Universitat Oberta de Catalunya

uoc.edu

# Análisis y predicción de movilidad a partir del tráfico de dispositivos móviles

**Álvaro Gallego Sacristán**

Máster en Ingeniería de Telecomunicaciones

Smart Cities

**David Crespo García**

**Víctor Monzón Baeza**

Fecha Entrega

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | *Análisis y predicción de movilidad a partir del tráfico de dispositivos móviles* |
| **Nombre del autor:** | *Álvaro Gallego Sacristán* |
| **Nombre del consultor/a:** | *David Crespo García* |
| **Nombre del PRA:** | *Víctor Monzón Baeza* |
| **Fecha de entrega (mm/aaaa):** | MM/AAAA |
| **Titulación:** | *Máster en Ingeniería de Telecomunicación* |
| **Área del Trabajo Final:** | *Smart Cities* |
| **Idioma del trabajo:** | *Español* |
| **Palabras clave** | *Tráfico, machine learning, COVID, algoritmos, data science, análisis de datos* |

**Resumen del Trabajo (máximo 250 palabras):** *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*


Este Trabajo Final de Máster utiliza un conjunto de datos de localización de tráfico móvil recogidos de antenas GPRS. Los datos estarán basados en diferentes localizaciones geográficas dentro de una ciudad con gran incidencia turística. Con el objetivo de reducir el gran volumen de registros obtenidos se analizarán tramos horarios concretos.

Una vez estructurados y atendiendo a las coordenadas de latitud y longitud y tramo horario se aplican diferentes técnicas de análisis de datos sobre un lenguaje de programación para ver las zonas con mayor concurrencia. Esta información analizada se representa geográficamente y se elaboran mapas de calor.

Posteriormente se utiliza la tecnología de *Machine Learning* para la predicción de la afluencia de personas según el tramo horario.

Tiene como objetivo principal analizar el aumento progresivo de la afluencia de personas causada por la disminución de las restricciones COVID en una localización turística y poder extraer información relevante para el ámbito de las Smart Cities. Por ejemplo, reforzar medios de transporte o servicios.

**Abstract (in English, 250 words or less):**

During this thesis, mobile connections datasets based on an important Spanish touristic city obtained from GPRS antennas have been analyzed. These antennas provide information about its coordinates and the exact time of the connection.

After giving a complete introduction in this knowledge area, the project is divided in three parts. Starting from the ETL process, where data are collected from an external database, a cleaning process of the information is carried out to ensure the datasets quality. After that, these datasets are loaded in an analytic system to start working with them.

Afterward, an important study of the datasets takes place. Analyzing different days, one in April, a month where the State of Alarm was still present with so many restrictions and another one from May where it was eliminated, significant differences are found. These variations are represented and explained carefully using different visualization tools.

The last step consists on using Machine Learning algorithms to make predictions. To do this, time series are created, one involving geographic coordinates and the other one with the number of connections per hour. In this case, several weeks are used to train the model. Finally, two different weeks are predicted one from April and one from May.

As a conclusion, the restrictions removal in May caused an increase in the number of connections per day that directly affect to the city mobility. Considering all the information provided, important changes to improve city life can be made, especially in Smart Cities ambit.

# Índice


**1. INTRODUCCIÓN ..... 1**

**1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO** ..... 1
**1.2 OBJETIVOS DEL TRABAJO** ..... 2
**1.3 ENFOQUE Y MÉTODO SEGUIDO** ..... 2
**1.4 PLANIFICACIÓN DEL TRABAJO** ..... 3
**1.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS** ..... 5
**1.6 BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA** ..... 5

**2. ESTADO DEL ARTE ..... 6**

**2.1 ANTECEDENTES** ..... 6
**2.2 RED MÓVIL** ..... 7
*2.2.1 Tipos de redes móviles ..... 8*
**2.3 ANÁLISIS DE DATOS** ..... 9
*2.3.1 Bases de datos ..... 12*
*2.3.2 Python ..... 13*
**2.4 *MACHINE LEARNING*** ..... 13
*2.4.1 Regresión Lineal ..... 14*
*2.4.2 Redes neuronales ..... 15*
*2.4.3 K-vecinos más cercanos ..... 15*
*2.4.4 Árboles de decisión ..... 15*
*2.4.5 Algoritmos Bayesianos ..... 16*
*2.4.6 Algoritmos de agrupación ..... 16*
**2.5 HERRAMIENTAS DE VISUALIZACIÓN DE MAPAS** ..... 16
**2.6 PROYECTOS RELACIONADOS** ..... 17

**3. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA ..... 19**

**3.1 PROCESO DE ETL** ..... 19
*3.1.1 Proceso de extracción y limpieza de los datos ..... 21*
*3.1.2 Proceso de exportación de datos ..... 23*
*3.1.3 Proceso de carga de datos ..... 26*
**3.2 ANÁLISIS DE DATOS** ..... 27
*3.2.1 Introducción ..... 27*
*3.2.2 Descripción del proceso ..... 29*
*3.2.3 Representación de los resultados ..... 33*
**3.3 ALGORITMOS PREDICTIVOS DE MACHINE LEARNING** ..... 37
*3.3.1 Preparación y análisis de las series temporales ..... 38*
*3.3.1 Modelo SARIMA ..... 44*
*3.3.2 Elaboración del modelo ..... 45*
*3.3.3 Resultados ..... 52*

**4. CONCLUSIONES ..... 59**

**5. GLOSARIO ..... 61**

**6. BIBLIOGRAFÍA ..... 62**

**7. ANEXOS ..... 65**

**7.1 CÓDIGO PYTHON ANÁLISIS DE DATOS** ..... 65
**7.2 CÓDIGO PREDICCIÓN SERIES TEMPORALES** ..... 71
*7.2.1 Número de conexiones por hora (igual, tanto para abril como para mayo) ..... 71*
*7.2.2 Latitud y longitud media por hora (igual, tanto para abril como para mayo) ..... 72*

## Lista de figuras

Ilustración 1: Diagrama de Gantt 4
Ilustración 2: Mapa restricciones COVID-19 por CCAA [6] 6
Ilustración 3: Funcionamiento red telefonía móvil [8] 8
Ilustración 4: Tipos de análisis de datos [12] 10
Ilustración 5: Etapas del análisis de datos [13] 11
Ilustración 6: Regresión Lineal 15
Ilustración 7: Árbol de decisión [26] 16
Ilustración 8: Mapa de calor con Leafset 17
Ilustración 9: Movilidad de la población durante el Estado de Alarma [38] 18
Ilustración 10: LEFT JOIN SQL [40] 19
Ilustración 11: Identificador único del municipio 21
Ilustración 12: Información relevante de las estaciones base 21
Ilustración 13: Número de estaciones base diferentes en Mérida 22
Ilustración 14: Información final a exportar 23
Ilustración 15: Sentencias de creación de tablas para exportación 24
Ilustración 16: Número de registros por tabla 24
Ilustración 17: Proceso de exportación 25
Ilustración 18: Número de registros de cada fichero 26
Ilustración 19: Carga de datos en Python 26
Ilustración 20: Muestra aleatoria de del dataset de partida 27
Ilustración 21: Información de los campos del dataset de partida 28
Ilustración 22: Estadísticas de los campos del dataset de partida 28
Ilustración 23: Media horaria de los dataset de partida 28
Ilustración 24: Conteo de los registros agrupados 29
Ilustración 25: Cálculo de la variación de registros entre dos días 30
Ilustración 26: Categorización de la variación de aumento 31
Ilustración 27: Lista con las variaciones categorizadas 31
Ilustración 28: Elaboración del mapa con marcadores de variaciones 32
Ilustración 29: Proceso de creación del mapa de calor 32
Ilustración 30: miércoles 26 de mayo Vs miércoles 21 de abril 33
Ilustración 31: sábado 22 de mayo Vs sábado 24 de abril 34
Ilustración 32: Gráfica variación principales puntos geográficos 34
Ilustración 33: Mapa de calor miércoles 21 de abril 35
Ilustración 34: Mapa de calor miércoles 26 de mayo 35
Ilustración 35: Mapa de calor sábado 24 de abril 36
Ilustración 36: Mapa de calor sábado 22 de mayo 36
Ilustración 37: Elaboración de la serie temporal con el número de registros totales. 38
Ilustración 38: Registros totales por hora 39
Ilustración 39: Serie temporal de las conexiones totales por hora en una semana (19-23 abril) 39
Ilustración 40: Serie temporal de las conexiones totales por hora en un día (19 de abril) 40
Ilustración 41: : Elaboración de la serie temporal con la media de las coordenadas por hora. 40

