 21.73 - 26.73)	
20012	36502	01-01-2019 [13:41:39]	01-01-2019 [13:41:35]	SALIDA ESTACIONAMIENTO APP	2494015578	[REDACTED]	97:15:00	92,87	-17,86 A (17.86 - 21.73 - 26.73)	
20001	306473	01-01-2019 [15:13:17]	01-01-2019 [15:12:00]	INGRESO ESTACIONAMIENTO	10017528	[REDACTED]	0:00	0	91,07 A (17.86 - 21.73 - 26.73)	
20001	306474	01-01-2019 [15:15:52]	01-01-2019 [15:15:00]	SALIDA ESTACIONAMIENTO	10017528	[REDACTED]	0:03	0	91,07 A (17.86 - 21.73 - 26.73)	
20065	202212	01-01-2019 [15:13:40]	01-01-2019 [15:13:00]	INGRESO ESTACIONAMIENTO	10126135	[REDACTED]	0:00	0	98,69 A (17.86 - 21.73 - 26.73)	
20065	202214	01-01-2019 [16:22:34]	01-01-2019 [16:22:00]	SALIDA ESTACIONAMIENTO	10126135	[REDACTED]	1:09	0	98,69 A (17.86 - 21.73 - 26.73)	
20009	268568	01-01-2019 [20:32:49]	01-01-2019 [20:32:00]	INGRESO ESTACIONAMIENTO	10147821	[REDACTED]	0:00	0	46,32 A (17.86 - 21.73 - 26.73)	
20018	247987	01-01-2019 [17:10:18]	01-01-2019 [17:10:00]	INGRESO ESTACIONAMIENTO	10196701	[REDACTED]	0:00	0	100 A (17.86 - 21.73 - 26.73)	
20018	247988	01-01-2019 [17:10:34]	01-01-2019 [17:10:00]	SALIDA ESTACIONAMIENTO	10196701	[REDACTED]	0:00	0	100 A (17.86 - 21.73 - 26.73)	

Parquímetro	Dirección	Cuadra	Zona	Aperturas	Número de operaciones por tipo						Importe
					Cierres	Reingresos	Cierres en Descubierto	Cierres Sin Cargo	Tiempo Total Estacionado		
20016 Belgrano 660	Belgrano 600-700	A (17.86 - 21.73 - 26.73)		0	1	0	0	0	10:00	239,56	
20060 Chacabuco 357	Chacabuco 300-400	A (17.86 - 21.73 - 26.73)		2	0	0	0	2	0:46	0	
20076 General Paz 547	General Paz 500-600	A (17.86 - 21.73 - 26.73)		1	1	0	0	1	76:42:00	66,14	

**Figura 4.6:** Visualización de uno de los archivos obtenidos mediante la petición de datos. Se observan las dos tablas en el mismo archivo. Se oculta la información de la patente para mantener la privacidad de los usuarios.

Estos datos, los cuales resultaron ser más de tres millones, tienen un formato muy similar a los anteriormente analizados (tabla 4.4), con la diferencia de las operaciones, las cuales podían ser: Salida, Entrada y Cierre, la cual se generaba automáticamente por falta de saldo. Para el análisis realizado se tomó los cierres como si fuesen salidas.

parquimetro	fecha_emision	tiempo_emision	operacion	tiempo_estacionado
16	2019-01-01	10:55:00	0	10:00
16	2019-01-02	06:49:00	1	0:00
16	2019-01-02	08:26:00	1	0:00
16	2019-01-02	09:19:00	1	0:00
16	2019-01-02	09:20:00	1	0:00

**Tabla 4.4:** Ejemplo de filas pertenecientes al dataset 2019, las cuales contienen fecha, hora, parquímetro (id de cuadra), operación (1: Entrada/0: Salida) y tiempo total estacionado.

La dirección y las coordenadas eran diferentes aunque del mismo parquímetro se realice la transacción, por lo que se optó por utilizar los valores de los otros datasets. Suponemos que esto se debe al uso de la aplicación móvil del sistema, el cual no asocia a ningún parquímetro y coloca la dirección actual del usuario. Para nuestro sistema únicamente interesa conocer la

ocupación por cuadra, por lo que unificar todos los valores de direcciones cercanas en uno solo simplifica el análisis y no perjudica la información ofrecida por los datos (tabla 4.5).

parquimetro	fecha_emision	tiempo_emision	operacion	tiempo_estacionado	direccion	ocupacion	latitud	longitud
16	2019-01-01	10:55:00	0	10:00	Belgrano 660	0	-37.328618	-59.134464
16	2019-01-02	06:49:00	1	0:00	Belgrano 660	1	-37.328618	-59.134464
16	2019-01-02	08:26:00	1	0:00	Belgrano 660	2	-37.328618	-59.134464
16	2019-01-02	09:19:00	1	0:00	Belgrano 660	3	-37.328618	-59.134464
16	2019-01-02	09:20:00	1	0:00	Belgrano 660	4	-37.328618	-59.134464

**Tabla 4.5:** Ejemplo de filas pertenecientes al dataset 2019, una vez agregados la dirección, coordenadas y ocupación a cada fila del DataFrame.

## 4.2. Análisis de datos

Se realizó una comparativa entre los tres datasets, analizando patrones de comportamiento de los usuarios y diferencias entre ellos. El objetivo de este análisis es comprender los datos que se poseen, para ofrecer mejores predicciones.

En varios casos se realizó un análisis general, teniendo en cuenta que los datos que se aprecian en el gráfico siguen los mismos patrones sin importar el dataset. Estos corresponden al mismo sistema de estacionamiento, por lo que las diferencias que posean son motivo del tiempo en el que se obtuvieron los datos o del modo en que se obtuvieron. El objetivo de realizar análisis sobre ellos es poder comprobar similitudes y diferencias, ya que mientras más se avanzaba en el análisis, más falta hacía un conjunto de datos aún mayor y actualizado.

Se propusieron distintos análisis comparando el promedio de los valores de todo el conjunto de datos con los valores ofrecidos por una sola cuadra, buscando diferencias entre ambos. Se concluyó que los usuarios del sistema siguen el mismo patrón de estacionamiento sin importar la cuadra en cuestión. Por esta razón no se realizaron análisis profundos sobre determinada cuadra, sino sobre el conjunto total de datos.

### 4.2.1. Problemas iniciales

Luego de realizar un primer análisis del conjunto de datos obtenidos y de realizar averiguaciones en los organismos a cargo del sistema, se plantearon dos problemas principales.

El primero se basa en si un automovilista estaciona su vehículo y no ingresa al sistema de estacionamiento, lo que se considera una infracción. En este caso se podría llegar a esperar que haya lugar libre de estacionamiento, aunque en realidad esté ocupado por lo que llamaremos “auto fantasma”.

El segundo problema, aún más común que el anterior, es el caso donde los conductores estacionan en una cuadra e ingresan al sistema de estacionamiento desde un parquímetro ubicado en otra, por lo que el sistema registrará que el ingreso fue en otra dirección. Esto genera dos problemas, un ingreso fantasma, ya que no estacionó en la dirección en la cual ingresó y lo anteriormente llamado auto fantasma, ya que no hay manera de conocer la dirección real de este vehículo.

No se encontró solución a ninguno de los dos problemas, aunque se estima que son la minoría de usuarios los que realizan estas acciones, por lo que no conlleva gran problema a la hora de predecir lugar de estacionamiento.

#### 4.2.2. Identificadores

La primer comparativa entre los datasets fue la cantidad de calles que poseen. Como se comentó anteriormente, hubo cambios en los parquímetros a partir del 2019, por lo que se esperaba encontrar resultados diferentes sobre las referencias de los ids.

Al analizar la cantidad de calles (ids y direcciones) diferentes que posee cada dataset, encontramos un dato interesante: el número máximo de id es distinto en los tres casos (tabla 4.6).

	Número máximo de id	Cantidad de id diferentes
<b>Dataset 2018</b>	92	90
<b>Dataset web scraping</b>	99	97
<b>Dataset 2019</b>	190	97

**Tabla 4.6:** cantidad de ids y valor máximo del mismo por dataset.

Desde el 2018 se agregaron parquímetros en las cuadras de las direcciones par de la Avenida España, por lo que es fácil comprender por qué tiene menor cantidad de calles diferentes. No es el mismo caso para las diferencias encontradas entre el dataset web scraping y el dataset 2019.

Analizando al detalle los id y las direcciones que posee cada dataset, se encontraron diferencias entre los mismos (tabla 4.7).

<b>id_cuadra</b>	<b>2018_direccion</b>	<b>ws_direccion</b>	<b>2019_direccion</b>
190	NaN	NaN	Tribunal de faltas
99	NaN	Av. España 970	NaN
98	NaN	Av. España 864	NaN
97	NaN	Av. España 774	NaN

**Tabla 4.7:** Visualización de identificadores junto con su dirección por dataset (2018: dataset 2018, ws: dataset web scraping, 2019: dataset 2019).

Se observa que las direcciones pares en la Avenida España no se encuentran en el dataset 2018, como se enunció anteriormente, ya que éstas fueron agregadas en el año 2019.

Las direcciones del dataset web scraping y del dataset 2019 son referenciadas por ids completamente diferentes. Para corroborar que el par id/dirección era el correcto, se buscó en los parquímetros físicos de Tandil, los cuales poseen su identificador. Con esta información, se llegó a la conclusión de que los valores reales son los del dataset 2019.

A partir de estas comparaciones se planteó la siguiente cuestión: por qué el dataset web scraping posee mal el par id/dirección, siendo que los datos fueron obtenidos de la web y eran datos actualizados. Se realizó un análisis en profundidad y se encontró, en la web [http://gpssumo.com/parquimetros/get\\_PA/id\\_cuadra](http://gpssumo.com/parquimetros/get_PA/id_cuadra), que cada dato posee dos identificadores. El primero, llamado código es igual al id presente en el dataset 2019 y, el segundo es id\_cuadra, el identificador de la cuadra correspondiente al dataset 2018. El código no se encontraba presente en el dataset web scraping ya que, al poseer el identificador de cuadra, no era relevante.

En resumen el dataset web scraping posee dos identificadores: código e id\_cuadra. El primero es el utilizado por el dataset 2019 como identificador y, el segundo, es utilizado por el dataset 2018 como identificador.

Para facilitar y evitar futuros inconvenientes se realizaron comparaciones por dirección en vez de por id.

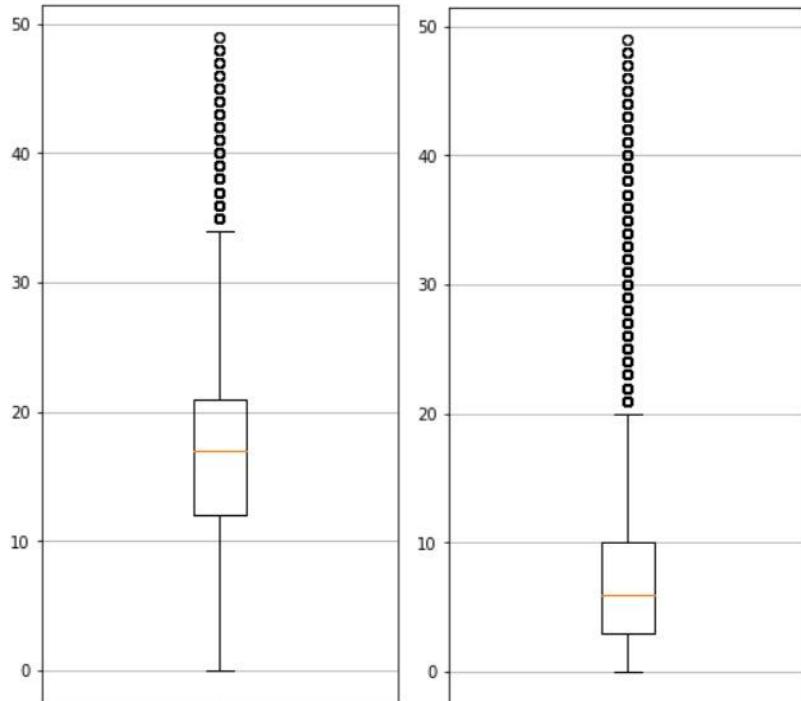
#### **4.2.3. Diagramas de caja**

Se plantearon dos diagramas de caja (boxplots en inglés) con el fin de conocer con más exactitud los datos. Se utilizó únicamente el dataset 2019 ya que es el más completo de los tres. Si se utilizase un dataset diferente las variaciones se verían reflejadas en los valores mostrados y en la obtención de máximos, aunque la idea general continuaría siendo la misma.

Analizando el boxplot de la cuadra Sarmiento 755 (figura 4.7), la mayor parte de la ocupación se encuentra entre los valores 12 y 21, teniendo casos en los que desciende hasta 0 (donde no hay ningún vehículo estacionado) o asciende hasta 34. También se aprecian valores atípicos, es decir, valores numéricamente distintos del resto de los datos. Se recomienda eliminar estos valores atípicos del conjunto de datos, ya que representan un porcentaje mínimo y pueden deberse a un error o información equivocada (autos fantasma, ingresos fantasma, etc.).

En el gráfico donde se muestra la información completa del dataset 2019 (figura 4.7) se puede observar que la mayoría de los valores de ocupación se mantienen entre 3 y 10, con un máximo de 20 y un mínimo de 0. Se registra una gran cantidad de valores atípicos. Esto se debe a que muchas cuadras cuentan con una ocupación estable, mientras que otras, como las más céntricas, poseen un valor de ocupación muy elevado en fechas especiales o festivas.

Ambos diagramas de caja muestran que los datos no forman una distribución normal y que poseen valores atípicos o elevados, los cuales ocurren en el menor de los casos.



**Figura 4.7:** Visualización en box plot de los valores de ocupación en Sarmiento 755 (izquierda) y del dataset completo (derecha).

#### 4.2.4. Máximos por cuadra

Conocer el máximo de ocupación por cada cuadra es necesario para comprender en qué calles se encuentran los picos más altos de autos estacionados, es decir, dónde se concentra la mayor cantidad de autos estacionados.

Analizando los dataset por la cantidad máxima de estacionados, se observó que ciertas direcciones son comunes para los tres dataset dentro del rango más alto. Un punto a destacar es el cambio de máximos que se visualiza. En el dataset 2018 encontramos máximos pequeños (hasta 27 de ocupación) (tabla 4.8), mientras que en el dataset web scraping se visualizan valores de máximos menores que el dataset 2019 (tabla 4.9) (tabla 4.10), lo cual tiene sentido ya que es sólo una porción de este último.

Se prosiguió con el cálculo del promedio de los máximos de ocupación por día. Este planteamiento surge como una posibilidad para evitar, en los máximos, el ruido generado por los ingresos y autos fantasma.

Utilizando datos calculados en el apartado anterior, se observan los valores máximos obtenidos en los diagramas de caja por cada cuadra. Estos valores representan el valor más alto perteneciente al dataset, ignorando los outliers. Esta alternativa se plantea como la mejor opción a tener en cuenta para utilizar como máximos por cuadra, ya que se utiliza la distribución de los datos para calcularla.

direccion	max_estacionados	avg_estacionados	max_boxplot_estacionados
Yrigoyen 655	27	12.821429	19.0
General Pinto 545	26	8.972414	12.0
San Martin 658	24	11.233577	16.0
General Paz 447	22	10.908451	18.0
Sarmiento 755	20	9.953947	13.0

**Tabla 4.8:** Comparación entre valores máximos globales, promedio y máximo de boxplot (sin outliers) del dataset 2018.

direccion	max_estacionados	avg_estacionados	max_boxplot_estacionados
Yrigoyen 655	42	21.811321	30.0
Fuerte Independencia 349	29	14.490196	20.0
Maipu 453	27	11.058824	16.0
Rodriguez 464	27	10.596154	22.0
Maipu 333	27	15.591837	14.0

**Tabla 4.9:** Comparación entre valores máximos globales, promedio y máximo de boxplot (sin outliers) del dataset web scraping.

direccion	max_estacionados	avg_estacionados	max_boxplot_estacionados
Sarmiento 755	49	21.746269	34.0
San Martin 658	47	18.426380	29.0
Rodriguez 654	45	13.201258	29.0
Belgrano 660	45	16.933735	21.0
Fuerte Independencia 349	43	12.375394	20.0

**Tabla 4.10:** Comparación entre valores máximos globales, promedio y máximo de boxplot (sin outliers) del dataset 2019.

#### 4.2.5. Cantidad de operaciones

##### 4.2.5.1. Por cuadra

Se calculó la cantidad total de operaciones de entrada y salida realizadas por cada cuadra, obteniendo de este modo qué cuadras poseen el mayor movimiento (tabla 4.11) (tabla 4.12).

Se puede observar una relación entre la cantidad de operaciones que posee una calle junto con sus máximos. Aunque no es una relación directa, se puede deducir que las calles con más movimiento son en las que estacionan la mayor cantidad de autos gran parte del tiempo.

En las calles de la periferia (teniendo en cuenta el sector que abarcan los parquímetros) se encuentra el menor porcentaje de operaciones totales. Esto se debe a que los conductores, con el fin de no estacionar en el área de los parquímetros, se alejan de su destino y buscan cuadras no afectadas por el estacionamiento medido.

Un ejemplo de bajo porcentaje de operaciones es el tribunal de faltas. El mismo se encuentra en la terminal de la ciudad de Tandil, lugar que aún no se vio afectado por el estacionamiento medido. Este parquímetro se utiliza mayormente para abonar las multas pendientes.

direccion	2018_op_totales	ws_op_totales	2019_op_totales
San Martin 658	18784	5497	70322
Yrigoyen 655	16989	5971	65800
General Pinto 759	15274	5412	65397
Alem 480	14690	5118	60345
Rodriguez 552	15250	5207	60285

**Tabla 4.11:** cantidad de operaciones por cuadra según cada dataset. Calles con mayor cantidad de valores.

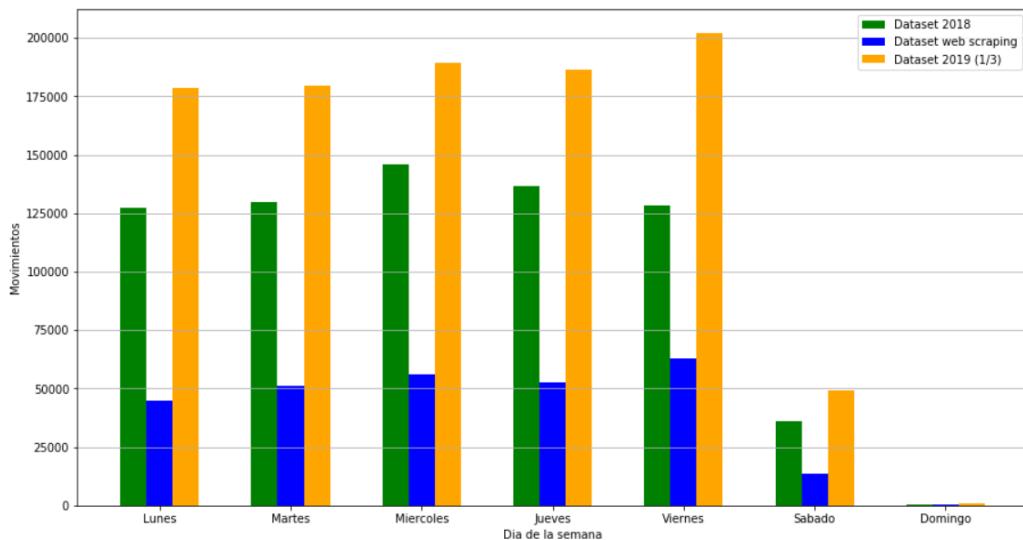
direccion	2018_op_totales	ws_op_totales	2019_op_totales
Tribunal de Faltas	22.0	8	298
Av. España 436	NaN	456	2978
Av. España 352	NaN	493	4137
14 de Julio 847	818.0	939	4435
Av. España 359	798.0	601	4681

**Tabla 4.12:** cantidad de operaciones por cuadra según cada dataset. Calles con menor cantidad de valores. Los NaN representan que ese dataset no posee la cuadra en cuestión.

#### 4.2.5.2. Por día de semana

En la figura 4.8 se puede observar, para cada dataset, la cantidad total de operaciones (entradas y salidas) realizadas por día de semana. Se observa que de lunes a viernes es donde hay mayor cantidad de movimiento. En el caso de los sábados, donde el estacionamiento medido sólo funciona hasta el mediodía, se visualizan menos operaciones y, los domingos, la actividad es casi nula, siendo el día en el que el estacionamiento medido no está en funcionamiento.

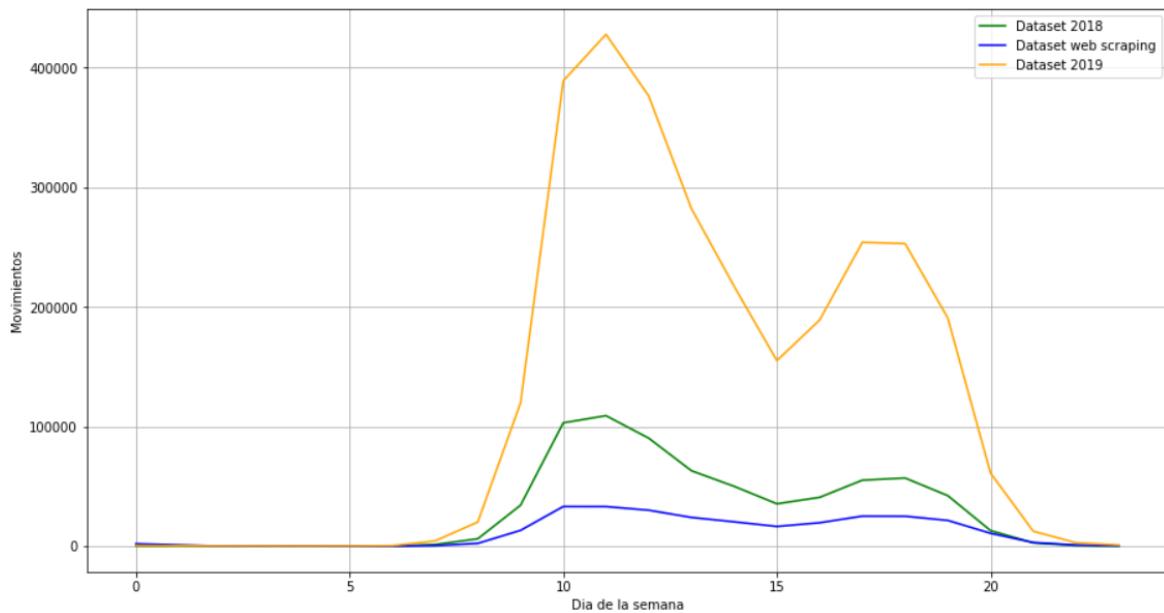
En el día viernes se visualiza un ligero incremento en el volumen de operaciones comparándolo con el resto de los días. Esto podría deberse a la gente que sale del trabajo y decide ir al centro a pasar el rato o por el movimiento del turismo en la ciudad.



**Figura 4.8:** Suma total de movimientos realizados por cada día de la semana sobre los tres datasets.

#### 4.2.5.3. Por hora

En la figura 4.9 se puede observar, para cada dataset, la cantidad total de operaciones (entradas y salidas) realizadas por cada hora del día. Se observa un gran volumen de operaciones totales desde las 10 de la mañana hasta las 12, donde luego disminuye drásticamente. El valor máximo con respecto a la cantidad de movimientos (entradas y salidas) se visualiza a las 11 horas. Durante el mediodía es donde hay más movimiento de vehículos en las calles de Tandil.

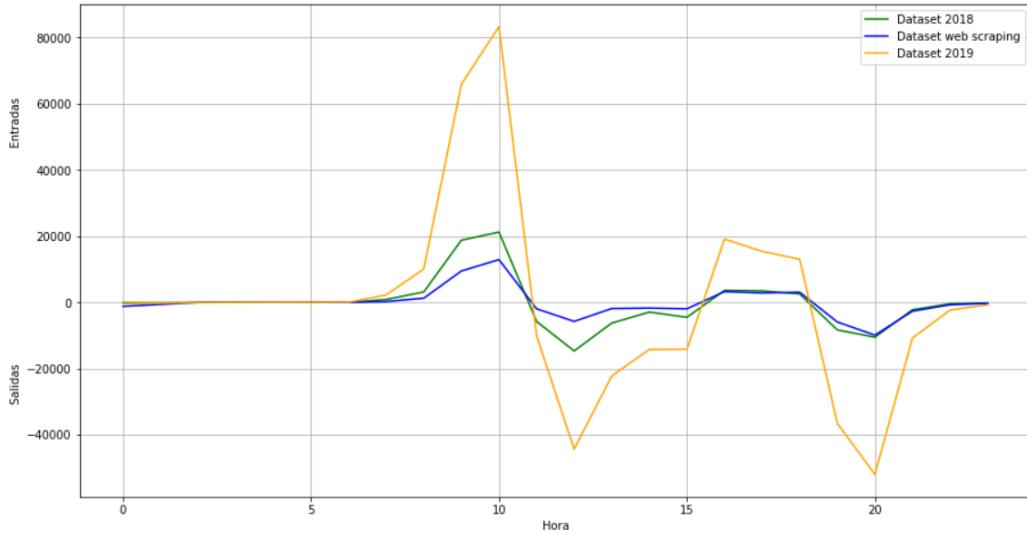


**Figura 4.9:** Suma total de movimientos realizados por cada hora del día sobre los tres datasets.

#### 4.2.6. Momentos de mayor ocupación

En la figura 4.10 se observa la cantidad de operaciones de entrada y de salida por cada hora del día. Dicha gráfica es similar a la realizada para la cantidad de operaciones por hora, reforzando los resultados obtenidos del análisis hasta el momento.

A partir de las 9 de la mañana se visualiza la mayor cantidad de entradas, lo que se traduce en bajas posibilidades de encontrar lugares para estacionar. Luego de las 11 de la mañana, se observa un gran número de operaciones de salida, trayendo como consecuencia la liberación de lugares de estacionamiento ocupados con anterioridad. A partir de las 16 horas, comienza a aumentar nuevamente pero en menor medida, hasta decaer llegando la noche.



**Figura 4.10:** Representación de cantidad de operaciones de entrada y salida durante un día. Valores positivos representan más entradas que salidas, mientras que los valores negativos representan más salidas que entradas.

#### 4.2.7. Estadísticas por mes

A partir de los gráficos de barras (figura 4.11, figura 4.12, figura 4.13) se puede visualizar fácilmente la cantidad de máximos por mes, los valores promedios de los mismos y las operaciones totales. El primero hace referencia a la cantidad de calles que poseen valores entre el máximo encontrado en la sección máximos por cuadra y un margen de error, contemplando así la posibilidad de los estacionamientos fantasma.

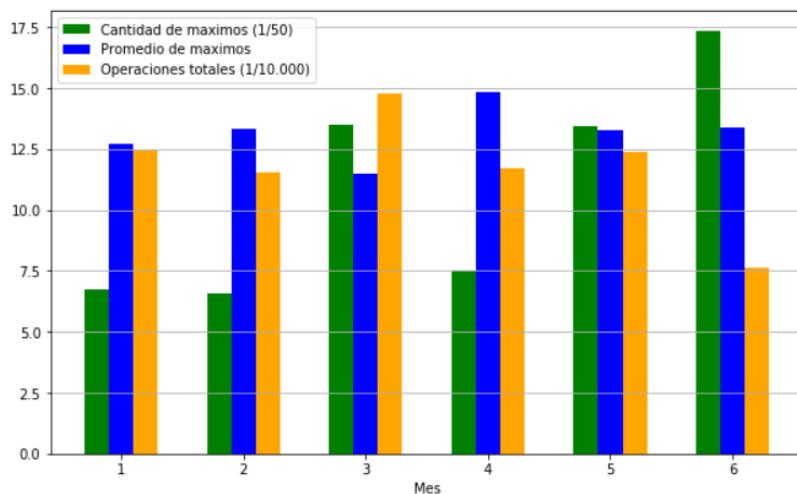
El dataset de 2018 contiene datos hasta el 14 de junio, por lo que cuenta con menos operaciones. Aún así, en el mes junio se visualiza el valor más elevado con respecto a la cantidad de máximos.

Con respecto al dataset de web scraping se analizó la diferencia entre el volumen de datos de cada mes a través de un gráfico de barras. La principal razón se debe a la cantidad desigual de datos recolectados por mes. No se reflejan la cantidad de operaciones reales. Como se comentó anteriormente, ocurrieron problemas mientras se obtenían los datos como, por ejemplo, la desconexión del servidor o fallas del mismo.

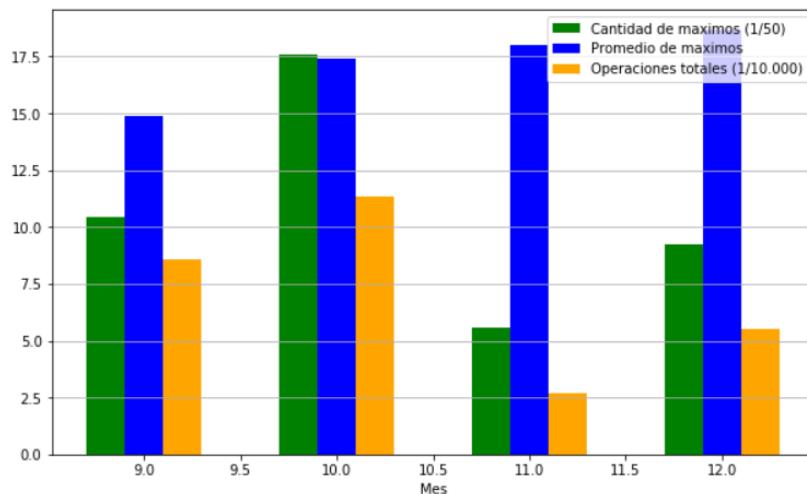
A partir del dataset del 2019 se logró realizar un análisis completo sobre la cantidad de máximos por mes, los valores promedios de los mismos y las operaciones totales de todos los meses del año. En primer lugar se observa que la cantidad de operaciones es similar entre todos los meses, por lo que se puede asumir un nivel de actividad estable a lo largo del año. Durante el primer cuatrimestre se visualiza el crecimiento en los valores de cantidad y promedio de máximos, indicando el incremento de vehículos estacionados simultáneamente.

En el mes de diciembre se observa que los valores de promedio y cantidad de máximos son los más elevados durante el año, mientras que la cantidad de operaciones totales es menor si se compara con meses anteriores. A partir de esto se deduce que la mayor cantidad de vehículos que ingresan al estacionamiento se mantienen a lo largo del día, trayendo como consecuencia que gran parte de los lugares para estacionar se encuentren ocupados.

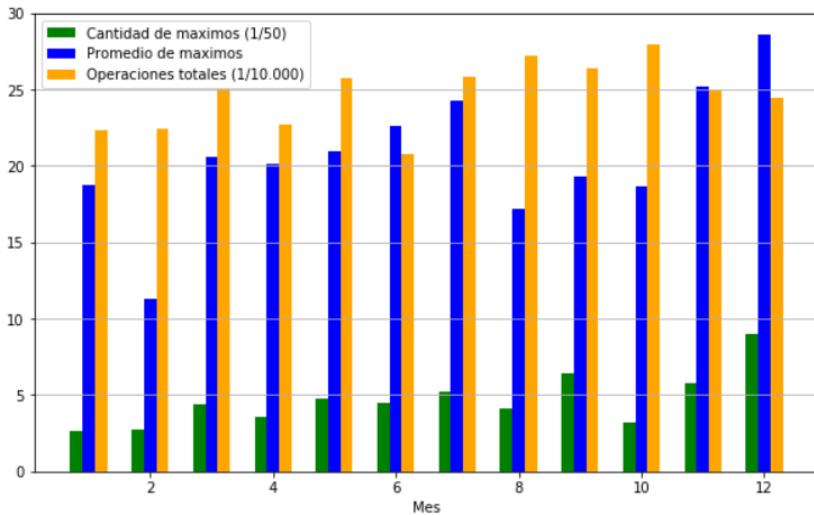
En enero y febrero es donde se visualiza la menor cantidad de autos.



**Figura 4.11:** División por meses de la cantidad de máximos (1/50 del valor total), el promedio entre estos y las operaciones totales (1/10.000 del valor total) realizadas en la ciudad de Tandil según el dataset 2018.



**Figura 4.12:** División por meses de la cantidad de máximos (1/50 del valor total), el promedio entre estos y las operaciones totales (1/10.000 del valor total) realizadas en la ciudad de Tandil según el dataset web scraping.

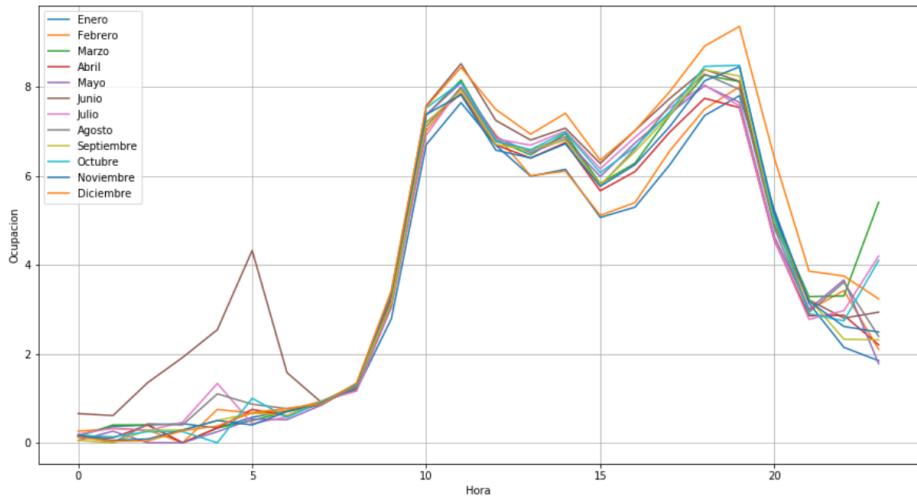


**Figura 4.13:** División por meses de la cantidad de máximos (1/50 del valor total), el promedio entre estos y las operaciones totales (1/10.000 del valor total) realizadas en la ciudad de Tandil según el dataset 2019.