Ilustración 42: Serie temporal de la media de latitud por hora en una semana (19-23 abril) 41
Ilustración 43: Serie temporal de la media de latitud por hora en un día (19 de abril) 41
Ilustración 44: Serie temporal de la media de longitud por hora en una semana (19- 23 abril) 41
Ilustración 45: Serie temporal de la media de longitud por hora en un día (19 abril) 42
Ilustración 46: Serie temporal de las conexiones totales por hora en una semana (24-28 mayo) 42
Ilustración 47: Serie temporal de la media de latitud por hora en una semana (24-28 mayo) 43
Ilustración 48: Serie temporal de la media de latitud por hora en un día (24 de mayo) 43
Ilustración 49: Serie temporal de la media de longitud por hora en una semana (24-28 mayo) 44
Ilustración 50: Selección de los datos de entrenamiento y de prueba de las conexiones diarias por hora (19-23 de abril) 45
Ilustración 51: Representación de los datos de entrenamiento y prueba de las conexiones diarias por hora (19-23 de abril). 46
Ilustración 52: Diferenciación de la serie temporal de las conexiones diarias por hora (19-23 de abril). 46
Ilustración 53: PACF de la serie temporal de las conexiones diarias por hora (19-23 de abril). 47
Ilustración 54: ACF de la serie temporal de las conexiones diarias por hora (19-23 de abril). 48
Ilustración 55: Entrenamiento modelo SARIMA 49
Ilustración 56: Predicción de los datos de prueba de las conexiones diarias por hora (19-23 de abril). 49
Ilustración 57: Predicción de los datos de prueba de la media de la longitud por hora (19-23 de abril). 50
Ilustración 58: Predicción de los datos de prueba de la media de la longitud por hora (19-23 de abril). 51
Ilustración 59: Predicción de las conexiones diarias por hora de abril (datos de prueba) 52
Ilustración 60: Predicción de las conexiones diarias por hora de mayo (datos de prueba) 52
Ilustración 61: Predicción de las conexiones diarias por hora de abril (semana 24-30 de abril) 53
Ilustración 62: Predicción de las conexiones diarias por hora de mayo (semana 22-28 de mayo) 53
Ilustración 63: Comparación de las predicciones de conexiones diarias por hora 53
Ilustración 64: Predicción de la media de la latitud de abril (datos de prueba) 54
Ilustración 65: Predicción de la media de la latitud de mayo (datos de prueba)54
Ilustración 66: Predicción de la media de la latitud por hora (semana 24-30 de abril) 55
Ilustración 67: Predicción de la media de la latitud por hora (semana 22-28 de mayo) 55
Ilustración 68: Comparación de las predicciones de latitud por hora 55

Ilustración 69: Predicción de la media de longitud de abril (datos de prueba) 56
Ilustración 70: Predicción de la media de longitud de abril (datos de prueba) 56
Ilustración 71: Predicción de la media de la longitud por hora (semana 24-30 de abril) 57
Ilustración 72: Predicción de la media de la longitud por hora (semana 22-28 de mayo) 57
Ilustración 73: Comparación de las predicciones de longitud por hora 57
Ilustración 74: Representación puntos geográficos de la predicción 58

# 1. Introducción

## 1.1 Contexto y justificación del Trabajo

La situación epidemiológica actual por motivo del COVID-19 ha provocado una gran cantidad de cambios en la sociedad. Durante el último año se han sucedido diferentes medidas y restricciones que han afectado al movimiento de las personas. Estas restricciones han tenido especial incidencia en destinos turísticos que han visto como se ha reducido considerablemente la entrada de personas.

Según fuentes oficiales el conjunto de actividades relacionadas con la industria del turismo cayó alrededor del 70% del año 2019 al año 2020 [1]. Derivado de esto uno de los aeropuertos más transitados del territorio español, concretamente el aeropuerto de Palma cerró el año 2020 con un desplome en torno al 80% en el tráfico de pasajeros [2].

Es por ello por lo que ha surgido la necesidad de estudiar detalladamente las consecuencias principales y como encontrar soluciones rápidas a esta situación.

A lo largo del año 2021 y con motivo de la desescalada y reducción de restricciones ha aumentado considerablemente el desplazamiento y con ello la afluencia de personas en los principales destinos turísticos.

Dentro del ámbito de las *Smart Cities,* todo esto influye indirectamente en la movilidad inteligente. Esta nueva modalidad consiste en conseguir formas de transporte innovadoras con el objetivo de contribuir a un entorno más limpio y eficiente. Por ejemplo, si una zona dentro de una ciudad es más transitada que otra intentar establecer medios de transporte más apropiados (patinetes eléctricos, bicis eléctricas, trenes ligeros…), mejorar las vías públicas (aceras más grandes, túneles, carriles bici, bus, etc.) o mejorar los servicios (limpieza, atención al ciudadano, oficinas de turismo, policía) [3].

En base a esta necesidad, se obtienen datos de tráfico móvil de diferentes puntos dentro de una misma ciudad para su estudio. Con ello se podrá apreciar cómo ha ido aumentado el movimiento de personas en diferentes partes de una ciudad a lo largo del año y en diferentes franjas horarias para así promover la movilidad inteligente.

Así se pretende analizar utilizando técnicas de análisis de datos la afluencia de gente durante este periodo de tiempo en el que ha aumentado considerablemente el movimiento en zonas turísticas para así poder reforzar determinados servicios.

Además, se realizan predicciones sobre la afluencia de personas en determinadas zonas en función de los distintos tramos horarios, utilizando modelos predictivos basados en la tecnología *Machine Learning.*

## 1.2 Objetivos del Trabajo

Este proyecto tiene como objetivo principal el análisis de los datos de tráfico móvil y el estudio de estos aplicando técnicas de Machine Learning.

No obstante, se han enumerado un listado con los siguientes objetivos de cara a la planificación del proyecto:

- Aprendizaje y utilización del lenguaje de programación Python.
- Obtención del *dataset* a partir de una base de datos externa.
- Diseño e implementación del sistema de tratamiento de los datos para su manipulación y análisis.
- Análisis de los datos.
- Estudio de las diferentes técnicas de *Machine Learning* para la aplicación de modelos predictivos sobre los datos.
- Representación geográfica de los resultados y elaboración de mapas de calor.

## 1.3 Enfoque y método seguido

El enfoque del proyecto será principalmente analítico, ya que no se basará en el desarrollo de un nuevo producto sino en el diseño de la arquitectura necesaria desde la obtención del dato hasta el tratamiento de este.

La metodología utilizada será la desarrollada por la empresa IBM y la más utilizada en proyectos de *Data Science*, el método CRIPS-DM [4]. La gran ventaja de esta metodología es que independientemente de la etapa del proyecto en la que se encuentre siempre existirá la posibilidad de regresar a cualquiera de las anteriores.

En la fase de entendimiento de negocio se llevará a cabo el estudio de las diferentes técnicas que se utilizarán en el proyecto además del entorno socioeconómico en el que se elabora.

Posteriormente se llevará a cabo el denominado *Data Understanding* que se basará en la obtención de los datos asegurando su calidad y entendimiento de

estos para proseguir con el *Data Preparation* que consistirá en la obtención del dataset final que se utilizará en el proyecto.

En la etapa de *Modeling* se emplearán una serie de técnicas estudiadas previamente de predicción y análisis de datos con el que se realizarán diferentes pruebas y simulaciones.

Por último, la etapa de Evaluación donde se elaborarán las conclusiones y se representarán los resultados logrados en las etapas anteriores.

## 1.4 Planificación del Trabajo

De acuerdo con las fechas establecidas para la entrega de cada una de las PECs que componen el proyecto final, se han considerado como hito cada una de ellas.

En el siguiente diagrama de Gantt se muestra la planificación con las tareas e hitos más relevantes del proyecto.

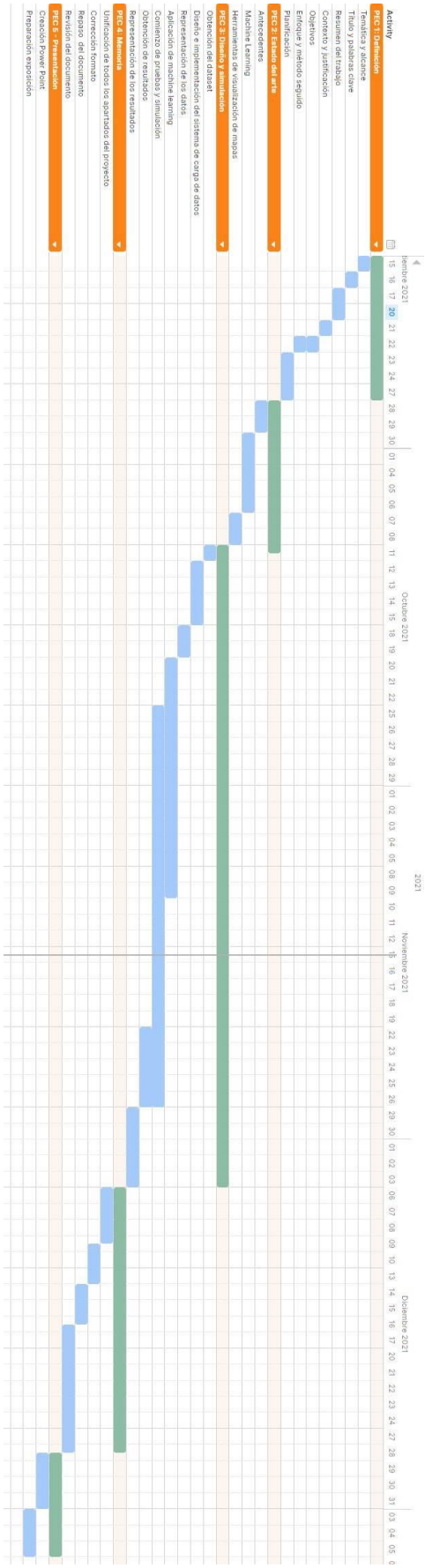


**Ilustración 1: Diagrama de Gantt**

## 1.5 Breve sumario de productos obtenidos

Como resultado de este proyecto se entregará:

- Mapa de las diferentes localizaciones.
- Mapa de calor con la afluencia de personas.
- Resultado del análisis de los datos.
- Modelo predictivo.
- Resultado de simulación y pruebas.

## 1.6 Breve descripción de los otros capítulos de la memoria

Tras un primer capítulo donde se establece el alcance del proyecto, el entorno socioeconómico donde se sitúa, los objetivos y la planificación del proyecto, el segundo capítulo será el estado del arte.

En el estado del arte se entrará al detalle de los antecedentes que han motivado la elaboración del proyecto. También se dará a conocer las técnicas empleadas de análisis de datos, *Machine Learning* y elaboración y representación de mapas.