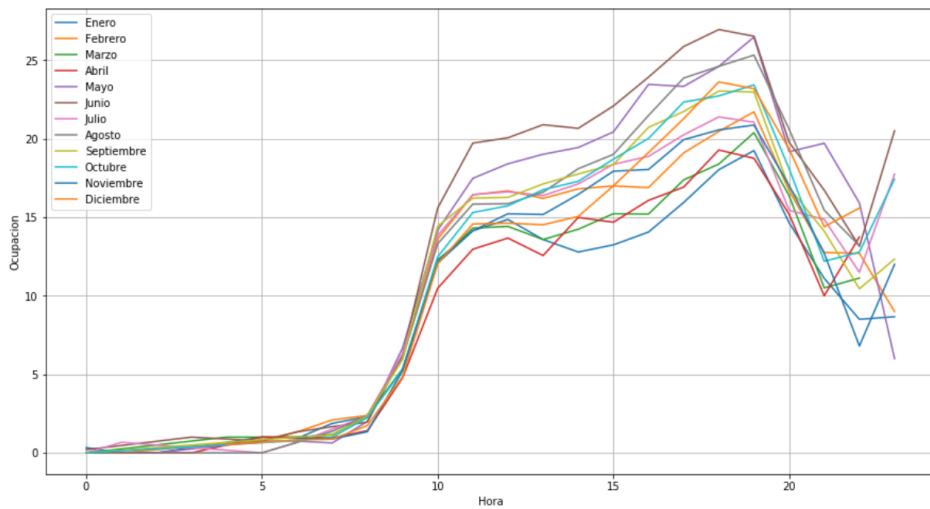
#### 4.2.8. Estacionamientos por mes

Se observa en los gráficos las diferencias en las ocupaciones sobre los datos dividido por mes. Estas diferencias fueron diagramadas únicamente con el dataset 2019 ya que es el único que posee datos a lo largo de un año completo. El objetivo de realizar este tipo de visualizaciones es comprender la importancia de utilizar este dato en el entrenamiento de los modelos de predicción.

A pesar de que los gráficos son similares para todos los meses, ya sea utilizando el conjunto completo de datos o solamente los pertenecientes a una cuadra, se notan diferencias en ocupación a lo largo del año. Esto evidencia las diferencias en los estacionamientos en cada mes, por ejemplo en la figura 4.15 se observa que en el mes de Noviembre hay mucha menos ocupación a lo largo de un día que en el mes de Junio.



**Figura 4.14:** Ocupación a lo largo de un día para cada mes del año, dataset 2019 completo.

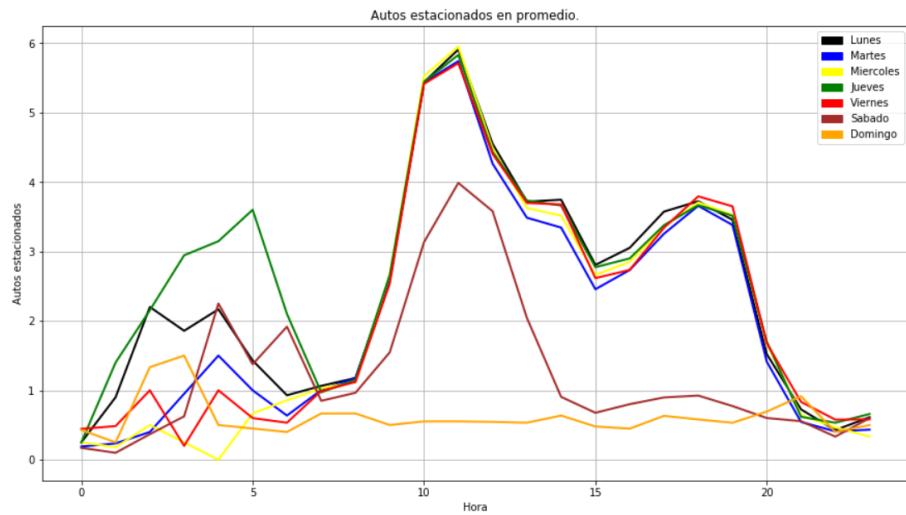


**Figura 4.15:** Ocupación a lo largo de un día para cada mes del año, datos de Sarmiento 755, dataset 2019.

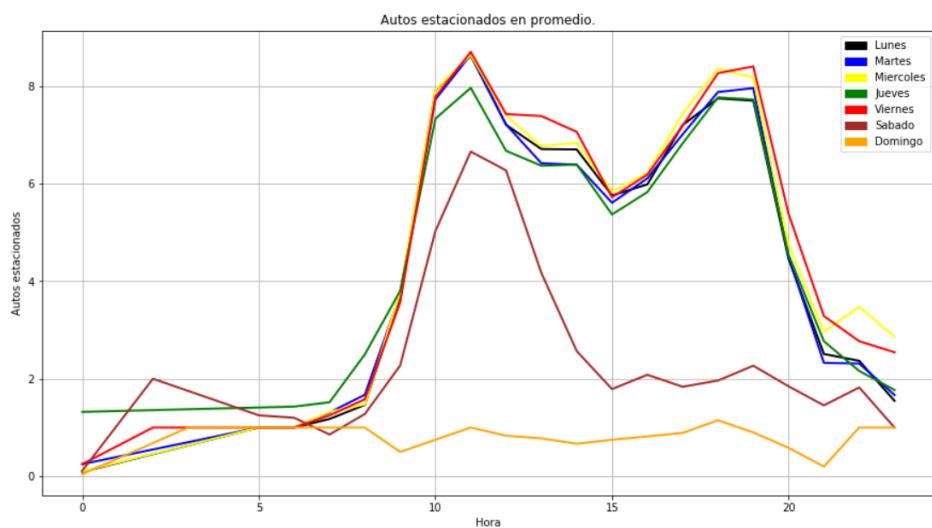
#### 4.2.9. Estacionamientos por día de semana

Se aprecia cierta diferencia de estacionamientos durante los días de semana (lunes a viernes) y el fin de semana (sábado y domingo) (figura 4.16) (figura 4.17) (figura 4.18). En los primeros, la cantidad promedio de autos estacionados es similar, observando valores más elevados los días viernes (al igual que en los gráficos de cantidad de operaciones por día de semana). Los sábados disminuye luego del mediodía y, los domingos el movimiento es casi nulo.

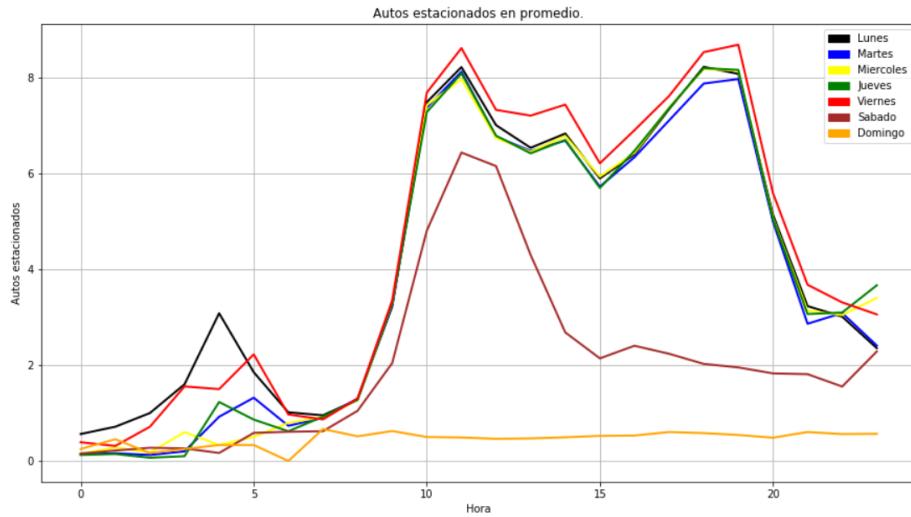
Los tres datasets poseen una media similar por hora y una desviación estándar elevada, aproximándose a  $\frac{1}{4}$  del valor de ocupación en esa hora. Se debe tener en cuenta que, teniendo una desviación tan alta, la cantidad de autos estacionados varía notablemente día a día, encontrando días con mucha ocupación y otros, con muy poca.



**Figura 4.16:** Ocupación por hora en promedio según cada día de la semana, dataset 2018.



**Figura 4.17:** Ocupación por hora en promedio según cada día de la semana, dataset web scraping.



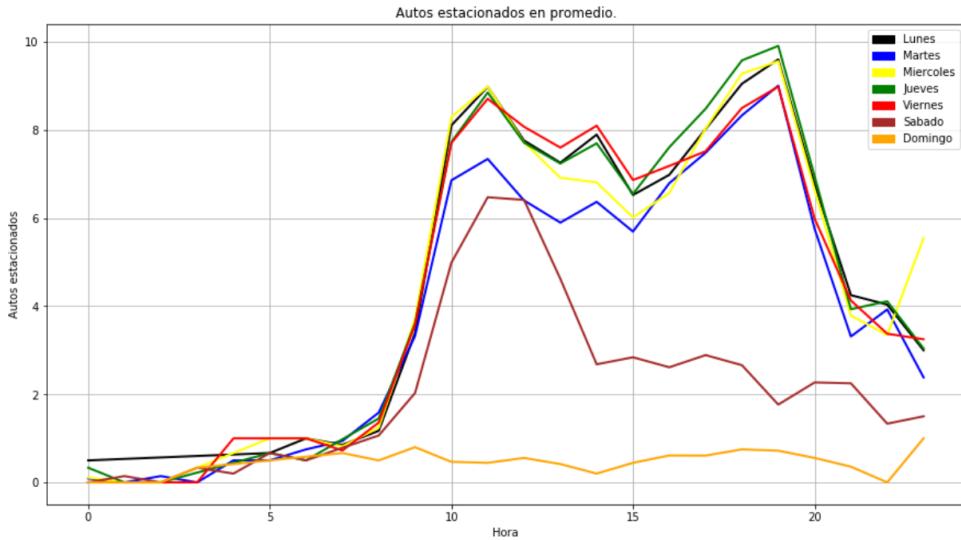
**Figura 4.18:** Ocupación por hora en promedio según cada día de la semana, dataset 2019.

Para cada mes, se graficó la ocupación dividida por día de semana con el fin de visualizar las diferencias de estos últimos de una forma más detallada.

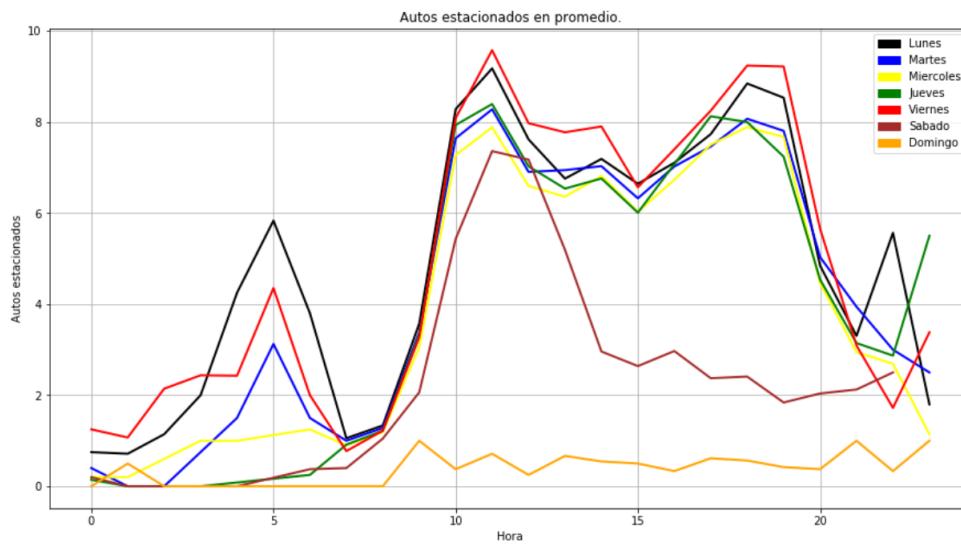
Se decidió mostrar las gráficas de los meses de Junio (figura 4.20) y Diciembre (figura 4.19) ya que se buscaba plasmar las diferencias encontradas en los días de la semana teniendo en cuenta distintos meses del año. El objetivo es demostrar las diferencias que pueden encontrarse entre los días de semana según el mes.

Si tomamos como ejemplo los días viernes a las 11 am, podemos observar que en Diciembre se visualiza 8.5 de tránsito mientras que en Junio, 9.5, por lo que podemos concluir, como se dijo anteriormente, en que existen diferencias en los patrones de estacionamiento por día de semana, aunque estos se vean similares, la ocupación de cada uno es diferente.

A partir de estas gráficas, se deduce la importancia de utilizar los atributos días de semana y mes para la implementación de los predictores.



**Figura 4.19:** Ocupación por hora en promedio según cada día de la semana, mes de Diciembre, dataset 2019.



**Figura 4.20:** Ocupación por hora en promedio según cada día de la semana, mes de Junio, dataset 2019.

#### 4.2.10. Mapas de calor

Para el análisis de la ocupación por cuadra de la ciudad de Tandil se utilizó el dataset 2019, ya que es el más completo y mantiene los mismos patrones de estacionamiento que los demás datasets. Para buscar una solución al problema de encontrar estacionamiento libre, se debe hacer hincapié en los momentos de mayor ocupación. Por esta razón se decidió realizar el mapa de calor de la ciudad a las 10 de la mañana, ya que es uno de los horarios más concurridos y con mayor cantidad de estacionados.

Para lograr una visualización más real de los casos críticos, se utilizaron únicamente los datos pertenecientes a días de la semana, descartando los relativos a fines de semana ya que el horario de funcionamiento de los parquímetros es restringido.

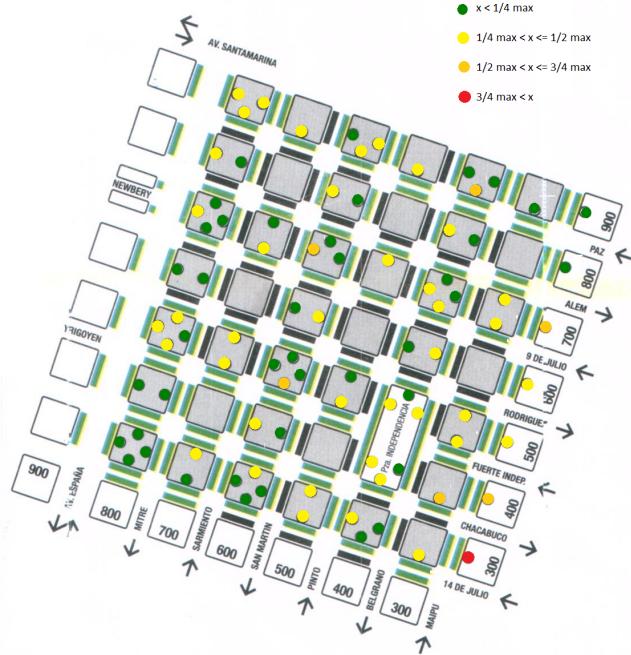
Se marcaron en cada cuadra la ocupación a las 10AM, con colores representativos del espacio libre, desde verde (valores menores a  $\frac{1}{4}$  del máximo de ocupación por cuadra) hasta rojo (valores mayores a  $\frac{3}{4}$  del máximo de ocupación por cuadra). Se comparó la media de ocupación de la cuadra en ese horario con el máximo (o límite superior) según corresponda.

En la figura 4.21 se utiliza como límite superior el máximo histórico encontrado por calle. Se observa que, en su mayoría, las calles están libres. Esto se debe a que la ocupación máxima encontrada representa los momentos en los que hubo picos elevados o autos fantasma. Por esta razón estos valores no son significantes para el análisis y se deben evitar.