El tercer capítulo consistirá en la obtención del dataset y el diseño del sistema para su manipulación.

En el cuarto capítulo se realiza el análisis de datos. Posteriormente se aplicará *Machine Learning* seleccionando el algoritmo predictivo más adecuado para el proyecto.

Finalmente, en el quinto capítulo tras las pruebas y simulaciones realizadas se llegará a una serie de conclusiones y se mostrarán los resultados.

# 2. Estado del arte

## 2.1 Antecedentes

Durante estos dos últimos años, debido a la situación provocada por el COVID-19, se han sucedido numerosas restricciones que han afectado a la movilidad de las personas.

En concreto, hasta bien entrado 2021, estas restricciones no se empezaron a reducir. A mediados del mes de mayo con el cese del estado de alarma [5] se eliminó la prohibición de movilidad entre comunidades autónomas. Hay que destacar que en todo el territorio español salvo Madrid y los archipiélagos esta medida estuvo vigente durante los primeros meses del año. Además, en muchos de los territorios se prohibía la movilidad interprovincial e incluso existían cierres perimetrales de municipios en función de la situación epidemiológica.

En el siguiente mapa se observa de manera resumida las principales restricciones.

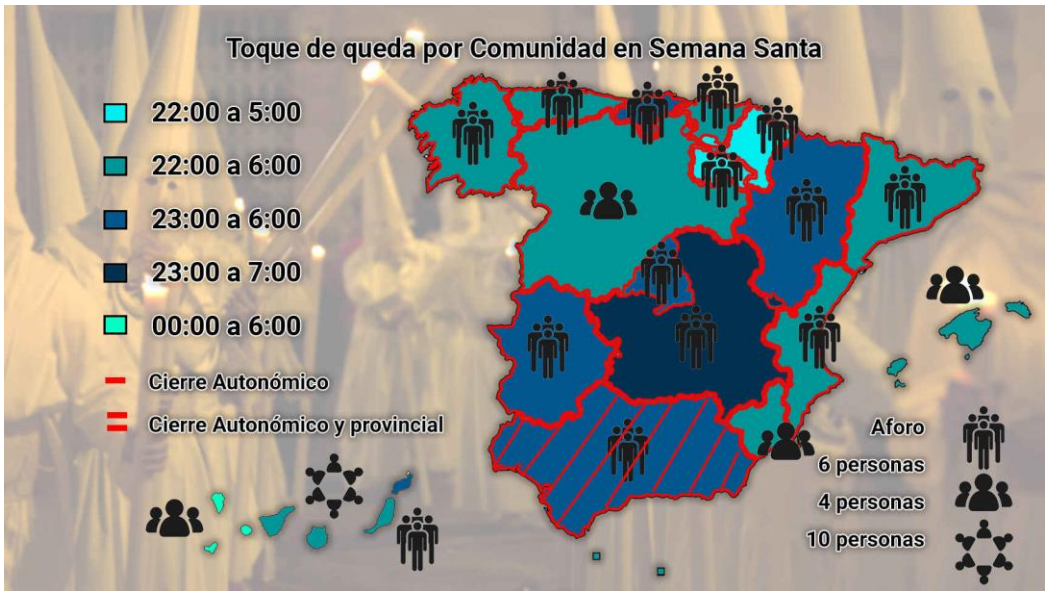


**Ilustración 2: Mapa restricciones COVID-19 por CCAA [6]**

Como se puede apreciar tanto el toque de queda como los cierres autonómicos y perimetrales están presentes en casi todo el territorio español.

Como era de suponer todas estas restricciones influyeron gravemente en el turismo nacional. Durante todo este tiempo la gran mayoría de los servicios se han visto reducidos o incluso muchos de ellos suprimidos por completo.

En este proyecto se llevará a cabo un estudio de una ciudad española habitualmente concurrida y con gran interés turístico.

A partir de datos de tráfico móvil recogidos por distintas estaciones base se realizará un análisis a lo largo de diferentes días en meses diferentes del año para estudiar la movilidad de las personas y ver cómo ha ido aumentando a raíz del fin de las restricciones.

El objetivo de este análisis consiste en ver en qué zonas de la ciudad ha aumentado la afluencia de personas y cuales están más concurridas. También se llevará a cabo una serie de predicciones para determinar donde se encontrará la gente en función del tramo horario analizado.

Con toda esta información lo que se pretende es ver las zonas de la ciudad que necesitarán de un mayor número de servicios con el objetivo de reforzar aquellos lugares que se han visto más afectados por las restricciones con el fin de intentar restablecer la situación prepandemia.

En los siguientes apartados se revisan los conceptos que entrarán en juego en este proyecto. En primer lugar, cómo se lleva a cabo la recolección de datos de tráfico de los diferentes dispositivos. Después se explica qué es el análisis de datos, dónde se almacenan y cómo acceder a ellos. Más tarde se estudian los principales conceptos de *Machine Learning* y técnicas más utilizadas, haciendo hincapié en el lenguaje de programación que se emplea para ello (Python). Por último, se mencionan las diferentes formas de representar mapas geográficos que aparecen en el proyecto.

## 2.2 Red móvil

Una red móvil consiste en una red de células o estaciones bases que cubren un área delimitada (celda) y tienen como objetivo establecer una comunicación entre dispositivos móviles. El responsable de este servicio recibe el nombre de operador de telefonía móvil.

El dispositivo móvil se comunica mediante ondas de radio con las diferentes antenas que a su vez comunican con la central de conmutación del operador, quien encamina la comunicación a través de otras antenas. Para que esta comunicación se produzca correctamente el dispositivo móvil debe encontrarse en el área de alcance de la antena. Las antenas tienen poco alcance y cubren zonas pequeñas llamadas celdas. Es por eso por lo que los diferentes operadores necesitan un gran despliegue de celdas, cada una formada por diferentes estaciones base para garantizar la máxima cobertura.

Las estaciones base tienen capacidad de transmisión limitada por lo que es necesario que en zonas urbanas exista una gran cantidad de celdas para una correcta comunicación. El tamaño de las celdas debe ser pequeño y separadas por distancia cortas. Este incremento de celdas provoca un aumento considerable del número de estaciones base [7].

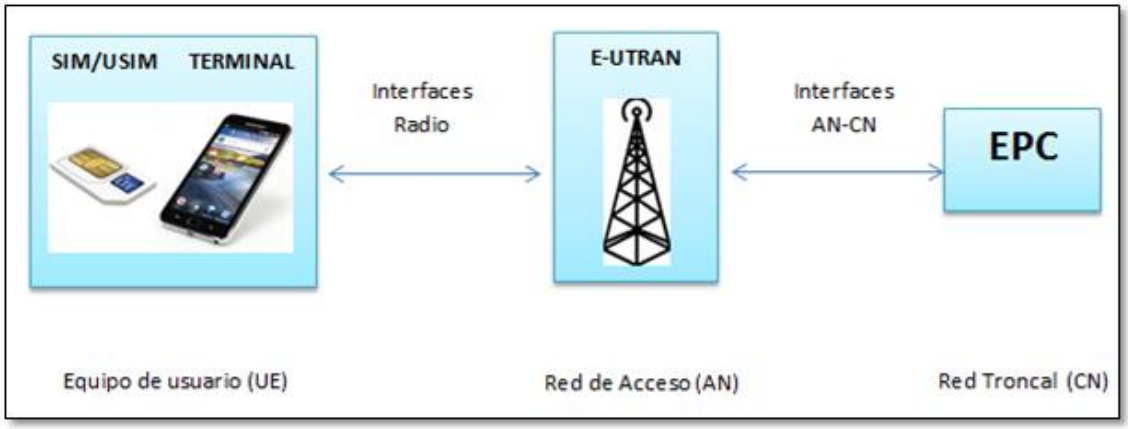


**Ilustración 3: Funcionamiento red telefonía móvil [8]**

En la imagen anterior se muestra el funcionamiento de la red móvil, concretamente el estándar LTE. Tanto la red de acceso E-UTRAN y la red troncal EPC utilizan tecnologías basadas en IP para conectar los diferentes equipos que forman la red de transporte. El elemento E-UTRAN es el que permite al equipo de usuario establecer la comunicación con una estación base.

### 2.2.1 Tipos de redes móviles

Las redes móviles más importantes por orden cronológico son las siguientes [9]:

- Primera generación (1G): se creó en los años 70 y actualmente está obsoleta. Solo permitía la transmisión de voz y utilizaba canales analógicos.

- GPRS (*General Packet Radio Service*) o 2G: es la tecnología más lenta hoy en día. Presentan tasas de velocidad que varían entre los 56 Kbps y los 114 Kbps. Se basa en divisiones de frecuencia sobre dúplex y TDMA.

- EDGE o EGPRS: es la evolución del GPRS y su velocidad puede llegar a los 384 Kbps.

- 3G: También conocida como UMTS. Pueden llegar a velocidad en torno a los 2 Mbps. Está basada en la tecnología W-CDMA.

- LTE: se trata de una evolución del estándar 3G con la gran ventaja de un incremento considerable de la velocidad de navegación, en torno a los 200 Mbps. Posteriormente se mejora con la aparición del LTE-Advanced.

- 4G: la principal diferencia con respecto al LTE es la mejora de la velocidad pudiendo llegar a 1 Gbps.

- 5G: es la red móvil más novedosa. Existen dos tipos de redes:

- NSA. Necesita de la infraestructura de 4G para operar. Es decir, la comunicación entre el dispositivo móvil y la antena se realiza con el nuevo protocolo, pero la comunicación entre antenas utiliza el 4G. Tienen velocidades de bajada cercanas a 2 Gbps y de subida de 150 Mbps.