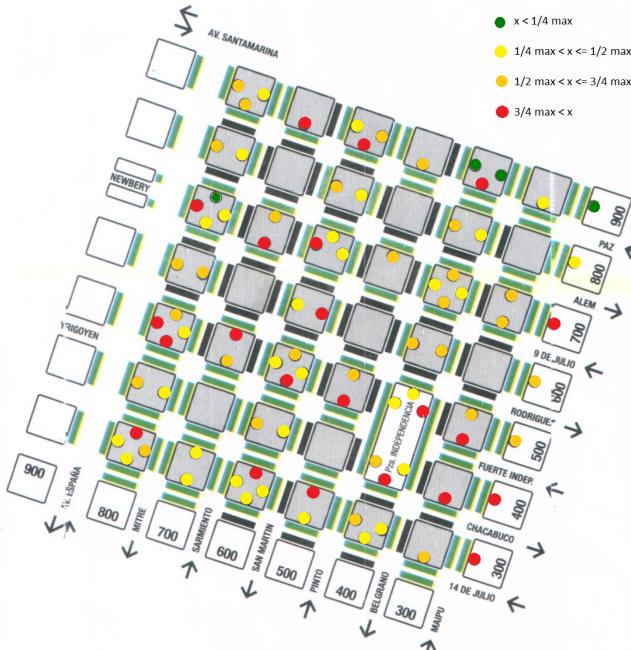
En la figura 4.22 se emplea como límite superior el máximo valor del boxplot, excluyendo los outliers. Se optó por esta opción ya que permite, con cierto grado de seguridad, obtener el valores cercanos al máximo verdadero que posee cada cuadra.

Como los conductores pueden variar el espacio ocupado al estacionar no es posible conocer la ocupación real por calle. Es por ello que se realizaron estas aproximaciones.

Analizando el mapa de calor de la figura 4.22, se observa que la mayoría de las calles poseen un porcentaje elevado de ocupación exceptuando las cercanas al final del estacionamiento medido en 14 de Julio. Se encuentran mayores lugares libres de estacionamiento en la periferia de la zona medida ya que los usuarios del sistema prefieren alejarse de su destino en lugar de abonar la tarifa.



**Figura 4.21:** Mapa de calor del dataset 2019 utilizando como máximo el máximo histórico por calle. Se eliminaron los valores correspondientes a fines de semana para evitar grandes modificaciones en los valores críticos.



**Figura 4.22:** Mapa de calor del dataset 2019 utilizando como máximo el promedio de máximos por día en cada calle. Se eliminaron los valores correspondientes a fines de semana para evitar grandes modificaciones en los valores críticos.

#### **4.2.11. Conclusión**

Se comenzó el análisis del trabajo con el dataset del 2018. Al contar con poca información y desactualizada se implementó un sistema para obtener datos periódicamente (web scraping). Como este procedimiento fallaba y se perdía cierta información, se buscaron nuevas opciones. A inicios del 2020, la oficina de SUMO proporcionó gran cantidad de datos actualizados. Como la información proviene de la misma fuente, los resultados de los tres datasets guardan cierta similitud.

A partir del análisis realizado, se comprobó la utilidad de ciertos atributos y la importancia de modificar algunos datos para obtener mejores resultados al entrenar los modelos.

Se pudo observar como atributos relevantes el tiempo en el que sucedió la transacción; la ocupación, que indica la cantidad de vehículos estacionados; el día de la semana junto con el mes ya que, como se analizó, se encuentran diferencias en la ocupación según estos atributos; el nivel o porcentaje de ocupación de una calle, parámetro utilizado para dibujar los mapas de calor con diferentes colores dependiendo cada cuadra, permitiendo conocer que tan ocupada está en un rango de 0 a 4.

También se comprobó que ciertos datos de los datasets no son relevantes. En el caso de los fines de semana, el horario de funcionamiento del parquímetro difiere de los días de semana. Por esta razón, los valores poseen distintos patrones de estacionamiento a lo largo del día y, a su vez, no alcanzan valores elevados, por lo que no es un caso crítico de estudio.

Comparando entre diferentes días se comprobó que los valores de estacionamiento cambian notablemente. Aunque las gráficas resultan similares, los valores de ocupación son diferentes en cada caso, teniendo días con alta y baja ocupación. Con valores tan dispares, las predicciones realizadas difícilmente serán certeras. Sumado a esto, al no tener acceso a los parámetros relacionados al clima y fechas festivas, no fueron contemplados pero serían de gran ayuda para realizar predicciones.

## **Capítulo 5: Predictores**

Una vez finalizado el análisis de datos, se comenzó a planear y desarrollar los modelos de predicción, los cuales contribuirán en la búsqueda de lugares libres de estacionamiento.

Como primera aproximación se buscaron problemas que podrían surgir al realizar los modelos. A continuación, se llevó a cabo un pre procesamiento de los datos, para que satisfagan las restricciones planteadas. Por último, se implementaron varios modelos de ML, tanto de clasificación como de regresión, con el fin de estudiar las ventajas y desventajas de cada uno y los beneficios que pueden ofrecer para resolver este tipo de problemas.

Para realizar el preprocesamiento, se optó por utilizar la librería pandas la cual, además de permitir el manejo de grandes volúmenes de datos en estructuras llamadas DataFrame, posibilita realizar tanto operaciones básicas (sumar, restar, entre otros) como complejas (agrupar por atributo, unir DataFrames, etc). A su vez, proporciona las herramientas para el manejo integrado de datos faltantes, la reestructuración y segmentación de conjuntos de datos, tanto la segmentación vertical basada en etiquetas como la horizontal de grandes conjuntos de datos. A su vez, permite la inserción y eliminación de columnas en estructuras de datos.

Para el desarrollo de los modelos de machine learning se emplearon dos librerías, scikit-learn y Tensorflow, las cuales facilitan su implementación. La primera provee diversos algoritmos de clasificación y regresión, como por ejemplo árboles de decisión y random forest. Además, provee gran cantidad de métricas y herramientas para comprobar qué tan buenas son las predicciones obtenidas. Tensorflow, por su parte, permite implementar modelos de redes neuronales complejos en pocas líneas de código, soportando tanto redes neuronales con capas densas como redes neuronales recurrentes. Para el desarrollo de los modelos se utilizó el módulo keras, implementado dentro de Tensorflow.

### **5.1. Problemas iniciales**

A partir del análisis realizado sobre el conjunto de datos y de plantear posibles modelos, junto con sus respectivas entradas y salidas, se planteó como solución tentativa realizar un clasificador que prediga cuál será la operación siguiente (entrada o salida de vehículo). Esta opción se descartó rápidamente ya que uno de los principales problemas encontrados fue conocer en qué tiempo suceden las predicciones. Este problema deriva de no tener los datos separados por períodos de tiempos fijos.

Como segunda solución se propuso un clasificador que prediga la operación y el tiempo de la siguiente operación. Esta opción también fue descartada ya que habría que tener en cuenta dos salidas, las cuales tendrían que ser optimizadas, trayendo como consecuencia un problema aún más complejo de resolver.

Estas problemáticas derivaron en la búsqueda de diferentes formas de plantear los datos y, a su vez, fue necesario pensar qué tipos de salida se esperan de dichos modelos. Se abordaron

dos posibles soluciones. En la primera, se buscó predecir cuál será la ocupación en el siguiente período de tiempo (problema de regresión) en base a la entrada. En la segunda, el objetivo fue predecir en qué nivel o porcentaje de ocupación se encontrará la calle en el siguiente período (problema de clasificación).

## 5.2. Pre-procesamiento de los datos

A partir del análisis realizado se deduce que los días sábados y domingos no son relevantes para las predicciones y podrían generar problemas a la hora de obtener resultados. Por esta razón, para armar los conjuntos de entrenamiento y prueba, se decidió eliminar dichos datos del dataset antes del procesamiento. A su vez, se descartaron los outliers de cada cuadra. Estos valores pueden no ser considerados errores y sería de gran ayuda predecir cuándo sucederán, aunque también pueden ser vehículos estacionados en una cuadra y fichados en otra. Al ser valores muy elevados, a diferencia del resto de los datos de cada calle, la inserción de los mismos en nuestro sistema introduciría ruido. Como estos valores equivalen a un reducido porcentaje, es sencillo predecir cuándo ocurrirán ya que los momentos de mayor estacionamiento se producen en fechas específicas.

Con respecto a los tiempos entre operaciones se encontraron diferencias de intervalos entre los datos, es decir, no hay un tiempo fijo entre operaciones. Este tipo de problemas es altamente complejo de resolver (se encuentran realizados pocos estudios sobre este tipo de datos), por lo que se propuso como primera solución la utilización del mayor valor por hora de cada día, aunque luego de evaluar lo que esto conlleva se concluyó que recorrer las distancias teniendo en cuenta el perímetro de la zona de estacionamiento medido no conlleva tiempos muy prolongados. Por lo tanto, esta opción fue descartada ya que las predicciones no serían acordes a la realidad del problema planteado, produciéndose valores para una hora después de la consulta. Agrupar los datos y forzar intervalos de tiempo, aunque no es óptimo para analizar este tipo de datasets [34], se realizó ya que hay más investigación realizada y más algoritmos desarrollados.

Se decidió utilizar el mayor valor cada media hora (valor que se ajusta a las distancias de la ciudad) ya que de otra forma no se podría conocer el momento en que se producen las salidas de los predictores desarrollados. Esto, como se dijo anteriormente, no es óptimo, ya que se pierde información de los datos y, en su lugar, se utiliza un nuevo valor el cual es un dato que debe representar todo el agrupamiento de los datos realizados. A pesar de esto, se vio como la única alternativa para poder desarrollar y probar soluciones al problema.

Para poder desarrollar modelos de machine learning es necesario establecer ciertos valores de entrada. En este caso se utilizaron id\_cuadra, dia de la semana, mes, tiempo, ocupación y ocupación anterior (la correspondiente a 30 minutos antes del tiempo de la ocupación actual). Los datos, a su vez, se clasifican en categóricos y numéricos. El primero engloba las variables id\_cuadra, dia de la semana y mes. El segundo, por su parte, tiempo, ocupación y ocupación

anterior. Esta división es necesaria ya que dependiendo el modelo elegido, se trabaja con un tipo de datos específico (algunos no aceptan los datos categóricos).

A su vez, antes de comenzar a trabajar con los modelos es necesario plantear las posibles salidas de los mismos. Al utilizar los modelos de regresión y clasificación fue necesario establecer dos escenarios. Para el primero, se decidió predecir la ocupación, ya que permitirá conocer los valores futuros de ocupación, los cuales podrán ser comparados con la ocupación máxima de la cuadra para establecer la cantidad de lugares disponibles o posibles. Para el segundo, se creó un nuevo atributo, el cual hace referencia al porcentaje de ocupación, representado en valores del 0 al 4. Se optó por esta salida ya que, al agrupar los datos en intervalos de treinta minutos, no es posible conocer la operación siguiente, ya que este valor no existe.

El objetivo principal es predecir lugares disponibles para estacionar. Utilizando el atributo porcentaje de ocupación es posible, a su vez, obtener el nivel de ocupación en un momento dado. Este último se creó tomando como base los valores de los mapas de calor. El rango se encuentra entre 0, representando la menor cantidad de vehículos estacionados, y 4, representando la mayor cantidad de vehículos estacionados, sin tener en cuenta los outliers. Se optó por este rango ya que la ocupación, en su mayoría, no suele ser elevada. En los casos de mayor amplitud de este valor, al incrementar el atributo porcentaje de ocupación, por ejemplo, de 0 a 9, el entrenamiento de los modelos se dificulta ya que habrá valores que representen pocos o hasta ningún dato, por lo que al entrenar los modelos podría haber clases escasas o nulas y generar así un dataset altamente desbalanceado.

Como producto final del pre procesamiento se obtuvo un dataset con 437.785 filas. Esta instancia fue fundamental ya que si se tomaba el dataset inicial (con mayor cantidad de datos) hubiese sido imposible entrenar los modelos debido a los problemas mencionados anteriormente. Para poder corroborar el correcto funcionamiento de la red con datos reales, se utilizó un 90% del dataset para entrenamiento y un 10% para testeo. Para esta división se empleó la estrategia “stratify”, la cual permite obtener una distribución equivalente de clases entre ambos conjuntos.

Se probó entrenar los modelos únicamente con los datos pertenecientes al rango horario de los parquímetro, de 10 a 20hs, aunque esto no modificó los resultados obtenidos, por lo que el análisis realizado a continuación es válido para ambos casos.

id_cuadra	mes	dia de la semana	tiempo	ocupacion	ocupacion anterior	porcentaje ocupación	target_ocup	target_porc_ocup
2	1	2	10.5	1	2	0	5	1

**Tabla 5.1:** Ejemplo de datos utilizados para el entrenamiento de los modelos. Algunos modelos utilizan sólo datos numéricos y otros utilizan también categóricos. target\_ocup es el output de los modelos de regresión, mientras que target\_porc\_ocup es el output de los modelos de clasificación.

### 5.3. Regresores

El objetivo de realizar modelos es regresión es predecir cuál será la ocupación en el siguiente período de tiempo. Se decidió comenzar por buscar esta solución ya que los resultados podrían brindar información más detallada, es decir, el valor de ocupación que se tendrá en cada cuadra. Para ello se plantearon varios modelos, desde una simple regresión lineal hasta redes neuronales recurrentes.

Las métricas elegidas son RMSE y  $R^2$ . La primera permite conocer el error entre los valores, haciendo énfasis en errores grandes [18]. Esto es especialmente útil para resolver este problema, ya que se desean evitar confusiones al predecir ocupación que podrían llevar a predecir erróneamente espacios libres u ocupados (equivocación por uno o dos valores de ocupación no es igual de grave que por cinco o diez, por ejemplo). La segunda métrica da a conocer que tan bueno es nuestro predictor en una escala de -infinito hasta 1. Mientras más cercano a 1 se encuentre el valor, más certero será el predictor [19]. La función de pérdida utilizada para todos los modelos es MSE, ya que posee menor costo computacional que RMSE y, de igual modo, se desea evitar los errores grandes. En el caso del árbol de regresión y random forest, se utilizó como criterio de separación entre nodos la función MSE.

El optimizador utilizado para los modelos es Adam, exceptuando árbol de regresión y random forest, los cuales no utilizan optimizadores. Se decidió por este algoritmo ya que ofrece buenos resultados en la mayoría de los casos y se recomienda como el optimizador a utilizar por defecto. Los parámetros de este modelo no fueron modificados y se utilizaron tal cual lo recomiendan sus creadores en [22]. Además, en estos modelos se utilizó un batch de tamaño 64 ya que ofrece una velocidad de cómputo aceptable y posee una cantidad de datos suficiente para poder entrenar al modelo sobre un conjunto significante de información.

Se realizó un modelo base, el cual delimitará si los modelos entrenados predicen correctamente los valores [36]. Este modelo siempre predecirá el valor de ocupación anterior. Por ejemplo, si actualmente hay tres autos estacionados y se consulta la ocupación para dentro de treinta minutos, el resultado será tres.

El valor de la métrica  $R^2$  del modelo base es de 0.8 y un RMSE de 2.251. Estos valores muestran que la mayoría de la información en el conjunto de datos posee el mismo target con respecto al valor de entrada en ocupación. En algunos casos en los que la ocupación siguiente

no es igual a la anterior, se pudo observar una gran diferencia entre estos valores, indicando que en caso de existir una variación entre estos dos valores, probablemente sea grande.

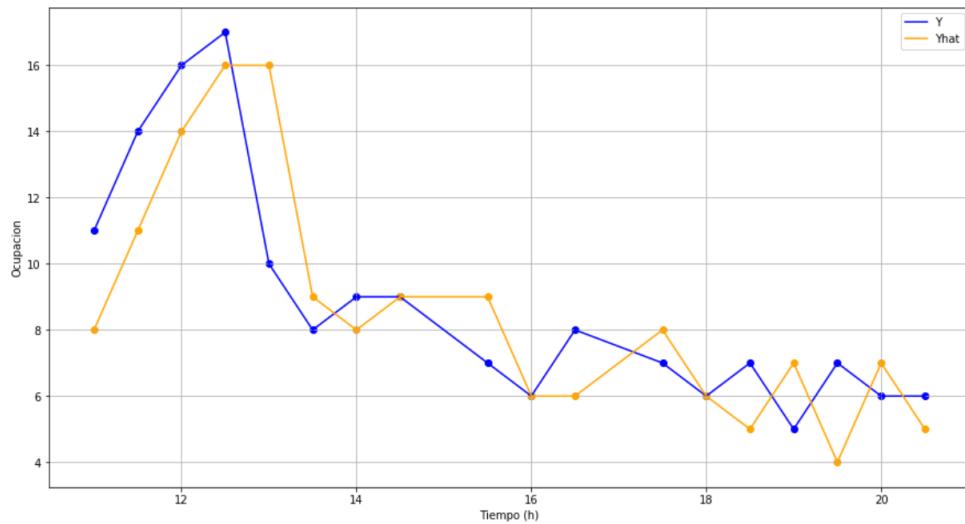
### 5.3.1. Modelos

- **Regresion lineal (valores de entrada numéricicos)**

Se implementó una regresión lineal que posee como valores de entrada el tiempo, la ocupación (t-1) y la ocupación anterior (t-2). Los datos utilizados son necesariamente numéricos ya que la regresión lineal como tal no acepta datos categóricos [30].