- SA. Tiene capacidad de operar de manera autónoma, con latencias de un solo milisegundo y velocidades de hasta 20 Gbps.

Los datos de tráfico móvil utilizados en este proyecto tienen como origen estas estaciones base mencionadas anteriormente. Estas estaciones recogerán información cada vez que un dispositivo se conecte a la red, ya sea siendo el transmisor o el receptor de los datos. Existirán datos de tráfico móvil procedentes de los diferentes sistemas explicados.

Esta información recoge tanto las coordenadas de latitud como de longitud en sistema decimal de las estaciones base y el momento exacto en el que se produce la comunicación, parámetros fundamentales que se emplearán en el proyecto.

A continuación, se explicará de manera resumida que es el análisis de datos y algunas de las herramientas más utilizadas.

## 2.3 Análisis de datos

El análisis de datos es la ciencia que se encarga de inspeccionar un conjunto de datos con el objetivo de extraer información relevante que ayude en la toma de decisiones.

En la última década han aparecido novedosas técnicas cada vez más perfeccionadas para llevar acabo el análisis de datos. Así, la ciencia de datos es la materia que reúne todas estas técnicas y que requiere de un gran conocimiento en campos como la programación, matemáticas y estadística [10]. Algunos ejemplos de estas técnicas son el aprendizaje automático o *Machine Learning*, el aprendizaje profundo o *Deep Learning* y el procesamiento del lenguaje natural [11].

Una de las grandes ventajas que proporciona es la capacidad para tratar con el *Big Data,* que consiste en grandes cantidades de datos cuyo análisis proporcionará una posición estratégica a la hora de tomar decisiones.

Se pueden distinguir diferentes tipos de análisis de datos en función de cuál sea nuestro objetivo:

- Si lo que interesa es saber cómo actuar ante una situación determinada, se llevará a cabo un análisis prescriptivo.

- Si por el contrario es necesario saber que va a pasar, el análisis será predictivo.

- Si hay que averiguar la razón por la que ha ocurrido algo, un análisis de diagnóstico será el más acertado.

- Ante una necesidad de saber lo que hay que hacer ante una determinada situación el análisis más adecuado será el descriptivo.

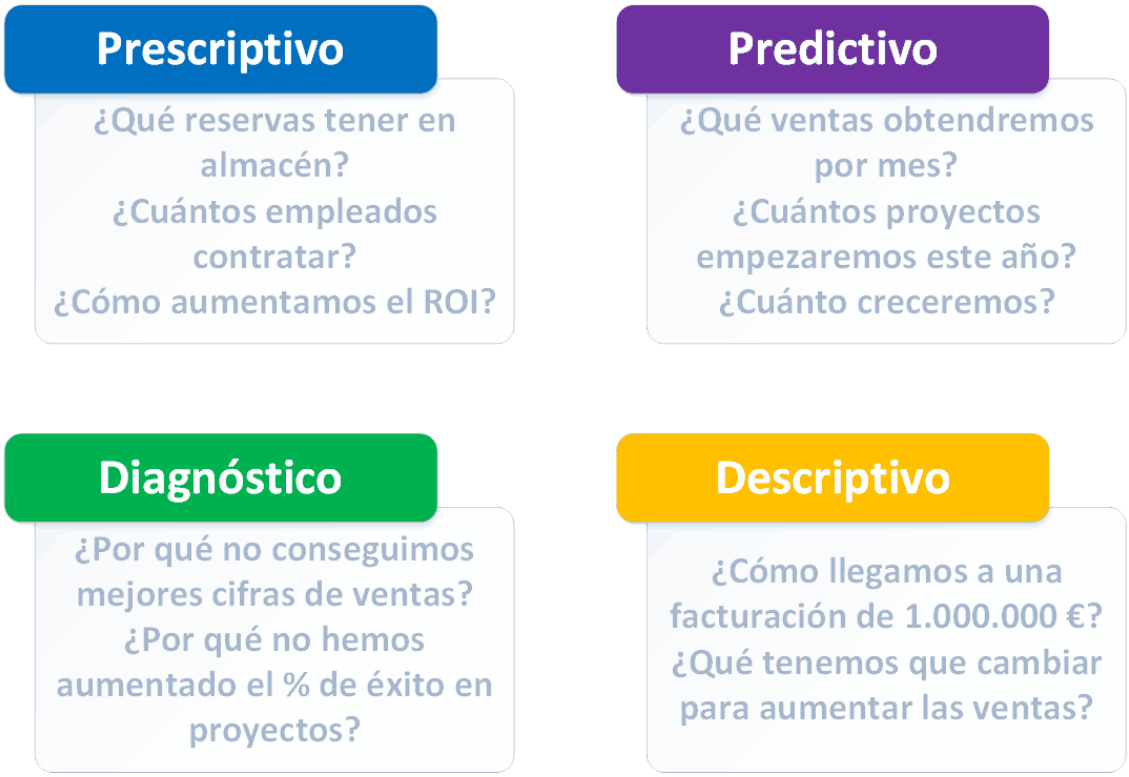


**Ilustración 4: Tipos de análisis de datos [12]**

En este proyecto se emplea el análisis prescriptivo para analizar los diferentes puntos de la ciudad en los que se ha detectado una mayor afluencia de gente. A partir de esa información se podrá ver donde reforzar los diferentes servicios.

También se llevará a cabo un análisis predictivo para averiguar la afluencia de personas que habrá en un determinado punto de la ciudad y en un tramo de hora concreto. Aquí es donde entrara la tecnología *Machine Learning* y los algoritmos predictivos que se explicarán más adelante.

Las fases que tienen lugar dentro del análisis de datos son las siguientes:

1) Obtención de los datos: en esta etapa se lleva a cabo la recopilación de los datos a partir de una fuente, normalmente bases de datos.

2) Procesado de los datos. En esta fase se realiza la manipulación necesaria de los datos con el objetivo de conseguir información utilizable.

3) Limpieza de los datos. Se eliminan registros erróneos que no aportan ninguna información al proceso.

4) Análisis exploratorio. El objetivo es explorar y analizar un conjunto de datos. Para ello se utilizan diferentes herramientas como lenguajes de programación o herramientas de visualización.

5) Modelado y algoritmos. Etapa en el que se usa la estadística para conseguir la información requerida. Dentro de nuestro proyecto se emplearán las diferentes técnicas de *Machine Learning* que se explican más adelante.

6) Producto de datos. En esta etapa se pretende dar forma al resultado obtenido

7) Visualización. Por último, hay que visualizar el resultado final del análisis.

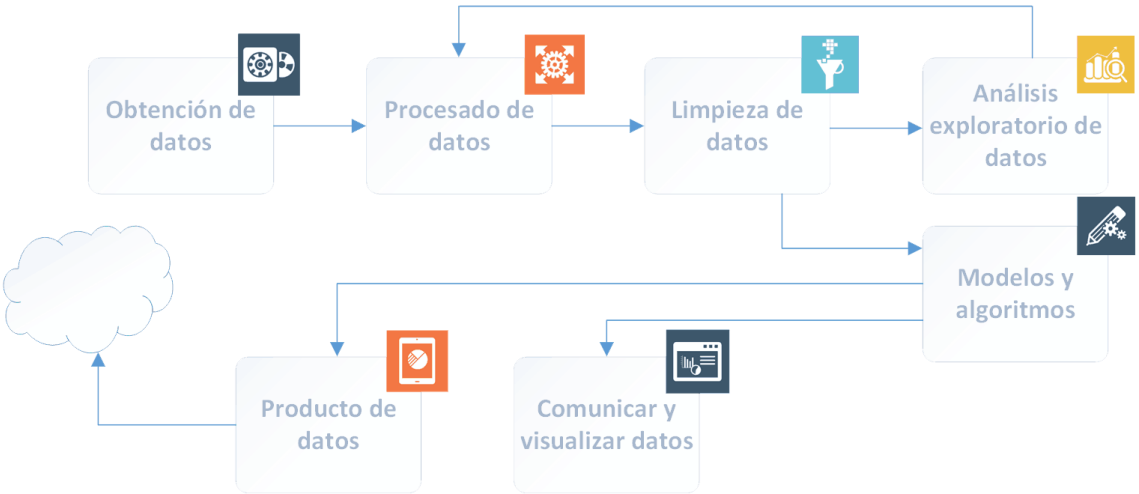


**Ilustración 5: Etapas del análisis de datos [13]**

Poniendo énfasis directamente en el proyecto, el origen de la información de donde se obtiene el *dataset* del tráfico móvil será una base de datos relacional externa. A partir de una serie de consultas utilizando lenguaje SQL se exporta un fichero CSV con los datos de tráfico móvil que se utilizan en el proyecto.

Posteriormente para la manipulación, análisis y visualización de datos se importan en Python usando una serie de herramientas que se explican a continuación.

### 2.3.1 Bases de datos

Básicamente consisten en un conjunto de información relacionada y estructurada y en unos programas, que se denominan sistemas de gestión de base de datos. Estos sistemas permiten la interacción de los usuarios y programas para facilitar la obtención de la información guardada en la base de datos [14].

Hay varios tipos de bases de datos según estas organicen y permitan el acceso a la información. Hoy en día, básicamente se dividen en dos grupos:

- Bases de datos relacionales.
- Bases de datos no relacionales, también llamadas NoSQL.

Como se ha mencionado antes en este proyecto se obtienen los datos del tráfico móvil de una fuente externa localizada en una base de datos relacional.

Las bases de datos relacionales son las más extendidas en la actualidad y han dominado este campo desde la década de los 70 en la que aparecieron en el mercado. Grandes empresas han desarrollado sistemas de gestión de bases de datos relacionales, como Oracle [15] o SQL Server [16]. En los últimos años también han aparecido SGBD de código libre (*Open Source*) cuyo uso se está extendiendo rápidamente como por ejemplo MySQL [17].