El modelo está conformado por un total de 4 parámetros entrenables. A partir del tercer epoch dejaron de verse mejoras en la función de pérdida. Esto se debe a que los pocos parámetros presentes en el modelo convergieron al valor que provee el menor valor de pérdida.

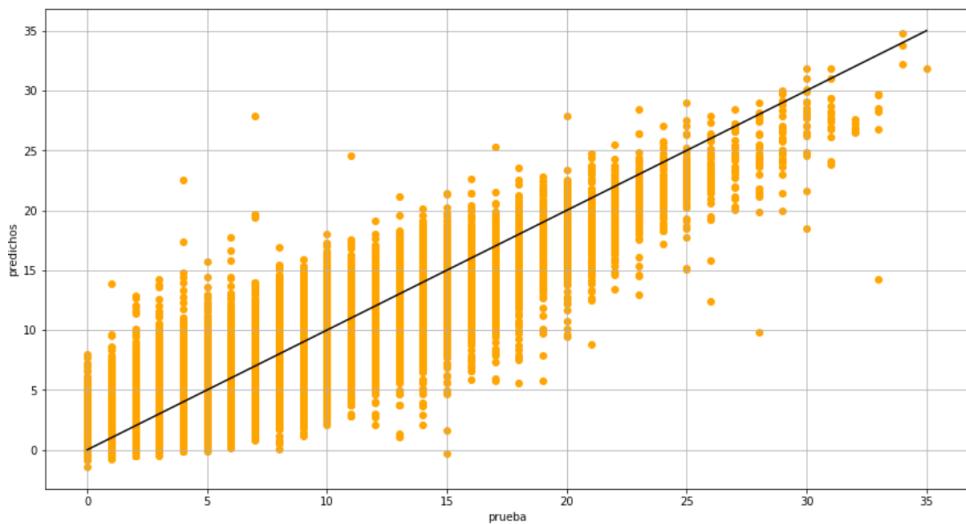
El valor de  $R^2$  obtenido es de 0.827 y, al observar el gráfico de un día de prueba, se verifica que el modelo tiende a predecir el mismo valor de entrada, con algunas excepciones (figura 5.1). El RMSE obtenido es de 2.134. Los valores obtenidos son similares a los obtenidos en el modelo base, por lo que este no proporciona grandes cambios.



**Figura 5.1:** Comparación entre valores reales (azul) y valores predichos por el modelo de regresión lineal con valores numéricos de un día en una calle.

Uno de los objetivos que tiene desarrollar un modelo tan simple es poder visualizar los pesos una vez entrenados. Para este caso, teniendo tres valores de entrada, se puede observar que se le asigna un valor de -0.104 al tiempo, 1.034 a la ocupación y -0.130 a la ocupación anterior. Como se podía suponer gracias al modelo de base, el valor más importante a tener en cuenta

es la ocupación. Este predictor la mayoría de las veces predecirá valores iguales o menores a la ocupación de entrada que posea.

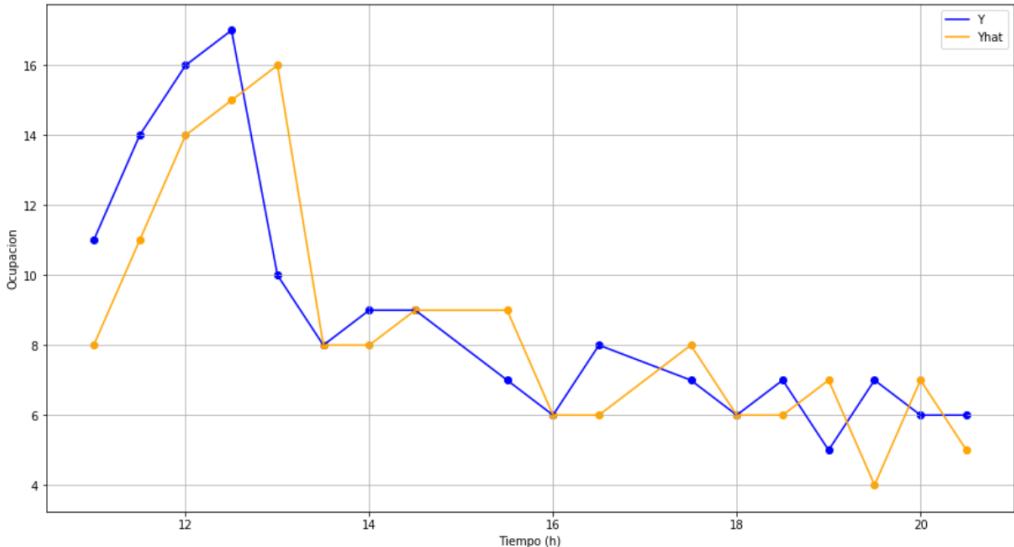


**Figura 5.2:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) para todo el conjunto de datos de prueba en un modelo de regresión lineal con valores numéricos.

#### - Regresion lineal (valores de entrada numéricos y categóricos)

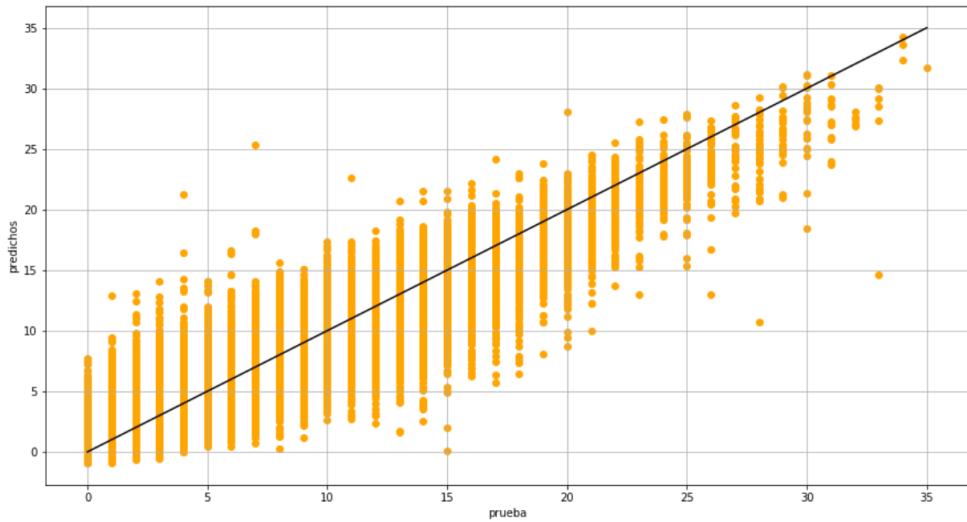
Se implementó un modelo similar al de regresión lineal con valores numéricos, agregando embeddings para incluir nuestros datos categóricos (id\_cuadra, mes, dia de la semana) como valores de entrada, ya que esto permite tenerlos expresados como vectores de valores numéricos [26], junto al tiempo, la ocupación y la ocupación anterior. Se probaron diferentes tamaños de embedding y se concluyó en utilizar un total de 10. Con valores más bajos no se obtenían resultados distantes, aunque sí menores, mientras que con valores más elevados el modelo empeoraba notablemente sus predicciones. La obtención de este valor se obtuvo a prueba y error.

Una vez entrenado, el modelo mostró mejores resultados que una regresión lineal entrada únicamente valores numéricos, con un  $R^2$  de 0.841 y un RMSE de 2.029. Aún así, no se ven grandes cambios al visualizar los datos predichos en los datos de validación (figura 5.3).



**Figura 5.3:** Comparación entre valores reales (azul) y valores predichos por el modelo de regresión lineal con valores numéricos y categóricos de un día en una calle.

Se puede observar que sobre las predicciones realizadas en los datos de entrenamiento tampoco hay un gran cambio, comparado con el modelo anterior (figura 5.4).



**Figura 5.4:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) para todo el conjunto de datos de prueba en un modelo de regresión lineal con valores numéricos y categóricos.

Los resultados de este modelo no son tan sencillos de comprender ya que posee una cantidad mucho mayor de parámetros y estos a su vez no muestran información directa sobre nuestros datos (ya que se incluyeron embeddings), aunque puede observarse que al tener información extra sobre la calle y datos temporales mejora la performance del modelo. Se probó implementar un modelo similar pero únicamente añadiendo como valor categórico `id_cuadra`, aunque se obtuvieron peores resultados que los presentados en este apartado.

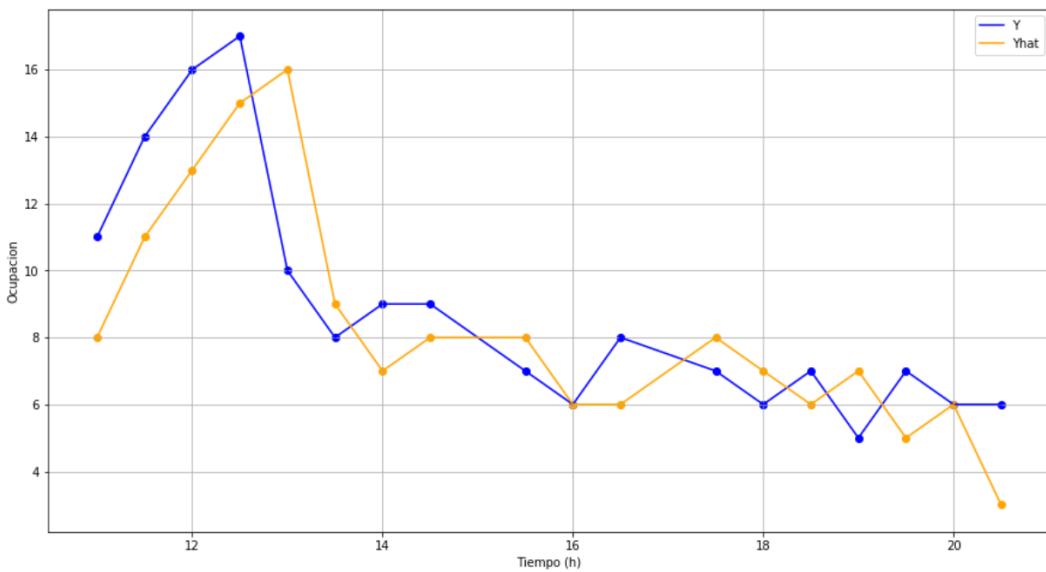
### - Árbol de regresión (dataset completo)

Los árboles de regresión son otro tipo de modelo que no soporta variables categóricas como entrada, por lo que se utilizó el tiempo, la ocupación y la ocupación anterior. No se podría utilizar embeddings para reemplazar los valores categóricos ya que los mismos necesitan ser entrenados, y, como se vio en el marco teórico, los árboles no modifican parámetros sino que el aprendizaje que realizan se ve representado en su propia construcción.

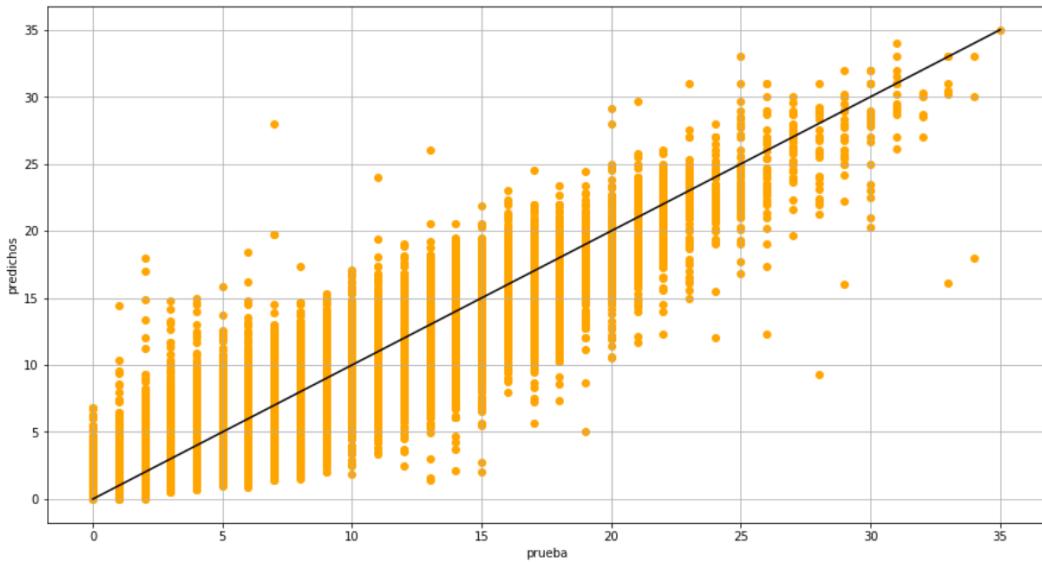
Se hicieron pruebas con diferentes profundidades permitidas de árbol, utilizando profundidades desde 1 hasta 20. Esto se realizó con el fin de encontrar el modelo que mejor se adapta al conjunto de prueba planteado. En base a estas pruebas se obtuvo que el mejor valor obtenido es con una profundidad de árbol de 11, poseyendo un total de 1150 nodos hoja.

Lo interesante de realizar este tipo de modelos es que se puede visualizar directamente por qué es elegido un resultado, aunque en este caso al ser un árbol tan grande, solo se puede revisar algún caso específico, siendo tedioso y complejo revisar cada decisión tomada por el modelo.

Con este modelo se obtuvieron valores un ligeramente elevados más de la métrica  $R^2$ , obteniendo un total de 0.859. Aún se puede observar que la mayor cantidad de veces predice valores iguales a la ocupación de entrada (figura 5.5), o a lo sumo menores, sin lograr buenos resultados. El RMSE obtenido es de 1.945, siendo el más bajo hasta el momento y, por lo tanto, el mejor valor obtenido.



**Figura 5.5:** Comparación entre valores reales (azul) y valores predichos por el modelo, entrenado con todo el conjunto de datos, de árboles de regresión de un día en una calle.



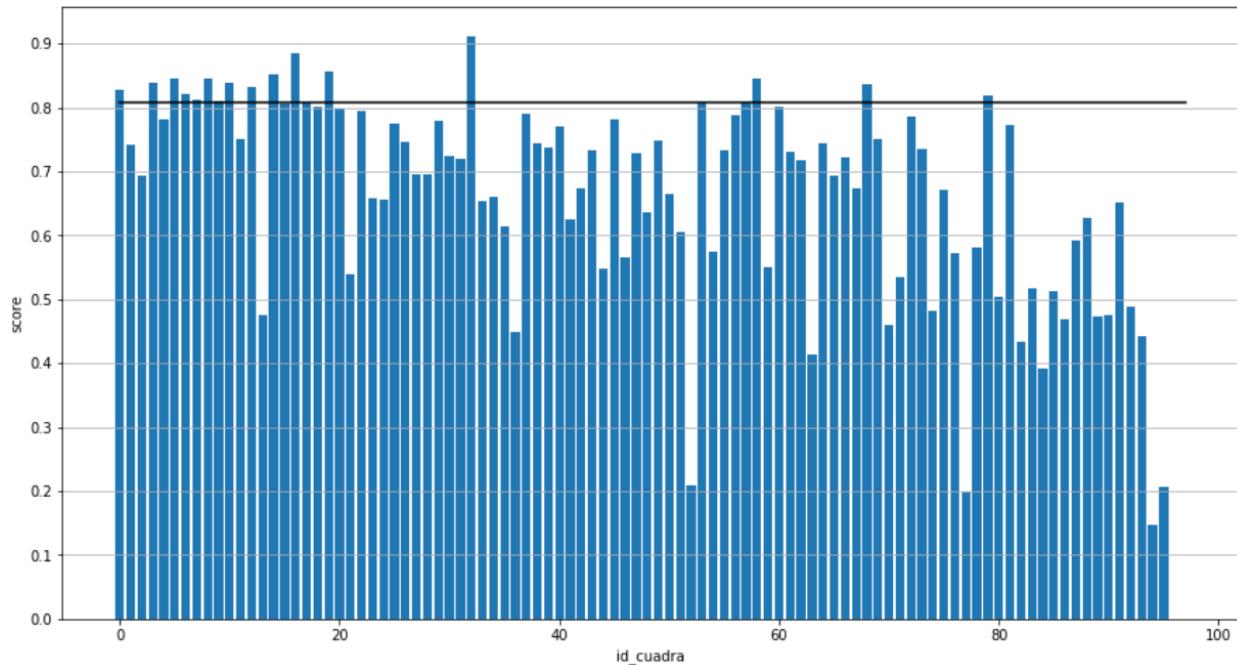
**Figura 5.6:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) para todo el conjunto de datos de prueba en un modelo de árboles de regresión.