El modelo de datos relacional organiza y representa los datos en forma de tablas o relaciones. Una base de datos relacional es un conjunto de relaciones (tablas).

El lenguaje utilizado en un modelo relacional es el llamado SQL (*Structured Query Language*) [18]. A partir de una serie de sentencias, utilizando este lenguaje se podrán realizar operaciones sencillas, tanto para definir como para manejar los elementos de la base de datos:

- Sentencias DDL (*Data Definition Language)*: Son las sentencias que permiten la creación de los elementos de la base de datos, como por ejemplo crear las tablas, los índices, los usuarios, los permisos sobre los elementos, etc.

- Sentencia DML (*Data Management Language)*. Estas sentencias permiten manejar los datos dentro de una base de datos, como por ejemplo insertar, modificar, borrar valores en una tabla, seleccionar valores de una tabla, etc.…

En las bases de datos orientadas al análisis, las sentencias DML más utilizadas son las que ejecutan consultas. Para ello se utiliza la palabra clave SELECT, la cual es interpretada por el SGBD como un comando de consulta.

Dentro de la sentencia SELECT se detallan por un lado los datos que se quiere consultar, principalmente campos de tablas almacenadas en la base de datos y por otro, donde encontrarlos mediante la palabra clave FROM.

Para extraer el dataset se utiliza este tipo de consultas.

### 2.3.2 Python

Dentro del proyecto el lenguaje que se utiliza para el manejo y análisis del dataset y posteriormente para las tareas de predicción y visualización de los resultados será Python [19].

Se trata de un lenguaje de programación orientado a objetos en la que cobra especial importancia la legibilidad del código. Hoy en día es el lenguaje más utilizado ya que abarca todo tipo de programas desde aplicaciones Windows o servidores hasta aplicaciones web.

Una de las ventajas que ofrece Python es que es un lenguaje interpretado, es decir no es necesario compilar el código para poder ejecutarlo. Además, es gratuito y contiene infinidad de librerías que permiten ahorrar tiempo en la programación.

En concreto las más utilizadas para el manejo y análisis de datos son las siguientes:

- Pandas: se trata de una librería de código abierto. Su función es tomar como entrada un conjunto de datos o dataset, normalmente en formato CSV, creando un objeto en Python con filas y columnas llamado *dataframe.* Se trata de una tabla que permite operar sobre sus registros con gran facilidad.

- NumPy: es la librería de Python especializada en el cálculo numérico y análisis de datos por excelencia, y más si se trabaja con grandes cantidades de datos. Proporciona potentes estructuras de datos pudiendo crear matrices multidimensionales.

- Matplotlib: es la librería encargada de la visualización en Python. Permite generar gráficos a partir de *arrays* o listas.

Estas tres librerías son las que se usan principalmente en el proyecto para la manipulación y el análisis de datos. En el siguiente capítulo se introduce la tecnología Machine Learning y sus principales herramientas las cuales permitirán realizar el análisis predictivo.

## 2.4 *Machine Learning*

Para entender que es el *Machine Learning* hay que empezar definiendo que es la Inteligencia Artificial (IA). A muy alto nivel podría definirse como la ciencia que permite a una máquina adquirir una serie de conocimientos que le lleven a poder pensar como un ser humano.

El Machine Learning o aprendizaje automático es por tanto una rama de la informática que otorga a la IA la capacidad de aprendizaje. Se trata de un subconjunto de la IA donde las personas entrenan a las máquinas para reconocer patrones basados en datos y realizar sus predicciones. Esto proporciona a las máquinas la capacidad de resolver problemas por sí mismas sin necesidad de ser programadas [20].

En *Machine Learning* se utilizan algoritmos ya programados que reciben y analizan los datos de entrada que provienen del *dataset* origen para predecir los valores, siempre dentro de un rango aceptable. Cada vez que se introducen nuevos datos, estos algoritmos aprenden mejorando su rendimiento y adquiriendo inteligencia. Existen dos tipos diferentes de algoritmos *Machine Learning* [21]:

- Aprendizaje supervisado: tiene como objetivo principal hacer predicciones a futuro basadas en comportamientos que se han visto en el histórico de datos. Permite encontrar patrones relacionando todos los campos con un campo objetivo.

- Aprendizaje sin supervisión: el algoritmo interpreta grandes cantidades de datos intentando descubrir su estructura. Para ello intenta organizarlos en grupos. Cada vez que interpreta más datos su capacidad para tomar decisiones mejorar gradualmente. La gran diferencia con respecto al aprendizaje supervisado es que los datos no están etiquetados.

En este proyecto se utilizan los algoritmos de aprendizaje más comunes mediante Python. Para ello existe una librería de código abierto llamada Scikit Learn que proporciona dichos algoritmos.

Existen distintos tipos de algoritmos predictivos: los algoritmos de clasificación, los cuales permiten predecir la pertenencia a una clase y los algoritmos de regresión que permiten predecir un valor. En los siguientes apartados se explica brevemente los más importantes.

### 2.4.1 Regresión Lineal

La regresión lineal simple representa la relación existente entre una variable dependiente y su respectiva variable independiente a partir de una función lineal.

Adopta la forma y=ax+b, siendo y la variable dependiente, x la independiente y a y b los coeficientes de la recta [22]. El algoritmo debe minimizar el coste de una función de error cuadrático medio y esos coeficientes representarán la recta óptima. Este método consiste en sumar las distancias al cuadrado de los diferentes puntos que conforman los datos y los puntos de la línea que genera el modelo. La mejor estimación será por tanto la que minimice estas distancias.

El resultado sería algo así:

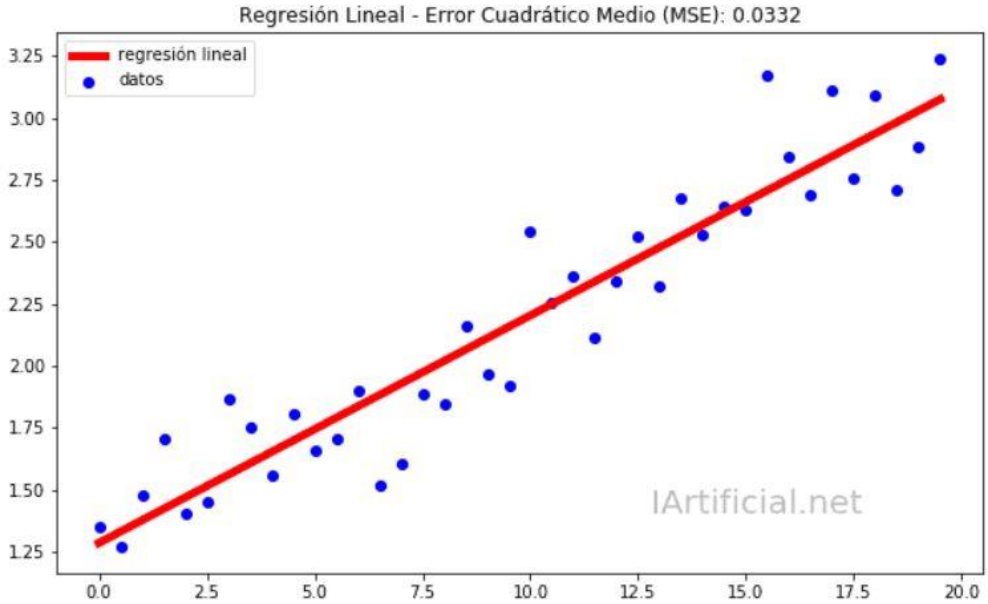


**Ilustración 6: Regresión Lineal**

### 2.4.2 Redes neuronales

Una red neuronal es una red de entidades interconectadas conocidas como nodos en la que cada nodo es responsable de un cálculo simple. Funciona de forma parecida a las neuronas del cerebro humano formándose conexiones entre los diferentes nodos [23].

### 2.4.3 K-vecinos más cercanos

Es un algoritmo de agrupamiento (*clustering*). Utiliza métodos estadísticos de reconocimiento de patrones para calcular la distancia de un dato a sus vecinos más cercanos dentro del conjunto de entrenamiento. No es realmente efectivo con pequeñas muestras. Al ser un algoritmo de clasificación el resultado será la probabilidad de que un elemento pertenezca o no a una clase [24].

### 2.4.4 Árboles de decisión

Consiste en un algoritmo predictivo que se utiliza para predecir la probabilidad de conseguir un determinado resultado en base a ciertas condiciones. Así se divide el espacio muestral en subregiones aplicando una serie de reglas para que cada subregión contenga la mayor proporción de individuos en uno de los hijos [25].

El tipo de problema dependerá de la variable a predecir siendo un problema de regresión si la variable es dependiente o de clasificación si esta es categórica.