#### - Árbol de regresión (un modelo por cuadra)

Además de modelarse un árbol de regresión con variables únicamente numéricas, se tomó en cuenta el dato categórico que más información aporta: id\_cuadra. Para introducir información de esta variable se optó por realizar n modelos diferentes, donde cada modelo fue entrenado con los datos de las n cuadras que poseen parquímetro. Por cada modelo creado, se realizó una búsqueda de mejor resultado obtenido por profundidad de árbol.

La principal desventaja de esta solución es la poca disponibilidad de datos en algunas cuadras, por lo que podría llevar a que el sistema no aprenda correctamente los patrones de estacionamiento correspondientes. Esto se observa con los resultados de las métricas  $R^2$  y RMSE, las cuales fueron calculadas como un promedio de su valor en todos los modelos. Respectivamente, se obtuvo un valor de 0.641 para  $R^2$  y un valor de 6.026 para RMSE. Estos valores están muy por debajo de los valores obtenidos por nuestro modelo base, por lo que podríamos decir que los modelos obtenidos no son útiles para resolver nuestro problema.

Si se inspeccionan más al detalle los resultados obtenidos (figura 5.7) se observa que hay cuadras en las que se obtuvo valores de  $R^2$  muy por debajo de nuestro base, así como también hay valores que se encuentran por encima, llegando hasta 0.89 en el mejor caso. Estos modelos brindan buenos resultados, aunque al ser poca la cantidad de cuadras, esta solución, en general, no brinda buenos resultados.

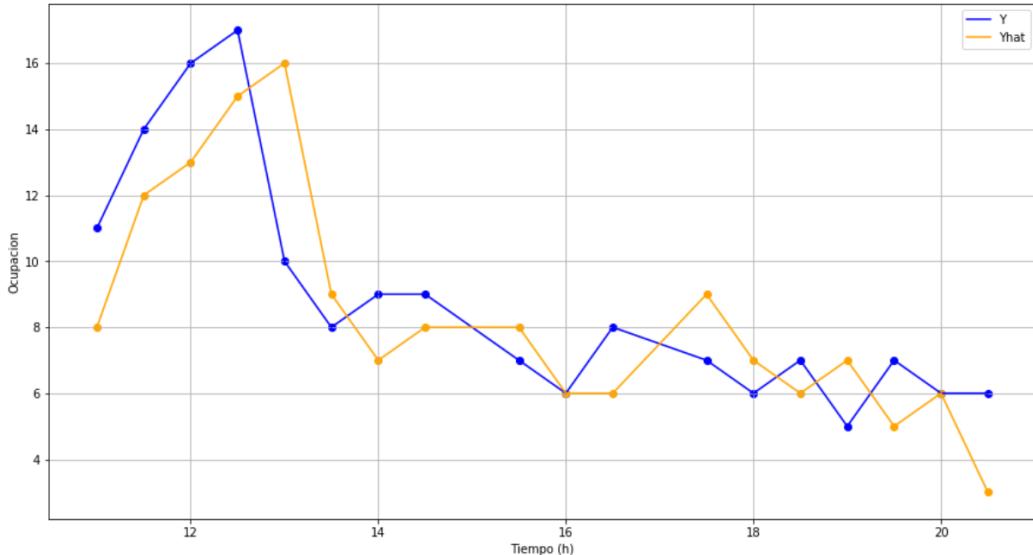


**Figura 5.7:** Gráfico con los valores de  $R^2$  obtenidos en cada modelo de árboles de decisión por cuadra. La línea horizontal negra representa el valor de  $R^2$  obtenido por el modelo base.

#### - Random forest para regresión

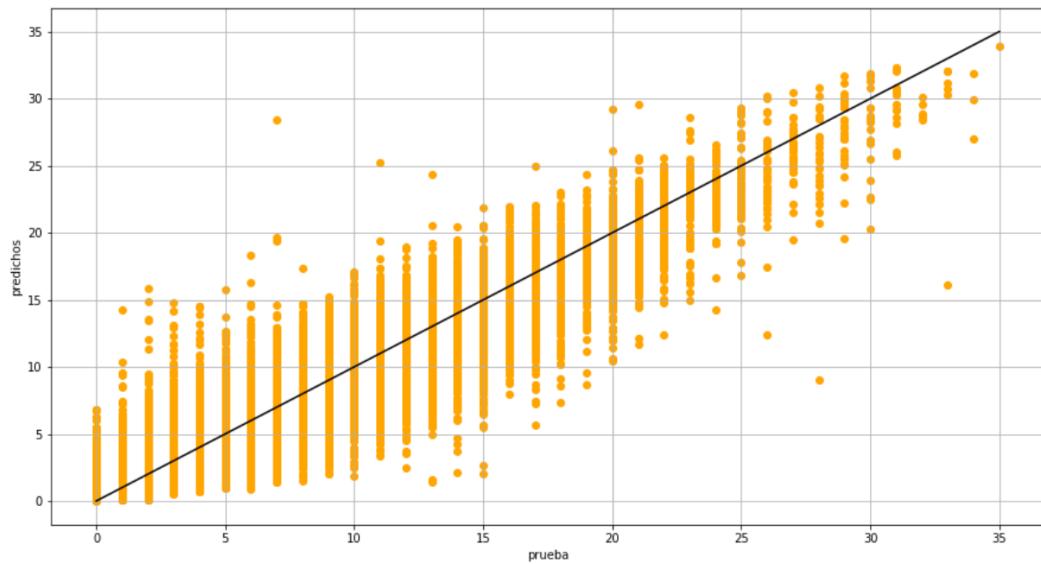
Como se comentó en el marco teórico, estos modelos son, en principio, un conjunto de árboles de regresión, donde cada uno predice un resultado y luego se promedian para obtener el valor total predicho. Cada uno de estos árboles es creado con un subconjunto de los datos de entrenamiento originales y el resultado obtenido es el promedio de los resultados obtenidos por cada árbol. Se utilizaron un total de 100 árboles con los mismos hiper parámetros que los árboles de regresión planteados en los puntos anteriores. Se obtuvo que la profundidad que arroja los mejores resultados es de 12.

Este modelo performo ligeramente mejor que el modelo de árboles de regresión, obteniendo el  $R^2$  más elevado hasta el momento, con un valor de 0.863. Así mismo, se obtuvo un RMSE de 1.913. Estos valores no denotan una gran mejoría con respecto a los anteriores modelos planteados, como puede observarse en la figura 5.8.



**Figura 5.8:** Comparación entre valores reales (azul) y valores predichos por el modelo de random forest para regresión de un día en una calle.

A pesar de que el modelo utilizado intenta cubrir las falencias de implementar un sólo árbol de regresión, se ve que los resultados obtenidos no mejoraron en gran medida. En la figura 5.9 se observa que el predictor logra acertar más veces mientras mayor es el valor a predecir, a comparación de un solo árbol de regresión.



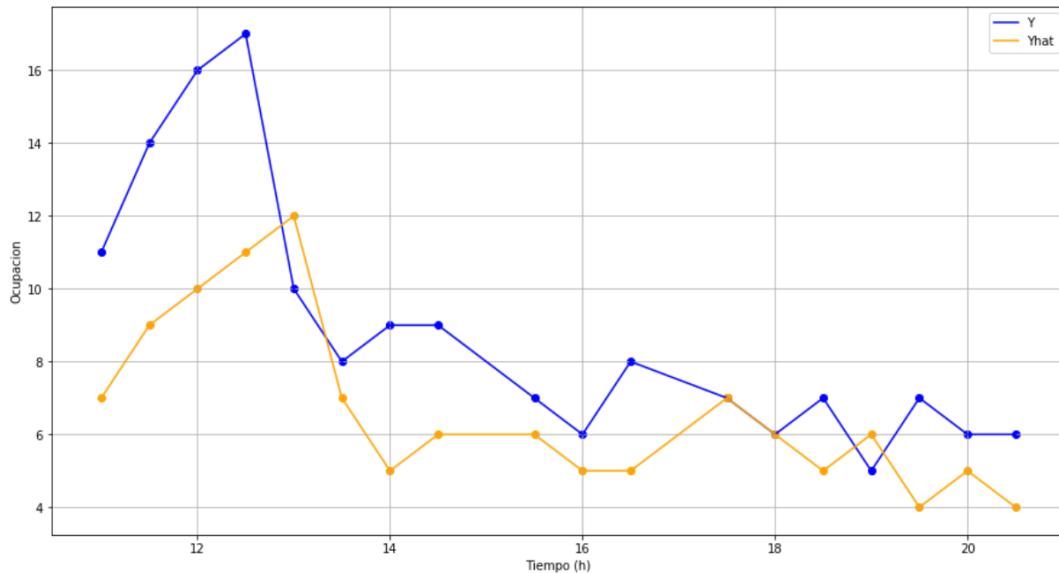
**Figura 5.9:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) en un modelo de random forest para regresión.

### - Red Neuronal Artificial (valores de entrada numéricos)

Luego de probar modelos sencillos de implementar, se continuó por modelos de redes neuronales artificiales. Estos se compone únicamente de capas densas, por lo que recibe, al igual que los anteriores modelos, valores numéricos. Comenzando con una capa de entrada que recibe el tiempo, la ocupación y la ocupación anterior, posee una capa densa con 64 neuronas, luego una capa de dropout con un 30% de chance y finaliza con otra capa densa de 64 neuronas. Ambas capas densas poseen como función de activación la función ReLU. Se eligió esta función de activación ya que realiza una cantidad reducida de cálculos y permite a los modelos aprender más rápido [14]. Estas capas se conectan a una capa final, la cual será la encargada de brindar la predicción final del modelo.

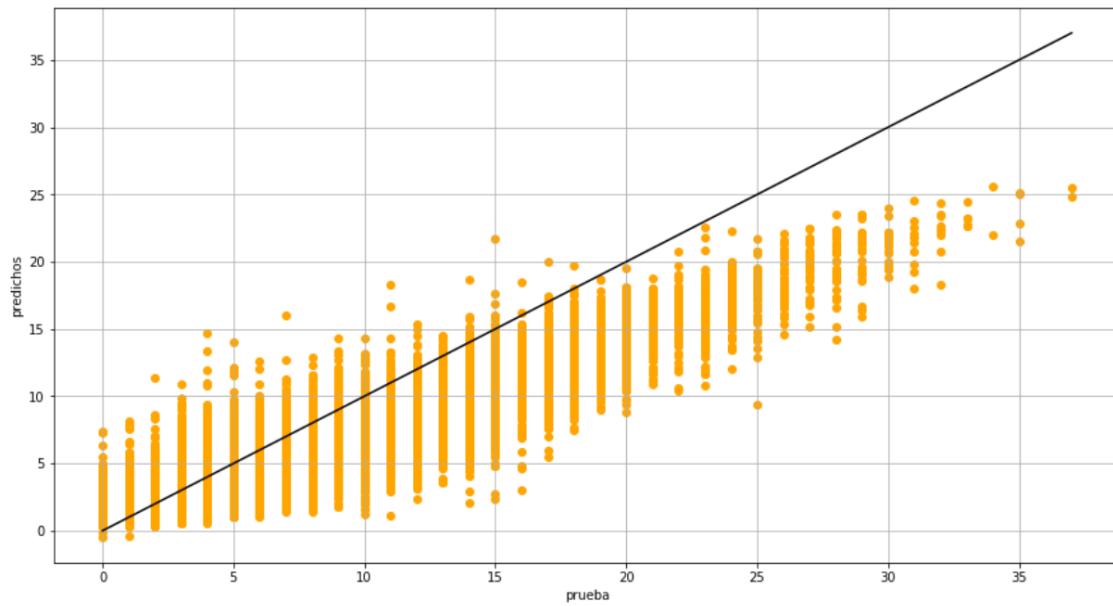
Las capas ocultas poseen 64 neuronas cada una. La elección de la cantidad fue con la intención de obtener un modelo que no posea tanto costo computacional con el fin de ser entrenado rápidamente, además, se obtuvieron mejores resultados que con otras configuraciones. La capa de dropout fue introducida para evitar el overfitting.

El valor de la métrica  $R^2$  obtenido es de 0.745, mucho menor a los valores obtenidos anteriormente. El valor de RMSE obtenido es de 2.559, el más alto hasta el momento. Esto se debe a que la red, al tener más capas, y más nodos, puede reconocer mejor los patrones de estacionamiento del conjunto de datos, aunque, en este caso, en vez de reconocer patrones de estacionamiento, opta por elegir que la ocupación futura decrece a comparación de la comparación futura (underfitting). Como puede observarse en imagen, la línea de predicciones posee valores mucho menores que la línea de valores reales.



**Figura 5.10:** Comparación entre valores reales (azul) y valores predichos por el modelo de redes neuronales profundas, con entradas numéricas, de un día en una calle.

Para comprobar lo analizado, se puede observar también en la siguiente imagen que los valores predichos tienden a ser inferiores a los valores reales.

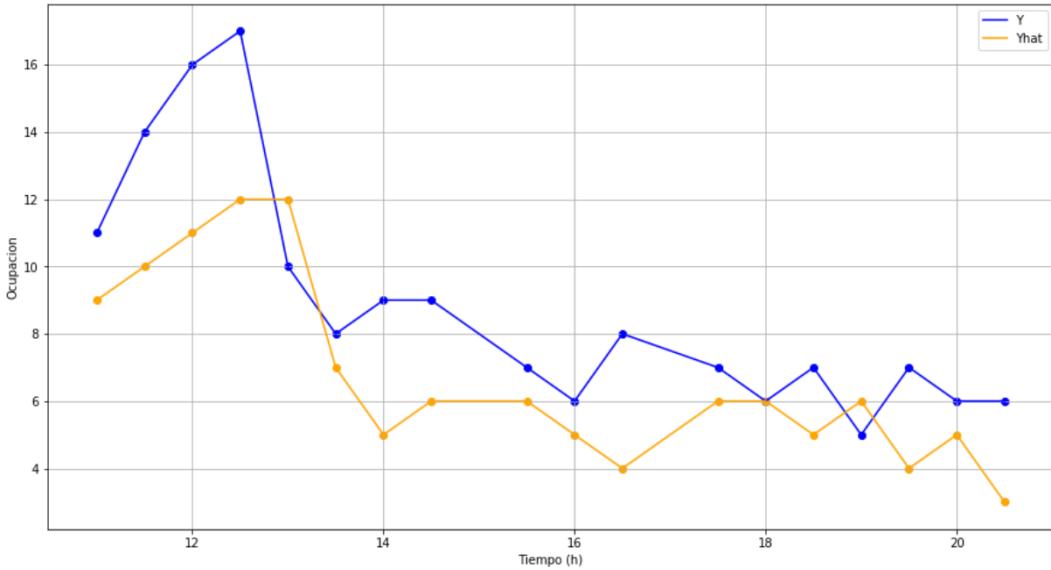


**Figura 5.11:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) para un modelo de redes neuronales profundas, con entradas numéricas.

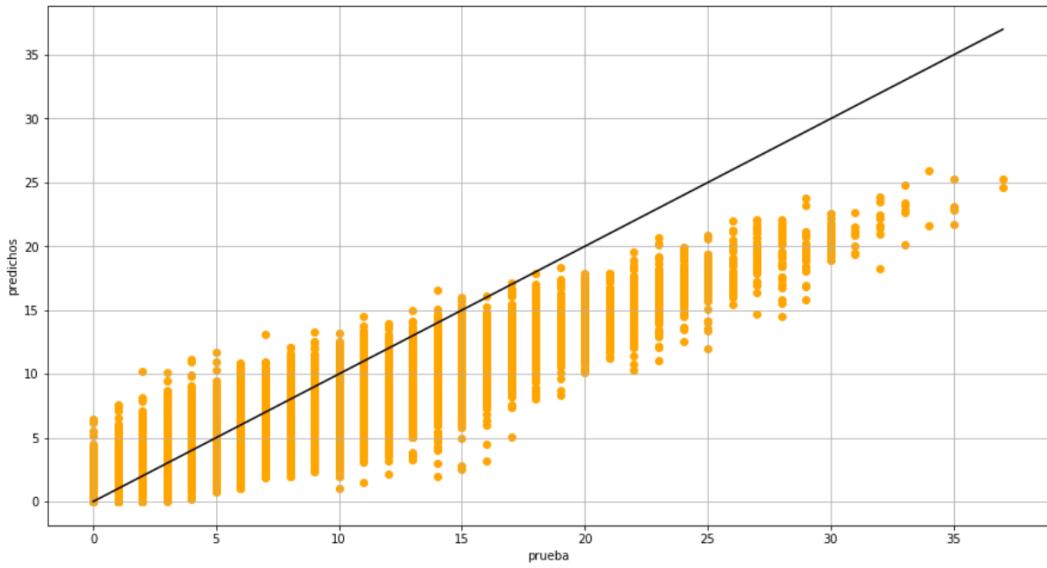
#### - Red Neuronal Artificial (valores de entrada numéricicos y categóricos)

Se implementó un modelo de red neuronal, similar al anterior, aunque en este caso se optó por incluir como valores de entrada a las variables categóricas. La misma cantidad de capas ocultas que en el caso anterior fueron usadas. Cada embedding posee 10 nodos, al igual que en el caso de la regresión multinomial con valores de entrada numéricos y categóricos.