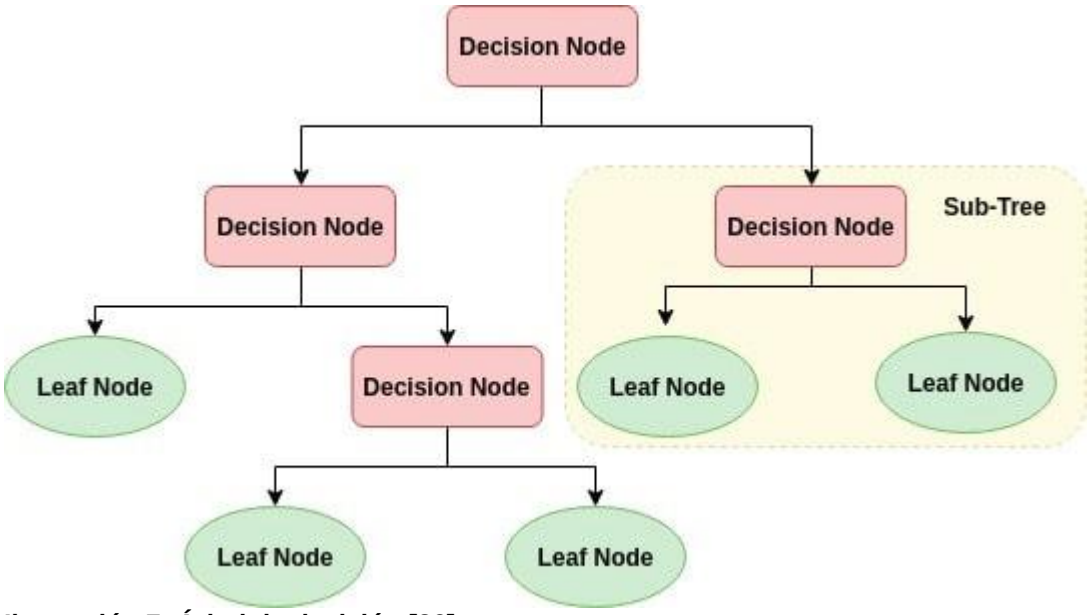


**Ilustración 7: Árbol de decisión [26]**

### 2.4.5 Algoritmos Bayesianos

Se basan en el teorema de Bayes [27]. Usando modelos probabilísticos predicen la pertenencia a una clase o categoría a partir de un conjunto dado, realizando clasificaciones de cada valor independientemente de los otros [28].

### 2.4.6 Algoritmos de agrupación

Se usan dentro del aprendizaje no supervisado con el objetivo de categorizar datos no etiquetados. Mediante una exploración de los datos identifican diferentes grupos representados con la variable K y de forma iterativa prosiguen hasta asignar cada dato a un grupo en función de sus características [29].

## 2.5 Herramientas de visualización de mapas

Para la representación de las diferentes coordenadas y mapas de calor se utiliza el módulo Folium [30] de Python. Se trata de una herramienta que permite manipular datos y visualizarlos en Leaflet [31].

Leaflet es una biblioteca de JavaScript [32] de código abierto que se utiliza para crear aplicaciones de mapas web. Esta herramienta permite crear los diferentes mapas para representar los puntos geográficos y además crear los mapas de calor sobre la afluencia de gente. Los mapas utilizados proceden de la aplicación *Open Street Map* [33].

**Ilustración 8: Mapa de calor con Leafset**

## 2.6 Proyectos relacionados

Estudiar el movimiento de las personas y descubrir que zonas presentan mayor acumulación es uno de los proyectos más solicitados. El problema, sin embargo, será el acceso a los datos, de ahí que sean tan importantes y que muchas empresas lleguen a pagar una gran cantidad por ellos.

Una de las organizaciones que tiene acceso a ellos es el Instituto Nacional de Estadística (INE) [34] gracias a su estrecha relación con los principales operadores de telefonía móvil. Se trata de un organismo autónomo cuyas funciones principales son la coordinación general de los servicios estadísticos de la Administración General del Estado [35] y que se encuentra dentro del Ministerio de Asuntos Económicos y Transformación Digital [36]. Uno de sus proyectos más recientes ha sido el estudio de la movilidad de la población durante el estado de alarma [37].

Para su análisis utilizaron datos de alrededor del 80% de los teléfonos móviles de toda España. El objetivo del proyecto era analizar la movilidad de las personas durante el Estado de Alarma con respecto a un día normal. Los resultados que ofrecieron abarcaban todo el territorio nacional con un total de 3200 áreas de movilidad. Para la comparación se consideró un día normal como la media de una serie de días de 2019.

Para cada área de movilidad la información relevante resultante fue:

- Población residente del área.
- Número de personas que se estima que se quedan en su área de residencia.
- Número de personas que se ausentan de su área de residencia.

- Los principales destinos a los que se dirigen los que salen de su área y el número de personas que se desplazan a ellos.

El área de residencia será la zona donde el teléfono ha pasado la mayor parte del tiempo en el horario de 0 a 6 horas. Se establece como área destino, la superficie donde se detecta el teléfono durante el periodo de 10 a 16 horas siempre y cuando haya estado en ella al menos durante dos horas. Así los operadores proporcionan, para cada área de residencia, el número de movimientos a las áreas de destino si este número supera las 10 unidades, para proteger la privacidad y respetar el secreto estadístico.

A continuación, se observa el resultado para un área de movilidad en concreto.

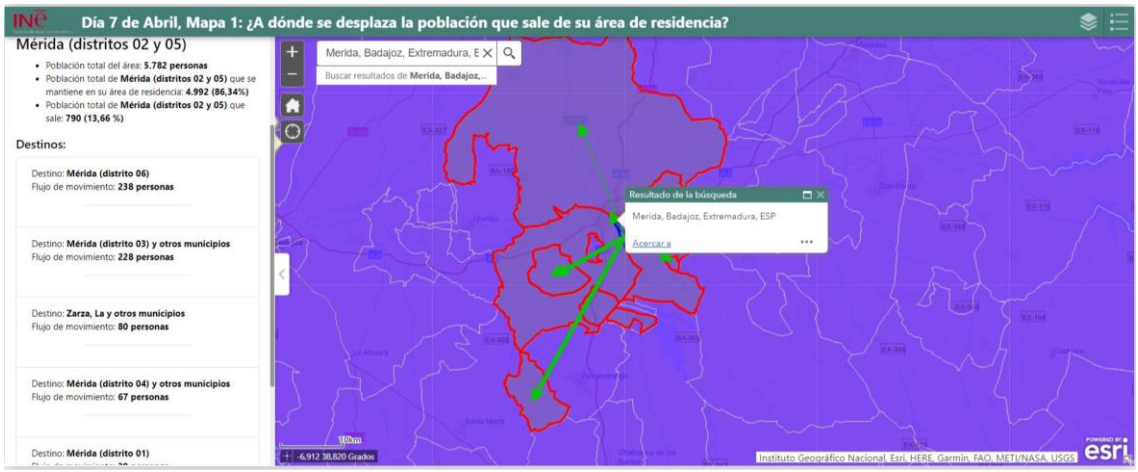

**Ilustración 9: Movilidad de la población durante el Estado de Alarma [38]**

En concreto, para Mérida (Badajoz) el 7 de abril de 2020. En torno al 86,34% de personas se quedaron en su residencia y el restante 13,66% sale de ella. También se pueden apreciar en el mapa las principales áreas de destino.

# 3. Descripción de la solución propuesta

El desarrollo del proyecto se dividirá en tres etapas:

- Proceso de ETL: etapa en la cual se extraen los datos de la fuente origen y se transforman para que puedan ser utilizados en el sistema deseado.

- Análisis de Datos: se lleva a cabo un estudio profundo de los datos con el objetivo de extraer conclusiones.

- Algoritmos predictivos de Machine Learning: se lleva a cabo un estudio de los algoritmos más convenientes para el conjunto de datos utilizado con el objetivo de realizar predicciones.

A continuación, se explicará minuciosamente cada etapa.

## 3.1 Proceso de ETL

El proceso de extracción y transformación de los datos se llevará a cabo en Oracle SQL Developer [39] un entorno de desarrollo integrado para trabajar con SQL en bases de datos Oracle. Para cargar los datos en el sistema deseado se empleará Python.

El origen de los datos es confidencial por lo que no se aportará información alguna sobre el mismo.

La información en este entorno está organizada en tablas que se relacionan entre ellas gracias a claves primarias que permiten hacer únicos cada uno de los registros de las tablas. Para llevar a cabo estas relaciones en SQL se emplean los denominados cruces. En el proceso de ETL utilizaremos el LEFT JOIN.

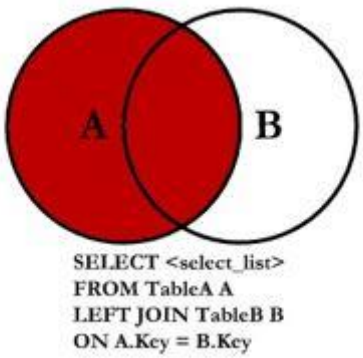


**Ilustración 10: LEFT JOIN SQL [40]**

La información utilizada en el proyecto se encuentra en dos tablas diferentes. Por un lado, los datos de tráfico móvil con todos los registros de conexión y por otro lado las estaciones base que recogen estos datos.

Es importante destacar el gran volumen de datos que contiene la tabla de tráfico móvil al abarcar todo el territorio español, con miles de millones de registros por día. Por esta razón y debido a la gran dificultad que supone su manipulación se ha decidido limitar la extracción a una ciudad y en diferentes días del año.

Como uno de los objetivos del proyecto es analizar el impacto de las restricciones COVID y que ha supuesto el fin del Estado de Alarma, la ciudad elegida no será un destino turístico estival. Así se evita que el posible aumento o disminución de la movilidad tenga esta causa. Además, la ciudad elegida tiene que ser turística independientemente de la época del año para así captar adecuadamente la afluencia de personas.

Por lo tanto, se decide centrar el estudio en la ciudad de Mérida, capital de la comunidad autónoma de Extremadura, a tan solo tres horas de Madrid. Al ser la capital albergará también instituciones importantes de la comunidad.

Para la elección de los días se tuvieron en cuenta dos aspectos. Uno la situación meteorológica con el fin de excluir días con condiciones adversas que afectarán al movimiento de personas. Para ello se ha utilizado una página web que almacena el histórico del clima por días llamada Climate Data [41]. Por otro lado, que se pudiera apreciar diferencia entre un día laboral (lunes a viernes) y fin de semana.