Este modelo arrojó mejores resultados que el implementado sin datos categóricos, aunque aún se mantiene por debajo de nuestro modelo base, con un valor de  $R^2$  de 0.750 y RMSE de 2.534. Como puede verse en las imágenes, arroja resultados similares al modelo entrenado sin datos categóricos, pudiendo comprender que la cantidad de datos que se poseen es una limitante y por lo tanto, el tamaño y el tipo de red utilizados resulta demasiado grande.



**Figura 5.12:** Comparación entre valores reales (azul) y valores predichos por el modelo de redes neuronales profundas, con entradas numéricas y categóricas, de un día en una calle.



**Figura 5.13:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) para un modelo de redes neuronales profundas, con entradas numéricas y categóricas.

#### - Redes recurrentes (LSTM)

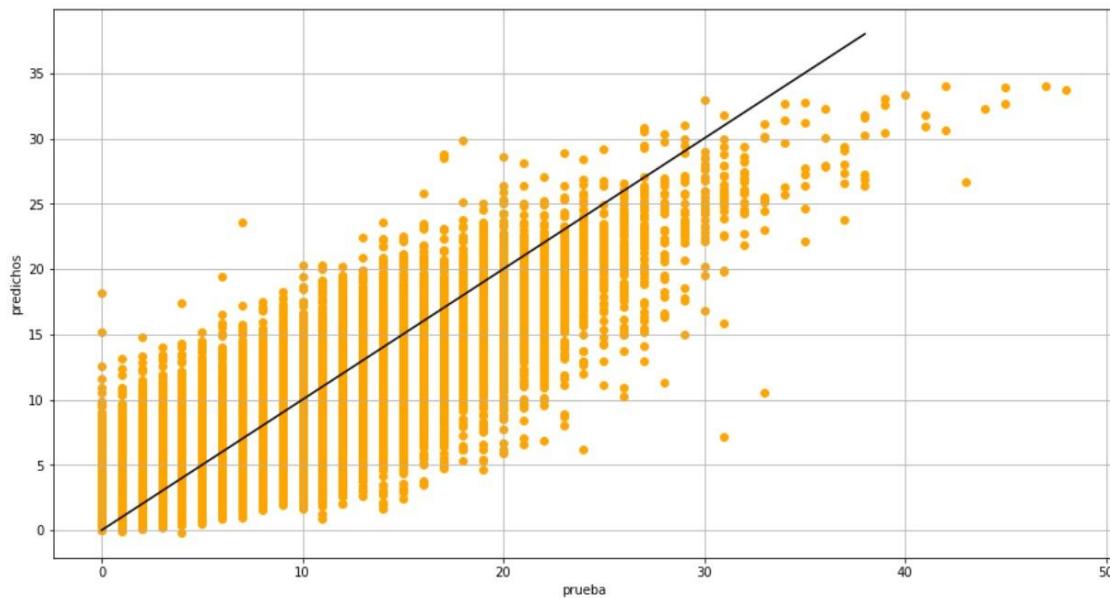
Se planteó como una solución al problema implementar redes recurrentes con LSTM, las cuales están pensadas para poder recordar sucesos pasados. Se utilizó una ventana de valores de entrenamiento de ocupación de tres, es decir, se brindó información al modelo para que tenga en cuenta sucesos ocurridos hasta una hora y media antes del suceso a predecir

(tres datos pasados). Con esto fueron entrenados dos modelos, uno únicamente con ocupación y otro con datos numéricos y categóricos.

La cantidad de embeddings utilizadas en el modelo de red recurrente con variables numéricas y categóricas es de 10, al igual que en los anteriores modelos que se utilizaron variables categóricas.

Ambos modelos presentaron valores de  $R^2$  y RMSE por debajo del base, con 0.700 y 2.825 respectivamente para el modelo que utiliza sólo la ocupación como entrada, mientras que el modelo que utiliza todas las features como entrada posee 0.733 y 2.814, respectivamente.

Este tipo de modelos no obtuvo buenos resultados, logrando valores muy dispersos. Como puede observar en la figura 5.14 estos valores obtenidos poseen una desviación muy grande, mucho mayor que en los otros modelos.



**Figura 5.14:** Comparación entre valores reales (eje-x) y valores predichos (eje-y) para un modelo de redes recurrentes.

### 5.3.2. Comparación entre modelos de regresión

Los modelos probados no arrojaron muy buenos resultados si se comparan con los resultados obtenidos con el modelo base. Con este último se obtuvo un valor de  $R^2$  de 0.8, lo que se podría considerar un valor elevado, aunque en este caso la información que brinda el modelo base es que hay muchos casos en donde la ocupación permanecerá igual, por lo que las predicciones tenderán a esto. Por ende los resultados deberían superar este valor de 0.8 para considerarse buenos.

En ciertos casos se pueden predecir correctamente algunos valores, siendo el random forest el que mejores resultados obtuvo. A pesar de esto, es apenas un 6.3% mejor que el modelo base y se puede observar que la mayoría de las veces opta por devolver el mismo valor de entrada.

También se observa que los modelos divididos por cuadra dieron el peor valor de  $R^2$  (que es el promedio del  $R^2$  de cada modelo), aunque para algunas cuadras dieron valores más elevados hasta llegar a casi 0.9, por lo que estos pueden ser de utilidad para predecir o corroborar predicciones de otros modelos en esas cuadras.

Los resultados obtenidos por este tipo de problemas no son óptimos, ya que a pesar de poseer valores altos y RMSE bajo, esto sucede porque hay muchos casos en los que la ocupación futura es igual a la ocupación actual. Con una muy elevada precisión podemos decir que la ocupación en la siguiente media hora será igual a la ocupación en el momento actual. El problema de estas predicciones es que no brindan información relevante para predecir lugares libres de estacionamiento por cuadra. Se podría ver la ocupación actual y, en caso de ser elevada, se podría decir que la cuadra no tiene espacios libres, aunque no podríamos estar seguros.

Modelo	$R^2$	RMSE
Base	0.8	2.248
Regresión lineal (valores numéricos)	0.827	2.134
Regresión lineal(valores numéricos y categóricos)	0.841	2.029
Árbol de regresión entrenado con todo el dataset	0.859	1.945
Árbol de regresión con un modelo por cuadra	0.641	6.026
<b>Random forest para regresión</b>	<b>0.863</b>	<b>1.913</b>
Red neuronal artificial (valores numéricos)	0.745	2.559
Red neuronal artificial (valores numéricos y categóricos)	0.750	2.534
Redes recurrentes entrenado sólo ocupación	0.700	2.825
Redes recurrentes entrenado con todas las features	0.733	2.814

**Tabla 5.2:** resultados de las métricas ( $R^2$  y RMSE) obtenidos por cada modelo.

#### 5.4. Clasificadores

Al ya haber implementado y observado los resultados de varios modelos de regresión, se implementaron los modelos de clasificación. Se optó por utilizar los mismos modelos

anteriormente implementados, adaptandolos para que funcionen como clasificadores en vez de regresores.

En el caso de los modelos de clasificación, a diferencia de los de regresión, se busca predecir el nivel o porcentaje de ocupación 30 minutos después de los valores ingresados. A pesar de que se podría evaluar como un problema de regresión, se buscó conocer el porcentaje de ocupación de cada calle, siendo planteados como clases (Rodriguez Alvarez J utiliza y explica, en [35], un clasificador utilizando valores numéricos, el cual obtuvo buenos resultados).

Para la implementación de los modelos y elección de métricas se tuvo en cuenta el desbalance de clases presente en los datos. Como se observa en la tabla 5.3 la mayor cantidad de datos hacen referencia al nivel de ocupación 1, mientras que la menor cantidad de datos la posee el nivel de ocupación 4, siendo este el valor al que más importancia le daremos (valor crítico), ya que nos permitirá conocer si una cuadra tiene espacio libre para estacionar o no.

Porcentaje de ocupación	cantidad de tuplas
0	151448
1	171976
2	90890
3	28247
4	5905

**Tabla 5.3:** cantidad de tuplas pertenecientes a cada clase luego del pre procesamiento.

Una manera de solucionar el desbalance de clases es calcular que peso poseen tomando en cuenta únicamente los datos utilizados para entrenamiento. El objetivo de esto es que las clases que poseen menor cantidad de datos afecten en mayor medida la pérdida resultante del modelo, ocasionando cambios más grandes en sus parámetros entrenables. El peso obtenido para cada clase se puede observar en la tabla 5.4.

Porcentaje de ocupación	peso
0	0.60008326
1	0.52031867
2	0.97521957
3	3.1306496
4	14.96270621

**Tabla 5.4:** cantidad de tuplas pertenecientes a cada clase luego del pre procesamiento. El valor obtenido en peso dará la posibilidad de ponderar los valores obtenidos y de este modo simular igualdad de clases.

Las métricas elegidas son f1-score y recall, ya que no se ven perjudicadas por el desbalance de clases [16]. De esta última métrica mayormente interesa para el nivel de ocupación con valor 4, el cual nos dirá si se encuentra o no libre la cuadra para estacionar. Ambas métricas fueron elegidas ya que la primera permite conocer qué tan bueno es el modelo en general, es decir, que tan bien predice las clases. La segunda, el recall sobre el nivel de ocupación 4, permite conocer que tan bien se predice sobre este valor. Un valor elevado de esta métrica demuestra que el predictor predice correctamente el valor 4 cuando debe hacerlo, es decir, si el valor real de salida es 4, nuestro predictor optará mayormente por este valor en lugar de los valores más bajos de nivel de ocupación.

El mejor modelo deberá tener un valor elevado en el balance entre los resultados de estas dos métricas, ya que, aunque buscamos un clasificador que pueda predecir correctamente todas las clases, priorizaremos el que tenga una alta capacidad predictiva junto a una buena capacidad para predecir nuestro valor crítico.

Tomando en cuenta únicamente el valor de recall con el valor crítico, podríamos tener el caso en el que la correcta salida posea un nivel de ocupación de 1, aunque el modelo únicamente predeciría nivel de ocupación 4, ya que es el único valor que se buscó maximizar.

Se implementó un modelo base desde el cual partir para definir la performance de los modelos resultantes. Este modelo, la igual que en la regresión, siempre predecirá el valor de nivel de ocupación anterior. Se obtuvo un valor de f1-score 0.63 y recall de 0.54, valores elevados tomando en cuenta que tiene que predecir correctamente entre 5 clases.

#### **5.4.1. Modelos**

Como se comentó anteriormente, los modelos implementados para clasificación son similares a los de regresión, adaptados para funcionar en este tipo de problemas. Se exceptuó la implementación de modelos por cuadra, ya que se vió en el caso anterior de que no se obtuvieron buenos resultados salvo para algunas cuadras específicas, lo que no resulta útil para el problema que se busca resolver.

Las salidas de los todos los modelos, salvo árboles de decisión y random forest, debe ser categórica, por lo que nuestros valores conocidos de salida deben pasar por una transformación one-hot-encoding, luego de la cual obtendremos una fila por valor de salida de igual tamaño que posible salidas, cargada con todos ceros, exceptuando la posición en la que se encuentra la salida, la cual posee un uno.

Como función de pérdida de las regresiones multinomiales y las redes neuronales implementadas se utilizó la función de pérdida cross-entropy categórico, ya que es ampliamente utilizada y recomendada para este tipo de problemas. A la salida de este tipo de modelos se le aplicó la función de activación softmax, ya que permite obtener, en porcentajes, cuál clase es la más probable para los valores de entrada utilizados.

Para los modelos de árbol de clasificación y random forest, se utilizó gini como criterio de separación de nodos, ya que es el algoritmo utilizado por defecto. Según Rokach et. al. [29] “muchos investigadores apuntan a que, en la mayoría de los casos, la elección del criterio de separación de nodos no muestra gran diferencia en la performance final de los árboles de decisión”.

#### **5.4.2. Comparación entre modelos de clasificación**

Los modelos resultantes, comparados con el modelo base, ofrecen resultados apenas hasta un 5% mejores, aunque los resultados obtenidos de recall llegan a superar al del modelo base en un 15%, lo que indica una notable mejoría en las predicciones obtenidas, mayormente en el nivel de ocupación crítico.

El modelo redes neuronales (valores numéricos) es el que mayor valor de F1-score tiene, aunque puede observarse que el recall para la salida 4 es muy bajo, con un valor de apenas 34%, siendo este un 10% más bajo que el modelo baseline.

Tres modelos son los que poseen mayor valor de recall para el valor crítico. Estos modelos son árbol de clasificación y random forest, con valor de 69%. Aunque estos modelos pudieron mejorar el valor de recall, su valor de F1-score es similar al del modelo base.

Se puede observar que los modelos no consiguieron mejorar en gran manera los resultados obtenidos con el modelo base. Apenas se consiguió mejorar el valor de recall en un 15%, aunque al ver que el valor de F1-score es similar, podemos asumir que disminuyó el valor de recall de otros valores para mejorar este.

Modelo	F1-score	Recall (salida 4)
Base	0.63	0.54
Regresión multinomial logística (valores numéricos)	0.62	0.02
Regresión multinomial logística (valores numéricos y categóricos)	0.64	0.28
<b>Árbol de clasificación entrenado con todo el dataset (profundidad 13)</b>	<b>0.63</b>	<b>0.69</b>
<b>Random forest para clasificación (profundidad 6)</b>	<b>0.63</b>	<b>0.69</b>
Red neuronal artificial (valores numéricos)	0.63	0.24
Red neuronal artificial (valores numéricos y categóricos)	0.67	0.34
Redes recurrentes entrenado sólo ocupación	0.64	0.28
Redes recurrentes entrenado con todas las features	0.67	0.44

**Tabla 5.5:** resultados de las métricas (F1-score y recall para la salida 4) obtenidos por cada modelo.

Se realizó una comparación de tres métricas (Precision, recall y F1-score) de todas las clases entre los modelos base y arbol de clasificación, ya que son el modelo más simple que mayor recall obtuvo. Al observar estos valores podemos ver que los predictores entrenados no pudieron superar en gran medida los valores del modelo base, aunque sí hubo un gran aumento en el recall de la clase 4 y una gran disminución en la precisión de esa misma clase. Esto indica que el modelo prefiere elegir como valor de salida la clase 4, aunque no sea la clase correcta.

	Precision M1	Precision M2	Recall M1	Recall M2	F1-score M1	F1-score M2
0	0.72	0.73	0.73	0.79	0.73	0.76
1	0.60	0.66	0.60	0.55	0.60	0.60
2	0.56	0.56	0.55	0.54	0.55	0.55
3	0.51	0.39	0.48	0.53	0.50	0.45
4	0.54	0.31	0.54	0.69	0.53	0.42

**Tabla 5.6:** Comparación de precisión y recall de los modelos M1 (modelo base) y M2(árbol de clasificación) por cada clase.

## 5.5. Complejización de los modelos de deep learning

Además de realizar las pruebas con los modelos planteados, se probó implementar modelos de deep learning de mayor complejidad, es decir, con más capas ocultas. Más específicamente, se añadieron, en el orden especificado, las siguientes capas antes de la capa de salida de la red:

Primero, una capa que normaliza los valores provenientes de la capa anterior, con el fin de evitar valores altamente diferentes y aumentar la velocidad de entrenamiento de la red.

- Una capa densa, con 256 nodos y como función de activación se optó por ReLU.
- Luego una capa de dropout con un 20% de posibilidad, buscando evitar que el modelo sufra de overfitting.
- Otra capa densa, con 512 nodos y como función de activación se optó por ReLU.
- Otra capa de normalización de batch.
- Otra capa de dropout con 20% de posibilidad.
- Por último, otra capa densa, con 512 nodos.

Al entrenar los modelos con esta estructura, se pudo observar que aumentó en gran medida el tiempo de entrenamiento. Esto es debido a la gran cantidad de nodos y parámetros entrenables agregados.

A pesar de esto, los modelos fueron entrenados y se pudo observar una mejora de los modelos en lo que respecta al valor de recall para el valor crítico en algunos casos. Para no extender el informe en gran medida, se ejemplifica uno de los resultados obtenidos para un único modelo de clasificación, ya que todos tuvieron una variación similar.

En el modelo de clasificación que utiliza redes neuronales con atributos numéricos y categóricos, se puede observar un incremento de 1% para la métrica de F1-score, obteniendo un valor total de 67%. A pesar de esto, el modelo únicamente consigue un valor de recall de 44% para la clase 4.

Aunque el valor de F1-score sea el más alto conseguido para los modelos de clasificación, el valor crítico (recall para la clase 4) es mucho menor que el obtenido en los demás modelos. Este valor es de apenas un 44%, contra un 69% conseguido con árbol de clasificación y random forest, los cuales no se vieron afectados por el incremento de complejidad. Si se quisiera obtener únicamente el modelo más certero, podríamos mantenernos con que ofrece un valor de 67% de F1-score. En este caso, buscamos un modelo capaz de predecir correctamente la mayor cantidad de clases posibles, pero tomando en cuenta que la clase crítica debe tener un nivel de predicción aceptable.

En definitiva, los modelos no sufrieron mejoras considerables. El valor de F1-score se mantuvo similar a los modelos con topologías más simples. El valor de recall obtuvo mejoras pero sin superar el modelo baseline, indicándonos que los modelos no pudieron aprender correctamente sobre los datos.

## **5.6. Comparación entre modelos de regresión y clasificación**

El objetivo de realizar modelos tanto de regresión como de clasificación fue para buscar diferentes maneras de resolver un mismo problema. En el caso de los modelos de regresión, se busca predecir la ocupación en la siguiente media hora. En los modelos de regresión, se buscó predecir el nivel de ocupación en ese tiempo, perdiendo la exactitud de la cantidad de autos estacionados, pero pudiendo de este modo plantear los modelos como clasificadores y poder predecir si va a haber lugar para estacionar o no.

Ambas maneras de resolver el problema son válidas y proporcionan resultados útiles. En caso de que se requiera conocer con exactitud la cantidad de autos estacionados, la mejor opción es recurrir a modelos de regresión, ya que el objetivo de estos es predecir valores numéricos. En cambio, si se busca conocer qué nivel de ocupación se va a encontrar en una calle, se pueden realizar ambos tipos de modelos, de regresión y clasificación. En este caso se optó por realizar modelo de clasificación ya que se buscaba incorporar nuevas formas de resolver este problema, utilizando diferentes algoritmos.

En ambos casos, los modelos base ofrecen valores elevados, ya que en los datos hay muchos casos donde la salida esperada es igual a la entrada.

En los modelos entrenados puede verse una leve mejoría en las predicciones obtenidas, aunque en ninguno de los dos casos se ofrecen valores muy elevados de sus respectivas métricas.

El mejor modelo de regresión, random forest, ofrece un 6.3% mejores predicciones que el modelo base y tan solo 1.913 de RMSE, por lo que podemos esperar que las equivocaciones realizadas por el predictor no sean extremadamente elevadas. Hay que tener en cuenta que el valor de RMSE obtenido es, aproximadamente, un 15% mejor que el de modelo base.

En los mejor modelo de clasificación, árbol de clasificación y random forest para clasificación, no se ve un incremento en la métrica F1-score, aunque sí un de 15% en recall para el valor de salida 4, por lo que, en este caso, podremos esperar un incremento aceptable en las predicciones aceptadas para nuestro valor crítico. Aunque, como vimos anteriormente, este resultado posee un alto recall aunque disminuye notablemente su precisión, por lo que el modelo tiende a predecir esta clase por sobre otras.

Ambos modelos ofrecen mejoras similares con respecto a las obtenidas por los modelos base. A pesar de esto, ambos resuelven de manera distinta el mismo problema, pudiendo utilizar ambos modelos para buscar mejorar las predicciones.

## **Capítulo 6: Conclusión**

El objetivo del trabajo fue obtener predicciones sobre los lugares de estacionamiento que se encontrarán libres en un momento determinado. Para ello se analizó la información ofrecida por los parquímetros de la ciudad de Tandil y se implementaron modelos de machine learning.

A partir del análisis de los datos se comprendió la utilidad de los atributos y su relevancia a la hora de aplicarlos en un modelo de machine learning. A su vez, se obtuvo información relevante sobre el comportamiento de los conductores a lo largo del día, en donde se pudieron identificar patrones y los horarios en donde se encuentran la mayor cantidad de autos estacionados.

La hipótesis en la que se afirmaba que es posible extraer información sobre las acciones de los conductores a través de los parquímetros de la ciudad de Tandil fue validada. Se pudo observar que los conductores tienden a seguir el mismo patrón de estacionamiento. A las 10 de la mañana se visualiza una gran cantidad de entradas al sistema, disminuyendo gradualmente hacia el mediodía y volviendo a aumentar el valor a la tarde. Además se advirtieron diferencias en la ocupación de cada calle según el día y el mes en el que ocurrieron.

Para poder entrenar los modelos con los datos obtenidos, fue necesario realizar un preprocesamiento de los mismos. Durante el análisis se observó que el tiempo entre ingresos o salidas por cada usuario difiere constantemente. Para solucionar este problema, se organizaron los datos para que las diferencias entre el tiempo de una dato y el siguiente fuese de treinta minutos, tomando el valor máximo dentro de este intervalo. Esta solución, a pesar de no ser la óptima (en el caso ideal se usarían los datos en los intervalos de tiempo irregulares, como se obtienen originalmente), proporciona un conjunto de datos que puede utilizarse en los modelos de aprendizaje de máquina planteados. Se realizó de este modo por la poca cantidad de algoritmos y el escaso estudio en el área de series de tiempo con intervalos irregulares.

Para abordar la problemática de encontrar lugares para estacionar se plantearon dos posibles soluciones: aplicar modelos de regresión y de clasificación. Se observó que en ninguna de estas alternativas se mostraron mejoras notables respecto a los modelos base planteados (prediciendo el mismo valor de salida que el valor actual). A pesar de esto, se puede concluir que los modelos brindaron buenos resultados, es decir, la mayoría de las veces se observó que el valor a predecir era el mismo que se poseía de entrada.

Este proyecto representó grandes dificultades a la hora del desarrollo. El principal inconveniente fue cuantificar el espacio libre por cuadra, lo cual fue imposible de realizar ya que esta variable depende de cómo los conductores estacionen sus vehículos. Además, pueden surgir vehículos fantasmas, como se detalló en el apartado de problemas iniciales. Aunque esto último no modifica el comportamiento de los predictores, dificulta la credibilidad de los mismos, ya que los resultados obtenidos pueden no ser ciertos a causa de vehículos estacionados fantasmas. Estas problemáticas traen aparejadas inestabilidad en las respuestas de los predictores. En el caso de que se obtengan buenos resultados en las métricas

correspondientes, estos se logran tomando en cuenta únicamente los datos ingresados por los usuarios de los parquímetros, por lo que si algún conductor estaciona y no ingresa al sistema no tenemos manera de saberlo.

## **6.1 Implicancias**

La principal contribución de esta investigación se centra en la predicción de lugares libres de estacionamiento en las cuadras de la ciudad de Tandil. Esto deriva de la creciente necesidad de administrar el tráfico en ciudades pequeñas y medianas las cuales, a pesar de no poseer la misma cantidad de vehículos circulando como en las grandes ciudades, poseen gran volúmen en ciertos horarios.

Los modelos de machine learning diseñados proporcionan predicciones sobre la ocupación de cada cuadra, lo cual permitirá a los conductores agilizar el proceso de encontrar lugares para estacionar. Como aspectos positivos de esta cuestión se desprenden la reducción del tiempo usualmente empleado en esta tarea, la reducción de la contaminación generada por los vehículos y la disminución del tráfico en la ciudad.

Al conocer, por ejemplo, las cuadras de mayor ocupación o los momentos del día y del año en donde hay picos de estacionamientos, se pueden desarrollar herramientas que complementen el trabajo de los encargados del estacionamiento en la ciudad en lo que respecta al control y organización del tránsito.

Como ventaja se puede destacar la posibilidad de extender el mismo análisis a diferentes conjuntos de datos obtenidos de la misma fuente, pudiendo comprender el comportamiento en ciertos períodos de tiempo o utilizando todos los datos históricos recolectados por los parquímetros de la ciudad.

## **6.2. Limitaciones**

Una de las principales limitaciones del trabajo es que, gracias al constante crecimiento del parque automotor, hay cada vez más vehículos en las ciudades, por lo que los modelos de machine learning generados deberán ajustarse continuamente para brindar las predicciones actualizadas al volumen de autos en circulación del momento. Esto se evidenció al comparar los datos del dataset 2018 con los del dataset 2019, en los cuales se observó un incremento significativo sobre la cantidad de ocupación en las cuadras.

Al emplear los datos recolectados por los parquímetros inteligentes, las predicciones obtenidas servirán únicamente en la ciudad de Tandil, por lo que no se podrá implementar esta solución en otras localidades. Si tenemos en cuenta el costo de implementación es un aspecto positivo ya que los datos utilizados fueron obtenidos a través del sistema de parquímetros vigente.

El hardware disponible para el desarrollo de los modelos no fue el óptimo. Al no contar con una placa de video de alta performance (se utilizó una integrada), las pruebas realizadas llevaron mucho tiempo, por lo que no se pudieron probar modelos con elevada complejidad ya que hubiese sido imposible entrenarlos.

### **6.3. Trabajos Futuros**

A partir de lo analizado y desarrollado, se observa la posibilidad de extenderlo realizando trabajos complementarios, los cuales serían de gran utilidad y no forman parte de esta investigación.

#### **6.3.1. Aplicación para dispositivos móviles**

El objetivo de este trabajo es facilitar la tarea de encontrar lugar para estacionar en la ciudad de Tandil, por lo que desarrollar una aplicación para dispositivos móviles e implementar lo desarrollado en este trabajo brindaría esta solución a los habitantes más fácilmente.

Además, se podrá recomendar otras cuadras cercanas para estacionar si la cuadra solicitada se encuentra completa. Esto permitirá a los usuarios encontrar con mayor facilidad lugar para estacionar ya que, en caso de que se prediga lugar libre y no haya, se evaluaron y mostraron otras posibilidades.

#### **6.3.2. Ampliación de datos**

En este trabajo los únicos datos utilizados fueron los obtenidos por el sistema SUMO en el año 2019. Una posibilidad es ampliar este conjunto de datos obteniendo datos del clima, los cuales demostrarán patrones de comportamiento en los usuarios que no se han podido observar en el informe. Además, podrán utilizarse datos de fechas festivas, como por ejemplo Semana Santa, en donde el nivel de autos estacionados incrementa notablemente.

Otra manera de poder utilizar mayor cantidad de datos sería obtener los datos en tiempo real del sistema, para poder entrenar continuamente al modelo y brindar siempre resultados actualizados. Como se vio en este trabajo, los datos de 2018 difieren en gran parte a los datos del 2019, por lo que podemos asumir que esto sucede año a año, ya que la ciudad crece constantemente.

## Bibliografía

1. Manville M, Shoup D. Parking, People, and Cities. *J Urban Plann Dev.* 2005;131: 233–245.
2. Lin T, Rivano H, Le Mouel F. A Survey of Smart Parking Solutions. *IEEE Trans Intell Transp Syst.* 2017;18: 3229–3253.
3. E. Moguel, M. Preciado, and J. Preciado. Smart parking campus: An example of integrating different parking sensing solutions into a single scalable system. *ERCIM News Smart Cities.* 2014; 29–30.
4. Evenepoel S, Van Ooteghem J, Verbrugge S, Colle D, Pickavet M. On-street smart parking networks at a fraction of their cost: performance analysis of a sampling approach. *Trans Emerging Tel Tech.* 2014;25: 136–149.
5. Villanueva FJ, Villa D, Santofimia MJ, Barba J, Lopez JC. Crowdsensing smart city parking monitoring. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). IEEE; 2015. pp. 751–756.
6. Kopecký J, Domingue J. ParkJam: Crowdsourcing Parking Availability Information with Linked Data (Demo). In: Simperl E, Norton B, Mladenic D, Della Valle E, Fundulaki I, Passant A, et al., editors. *The Semantic Web: ESWC 2012 Satellite Events.* Berlin, Heidelberg: Springer Berlin Heidelberg; 2015. pp. 381–386.
7. Stenneth L, Wolfson O, Xu B, Yu PS. PhonePark: Street Parking Using Mobile Phones. 2012 IEEE 13th International Conference on Mobile Data Management. IEEE; 2012. pp. 278–279.
8. Xu B, Wolfson O, Yang J, Stenneth L, Yu PS, Nelson PC. Real-Time Street Parking Availability Estimation. 2013 IEEE 14th International Conference on Mobile Data Management. IEEE; 2013. pp. 16–25.
9. Gupte S, Younis M. Participatory-sensing-enabled efficient Parking Management in modern cities. 2015 IEEE 40th Conference on Local Computer Networks (LCN). IEEE; 2015. pp. 241–244.
10. Venners, Bill (13 January 2003). "The Making of Python". Artima Developer. Artima. Retrieved 22 March 2007.
11. McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14(9).

- 12.** Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283).
- 13.** Evans, J. (1992). Optimization algorithms for networks and graphs. CRC Press.
- 14.** Gulli, A., & Pal, S. (2017). Deep learning with Keras. Packt Publishing Ltd.
- 15.** Beauxis-Aussalet, E., & Hardman, L. (2014). Visualization of confusion matrix for non-expert users. In IEEE Conference on Visual Analytics Science and Technology (VAST)-Poster Proceedings.
- 16.** Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013, September). Facing imbalanced data--recommendations for the use of performance metrics. In 2013 Humaine association conference on affective computing and intelligent interaction (pp. 245-251). IEEE.
- 17.** Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern recognition*, 37(9), 1757-1771.
- 18.** Botchkarev, A. (2018). Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. arXiv preprint arXiv:1809.03006.
- 19.** Barten, A. P. (1987). The coefficient of determination for regression without a constant term. In *The Practice of Econometrics* (pp. 181-189). Springer, Dordrecht.
- 20.** Stephen, I. (1990). Perceptron-based learning algorithms. *IEEE Transactions on neural networks*, 50(2), 179.
- 21.** Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Physica-Verlag HD.
- 22.** Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- 23.** Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781.
- 24.** Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

- 25.** Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196).
- 26.** Li, Y., Xu, L., Tian, F., Jiang, L., Zhong, X., & Chen, E. (2015, June). Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In Twenty-Fourth International Joint Conference on Artificial Intelligence.
- 27.** Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, 14(8).
- 28.** Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. arXiv preprint arXiv:1804.07612.
- 29.** Rokach, L., & Maimon, O. Z. (2008). Data mining with decision trees: theory and applications (Vol. 69). World scientific.
- 30.** Peláez, I. M. (2016). Modelos de regresión: lineal simple y regresión logística. Revista Seden.
- 31.** Jabbar, H., & Khan, R. Z. (2015). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Computer Science, Communication and Instrumentation Devices.
- 32.** Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.
- 33.** Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
- 34.** Eckner, A. (2012). A framework for the analysis of unevenly spaced time series data. Preprint.
- 35.** Rodriguez Alvarez J, Arroqui M, Mangudo P, Toloza J, Jatip D, Rodriguez JM, Teyseyre A, Sanz C, Zunino A, Machado C, Mateos C. (2018): Body condition estimation on cows from depth images using Convolutional Neural Networks. *Computers and electronics in agriculture*, 155, 12-22. doi.org/10.1016/j.compag.2018.09.039
- 36.** Carson, J. S. (2002, December). Model verification and validation. In Proceedings of the winter simulation conference (Vol. 1, pp. 52-58). IEEE.
- 37.** Rodriguez JM, Zunino A, Tommasel A, Mateos C. Recurrent Neural Networks for Predicting Mobile Device State. In: Khosrow-Pour D.B.A. M, editor. Encyclopedia of Information Science and Technology, Fourth Edition. IGI Global; 2018. pp. 6658–6670.

- 38.** Hirsch M, Rodríguez JM, Mateos C, Zunino A. A Two-Phase Energy-Aware Scheduling Approach for CPU-Intensive Jobs in Mobile Grids. *J Grid Computing*. 2017;15: 55–80.
- 39.** Rodriguez JM, Godoy D, Mateos C, Zunino A. A multi-core computing approach for large-scale multi-label classification. *IDA*. 2017;21: 329–352.
- 40.** Armentano MG, Amandi AA. Modeling sequences of user actions for statistical goal recognition. *User Model User-adapt Interact*. 2012;22: 281–311.
- 41.** Christensen I, Schiaffino S, Armentano M. Social group recommendation in the tourism domain. *J Intell Inf Syst*. 2016;47: 209–231.
- 42.** Monteserin A, Armentano MG. Influence-based approach to market basket analysis. *Inf Syst*. 2018; doi:10.1016/j.is.2018.01.008
- 43.** CommuniThings SmartParking. <https://www.communitings.com/sensors.php>
- 44.** Yang, Z., Jia, P., Liu, W., & Yin, H. (2017). Car ownership and urban development in Chinese cities: A panel data analysis. *Journal of Transport Geography*, 58, 127-134.
- 45.** Gössling, S. (2020). Why cities need to take road space from cars-and how this could be done. *Journal of Urban Design*, 1-6.