Con todo esto en mente se decide extraer información de 4 días:

- Miércoles 21/04/2021: clima despejado con buena temperatura en la que la comunidad autónoma se encuentra cerrada.

- Miércoles 26/05/2021: clima despejado con buena temperatura en la que la comunidad autónoma se encuentra abierta tras el fin del Estado de Alarma.

- Sábado 24/04/2021: clima despejado con buena temperatura en la que la comunidad autónoma se encuentra cerrada.

- Sábado 22/05/2021: clima despejado con buena temperatura en la que la comunidad autónoma se encuentra abierta tras el fin del Estado de Alarma.

A continuación, se explican en detalle las tres etapas llevadas a cabo en el proceso de ETL.

### 3.1.1 Proceso de extracción y limpieza de los datos

La primera parte de esta etapa será analizar los datos de las estaciones base. La base de datos que alberga esta información se llama *datatraffic* y el nombre de la tabla *bts_cells.*

Como solo se necesita la información de Mérida se realiza una consulta a la tabla filtrando por el nombre de la ciudad para así averiguar el identificador único asociado.

```
--CODIGO DE MUNICPIO PARA MERIDA
select distinct id_municipio
from datatraffic.bts_cells
where municipio = 'MERIDA'
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 1 en 2,01 segundos

| | ID_MUNICIPIO |
|---|---|
| 1 | 5005 |

**Ilustración 11: Identificador único del municipio**

Una vez localizado el código único del municipio será necesario realizar una limpieza de los datos de la tabla con el objetivo de recopilar solo la información más importante para el proyecto. Así de todos los campos existentes se seleccionan los siguientes.

```
--INFORMACIÓN RELEVANTE DE CADA ESTACIÓN BASE
select id_celula, cast(fecha_alta AS DATE) as fecha_alta, codigo_postal, id_municipio, municipio, id_provincia,
provincia,  id_com_autonoma,  com_autonoma, latitud, longitud
from datatraffic.bts_cells
where baja is null and id_munipio = 5005
```

Resultado de la Consulta | Resultado de la Consulta 1

SQL | Se han recuperado 50 filas en 0,084 segundos

| | ID_CELULA | FECHA_ALTA | CODIGO_POSTAL | ID_MUNICIPIO | MUNICIPIO | ID_PROVINCIA | PROVINCIA | ID_COM_AUTONOMA | COM_AUTONOMA | LATITUD | LONGITUD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 171810 | 10/12/17 | 06800 | 5005 | MÉRIDA | 6 | BADAJOZ | 5 | EXTREMADURA | 38,92970757 | -6,33276253 |
| 2 | 1352839 | 01/10/19 | 06800 | 5005 | MERIDA | 6 | Badajoz | 5 | Extremadura | 38,9140993 | -6,33271694 |

**Ilustración 12: Información relevante de las estaciones base**

Se puede apreciar en la consulta una conversión a fecha de un campo que previamente tenía fecha y hora. En los filtros se puede ver que el campo baja tiene que ser nulo para así seleccionar solo las estaciones base operativas. Los campos seleccionados serán principalmente los nombres y códigos del municipio, provincia y comunidad autónoma además de las coordenadas de latitud y longitud. Además, se pueden apreciar el identificador único de la estación base llamado *id_celula* y la fecha de alta de esta. Este identificador nos permitirá más adelante realizar el cruce con la tabla de tráfico móvil.

Como realmente para el proyecto solo se necesitan las coordenadas de latitud y longitud de cada estación base, realizaremos un conteo con el objetivo de calcular el número de estaciones base que contiene la ciudad de Mérida.

```
--CAMPOS RELEVANTES PARA EL PROYECTO:LATITUD Y LONGITUD
select distinct latitud, longitud
from datatraffic.bts_cells
where baja is null and id_munipio = 5005
```

Resultado de la Consulta | Resultado de la Consulta 1

SQL | Todas las Filas Recuperadas: 31 en 0,281 segundos

| | LATITUD | LONGITUD |
|---|---|---|
| 1 | 38,91839464 | -6,34922036 |
| 2 | 38,89430584 | -6,39329144 |
| 3 | 38,92742437 | -6,33785753 |
| 4 | 38,91785431 | -6,35836258 |
| 5 | 38,91248564 | -6,36357801 |
| 6 | 38,92226409 | -6,33124609 |

Recuento...

31 Filas

Aceptar

**Ilustración 13: Número de estaciones base diferentes en Mérida**

En la ilustración anterior se puede ver que el número de estaciones bases es de 31. La localización exacta de cada estación base se representará en el apartado de análisis de datos.

El siguiente paso es hacer lo mismo con la tabla de tráfico móvil llamada *mobile_traffic* que se encuentra en la misma base de datos que la anterior. Como se ha comentado previamente se trata de una tabla de gran volumen por lo que las consultas suelen ser bastante pesadas y utilizan una gran cantidad de recursos.

De esta tabla se necesitará la fecha y hora exacta en la que se produce la conexión para poder analizar los diferentes tramos horarios. Además, es necesario el identificador único de la estación base que recoge la información, para poder así relacionarla con la tabla *bts_cells.*

Como se pretende analizar la movilidad, el resto de información de la tabla se excluye para así reducir el número de campos y agilizar las etapas posteriores.

La consulta que relaciona ambas tablas y se queda con la información necesaria es la siguiente:

```
--INFORMACION FINAL DE LOCALIZACION
select cel.latitud, cel.longitud,  tr.fecha_registro
from datatraffic.mobile_traffic tr
left join datatraffic.bts_cells cel on tr.id_celula = cel.id_celula
where tr.dia_registro = to_date('22/05/2021' , 'dd/mm/yyyy')
and cel.id_municipio = 5005
```

Resultado de la Consulta

Se han recuperado 50 filas en 0,235 segundos

| | LATITUD | LONGITUD | FECHA_REGISTRO |
|---|---|---|---|
| 1 | 38,89430584 | -6,39329144 | 22/05/21 11:32:38,000000000 |
| 2 | 38,89430584 | -6,39329144 | 22/05/21 15:05:46,000000000 |

**Ilustración 14: Información final a exportar**

Se aprecia el cruce de información comentado anteriormente llamado *left join* a través del identificador único de estación base llamado *id_celula.* Además, en el filtro aparece un solo día y el identificador único del municipio.

Esta consulta, con los datos ya limpiados adecuadamente, es la que se utilizará en el siguiente apartado para llevar a cabo las diferentes exportaciones.

### 3.1.2 Proceso de exportación de datos

Una vez finalizado el proceso de extracción y limpieza de los datos habrá que ver cuál es la manera más eficiente y rápida para llevar a cabo su exportación. Como se ha mencionado anteriormente es necesario extraer los datos del tráfico móvil de cuatro días diferentes.

En un primer momento se decidió llevar a cabo una exportación manual de las diferentes consultas finales a través del entorno Oracle SQL Developer. Esta opción era prácticamente inviable ya que dicho entorno no está preparado para exportaciones y consultas de tanto volumen. Se realizaron pruebas y no se llegó a finalizar ninguna exportación tras horas de espera.

Tras explorar otras maneras más eficaces se decidió dividir la información en cuatro tablas más pequeñas cada una con la información del tráfico móvil de los días establecidos. Esto permite, una vez creadas las tablas, que las exportaciones sean muchísimo más rápidas al atacar estructuras más pequeñas. Lo único costoso aquí será por tanto la creación de estas tablas, pero nada en comparación a las exportaciones manuales.

Para agilizar el proceso se decide utilizar una herramienta externa llamada SQL PLUS [42]. Se trata de un programa de línea de comandos de Oracle que permite ejecutar consultas SQL y automatizarlas en un script. Esto va a permitir que se lleve a cabo tanto las consultas de creación como las exportaciones sin necesidad de estar pendientes del programa únicamente lanzando un archivo .bat [43] de llamada a las tareas de SQL PLUS.

A continuación, se muestran las consultas de creación de las diferentes tablas usadas en el proceso de exportación.

```
--MERIDA 24/04/2021 SABADO DESPEJADO BUENA TEMPERATURA COMUNIDAD AUTONOMA CERRADA
create table registros_240421 as (
select cel.latitud AS LATITUD, cel.longitud AS LONGITUD,  tr.fecha_registro as IT_REGISTRO
from datatraffic.mobile_traffic tr
left join  datatraffic.bts_cells cel on tr.id_celula = cel.id_celula
where tr.dia_registro = to_date('24/04/2021' , 'dd/mm/yyyy')
and cel.id_municipio = 5005
);

commit;

--MERIDA 22/05/2021 SABADO DESPEJADO BUENA TEMPERATURA COMUNIDAD AUTONOMA ABIERTA
create table registros_220521 as (
select cel.latitud AS LATITUD, cel.longitud AS LONGITUD,  tr.fecha_registro as IT_REGISTRO
from datatraffic.mobile_traffic tr
left join  datatraffic.bts_cells cel on tr.id_celula = cel.id_celula
where tr.dia_registro = to_date('22/05/2021' , 'dd/mm/yyyy')
and cel.id_municipio = 5005
);

commit;

--MERIDA 21/04/2021 MIERCOLES DESPJEADO BUENA TEMPERATURA COMUNIDAD AUTONOMA CERRADA
create table registros__210421 as (
select cel.latitud AS LATITUD, cel.longitud AS LONGITUD,  tr.fecha_registro as IT_REGISTRO
from datatraffic.mobile_traffic tr
left join  datatraffic.bts_cells cel on tr.id_celula = cel.id_celula
where tr.dia_registro = to_date('21/04/2021' , 'dd/mm/yyyy')
and cel.id_municipio = 5005
);

commit;

--MERIDA 26/05/2021 MIERCOLES DESPEJADO BUENA TEMPERATURA COMUNIDAD AUTONOMA ABIERTA
create table registros_260521 as (
select cel.latitud AS LATITUD, cel.longitud AS LONGITUD,  tr.fecha_registro as IT_REGISTRO
from datatraffic.mobile_traffic tr
left join  datatraffic.bts_cells cel on tr.id_celula = cel.id_celula
where tr.dia_registro = to_date('26/05/2021' , 'dd/mm/yyyy')
and cel.id_municipio = 5005
);

commit;
```

**Ilustración 15: Sentencias de creación de tablas para exportación**

Se puede observar que la sentencia de creación de las tablas procede de la consulta final del apartado 3.1.1.

Una vez creados comprobamos el número de registros que tiene cada una.

| | TABLA | REGISTROS |
|---|---|---|
| 1 | registros_210421 | 1743915 |
| 2 | registros_220521 | 1553492 |
| 3 | registros_240421 | 1338863 |
| 4 | registros_260521 | 1868216 |

**Ilustración 16: Número de registros por tabla**

Para reducir más aún el número de registros se fija el tramo horario de las 12:00 a las 14:00 horas como el más relevante ya que es el que presenta mayor actividad y el que aportará más información al análisis de datos posterior.

Ahora es el momento de comenzar con la exportación.

```
--EXPORTACION DE TABLAS VIA SQL PLUS.
SET HEADING OFF
SET timing OFF
SET ECHO OFF
SET VERIFY OFF
SET FEEDBACK OFF
SET LINESIZE 10000
SET PAGESIZE 0
SET NEWPAGE 0
SET SPACE 0
SET TRIMSPOOL ON
SET TERMOUT OFF
SET ARRAYSIZE 200
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
--INICIO EXPORTACION 24/04/21
SPOOL I:\Procesos_Exadata\VARIOS\OTROS\datos240421.csv
SELECT 'LATITUD' || ';' || 'LONGITUD' || ';' || 'IT_REGISTRO' FROM DUAL
UNION ALL
SELECT LATITUD  ||';'|| LONGITUD    ||';'||TO_CHAR(IT_REGISTRO,'dd/mm/yy hh24:mi:ss')
from registros_240421
where IT_REGISTRO between TO_TIMESTAMP('24/04/21 12:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF')
and TO_TIMESTAMP('24/04/21 14:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF');
SET FEEDBACK ON
SET HEADING ON
SPOOL OFF
--INICIO EXPORTACION 22/05/21
SPOOL I:\Procesos_Exadata\VARIOS\OTROS\datos220521.csv
SELECT 'LATITUD' || ';' || 'LONGITUD' || ';' || 'IT_REGISTRO' FROM DUAL
UNION ALL
SELECT LATITUD  ||';'|| LONGITUD    ||';'||TO_CHAR(IT_REGISTRO,'dd/mm/yy hh24:mi:ss')
from registros_220521
where IT_REGISTRO between TO_TIMESTAMP('22/05/21 12:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF')
and TO_TIMESTAMP('22/05/21 14:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF');
SET FEEDBACK ON
SET HEADING ON
SPOOL OFF
--INICIO EXPORTACION 21/04/21
SPOOL I:\Procesos_Exadata\VARIOS\OTROS\datos210421.csv
SELECT 'LATITUD' || ';' || 'LONGITUD' || ';' || 'IT_REGISTRO' FROM DUAL
UNION ALL
SELECT LATITUD  ||';'|| LONGITUD    ||';'||TO_CHAR(IT_REGISTRO,'dd/mm/yy hh24:mi:ss')
from registros_210421
where IT_REGISTRO between TO_TIMESTAMP('21/04/21 12:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF')
and TO_TIMESTAMP('21/04/21 14:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF');
SET FEEDBACK ON
SET HEADING ON
SPOOL OFF
--INICIO EXPORTACION 26/05/21
SPOOL I:\Procesos_Exadata\VARIOS\OTROS\datos260521.csv
SELECT 'LATITUD' || ';' || 'LONGITUD' || ';' || 'IT_REGISTRO' FROM DUAL
UNION ALL
SELECT LATITUD  ||';'|| LONGITUD    ||';'||TO_CHAR(IT_REGISTRO,'dd/mm/yy hh24:mi:ss')
from registros_260521
where IT_REGISTRO between TO_TIMESTAMP('26/05/21 12:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF')
and TO_TIMESTAMP('26/05/21 14:00:00.000000', 'DD-MM-RR HH24:MI.SS.FF');
SET FEEDBACK ON
SET HEADING ON
SPOOL OFF
SET HEADING ON
SET FEEDBACK ON
SET TERMOUT ON
QUIT
```

**Ilustración 17: Proceso de exportación**

En la imagen anterior se observa que para fijar un tramo horario se ha filtrado convirtiendo un campo texto con la hora exacta a un campo de tipo *Timestamp* (fecha con hora, minutos y segundos) igual que el campo de la tabla *IT_REGISTRO*. El resto de los parámetros que aparecen en el script son los estándares en un proceso de exportación de Oracle.

La información resultante será almacena en ficheros CSV, cada uno con los datos del tráfico móvil de cada día en función del tramo horario seleccionado.

| | FICHERO | REGISTROS |
|---|---|---|
| 1 | datos210421.csv | 245529 |
| 2 | datos220521.csv | 253704 |
| 3 | datos240421.csv | 219119 |
| 4 | datos260521.csv | 255205 |

**Ilustración 18: Número de registros de cada fichero**

Finalmente, se logra reducir considerablemente el número de registros por lo que, se puede dar por finalizado el proceso de exportación. Únicamente quedaría cargar estos datos en el entorno de Python donde se utilizarán más adelante.

### 3.1.3 Proceso de carga de datos

Este proceso es bastante rápido gracias a que el lenguaje de programación de Python posee una sintaxis simple tal y como se mencionó en el Estado del Arte del proyecto.

El procedimiento es el siguiente.

```
In [2]:
import pandas as pd
import numpy as np

#CARGAMOS LOS DATOS DE TRAFICO MOVIL
#MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file210421 = '../DATOS\datos210421.csv'
datos210421 = pd.read_csv(file210421, sep=';', engine='python')
datos210421['IT_REGISTRO'] = pd.to_datetime(datos210421['IT_REGISTRO'])

#SABADO 24 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file240421 = '../DATOS\datos240421.csv'
datos240421 = pd.read_csv(file240421, sep=';', engine='python')
datos240421['IT_REGISTRO'] = pd.to_datetime(datos240421['IT_REGISTRO'])

#SABADO 22 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO
file220521 = '../DATOS\datos220521.csv'
datos220521 = pd.read_csv(file220521, sep=';', engine='python')
datos220521['IT_REGISTRO'] = pd.to_datetime(datos220521['IT_REGISTRO'])

#MIERCOLES 26 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO
file260521 = '../DATOS\datos260521.csv'
datos260521 = pd.read_csv(file260521, sep=';', engine='python')
datos260521['IT_REGISTRO'] = pd.to_datetime(datos260521['IT_REGISTRO'])
```

**Ilustración 19: Carga de datos en Python**

Como se contempla en la ilustración anterior se crean cuatro *dataframes* distintos a partir de los datos de los ficheros CSV importados mediante pandas. Estos *dataframes* serán la base de los siguientes apartados de análisis de datos y *Machine Learning.*

Hay que destacar que se ha elegido la aplicación Jupyter Notebook [44] para trabajar con Python ya que es gratuita, es de código abierto, presenta código en vivo y funciona en el navegador entre otras muchas ventajas.

## 3.2 Análisis de datos

### 3.2.1 Introducción

Una vez completado el proceso de extracción, transformación y carga de los datos se procede a realizar un análisis profundo del mismo.

En primer lugar, se va a mostrar un ejemplo del *dataset* de partida. En concreto del *dataset* llamado *datos210421,* que contiene tal y como se ha explicado anteriormente los datos de tráfico móvil del tramo de las 12:00 a las 14:00 horas del día 21 de abril de 2021, en la ciudad española de Mérida. El objetivo es que sirva de introducción para su posterior estudio.

```
In [20]:  datos210421.sample(5)
```

Out[20]:

| | LATITUD | LONGITUD | IT_REGISTRO |
|---|---|---|---|
| **24906** | 38.913776 | -6.337503 | 2021-04-21 12:51:32 |
| **119507** | 38.919097 | -6.349978 | 2021-04-21 12:03:08 |
| **126612** | 38.894306 | -6.393291 | 2021-04-21 13:02:34 |
| **103769** | 38.905467 | -6.361947 | 2021-04-21 13:40:18 |
| **196717** | 38.927756 | -6.334627 | 2021-04-21 12:56:13 |

Ilustración 20: Muestra aleatoria de del dataset de partida

Las variables presentes en el dataset son las coordenadas geográficas de las estaciones base que captan las conexiones de los dispositivos móviles. Aparecen bajo el nombre de LATITUD y LONGITUD. Por otra parte, la variable IT_REGISTRO está directamente relacionada, al tratarse del momento exacto en el que se produce la conexión. Al haber realizado el correspondiente proceso de limpieza, no existen valores anómalos presentes en el dataset.

```
In [22]:  datos210421.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245529 entries, 0 to 245528
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   LATITUD      245529 non-null  float64
 1   LONGITUD     245529 non-null  float64
 2   IT_REGISTRO  245529 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(2)
memory usage: 5.6 MB
```

**Ilustración 21: Información de los campos del dataset de partida**

```
In [21]:  datos210421.describe()
```

Out[21]:

| | LATITUD | LONGITUD |
|---|---|---|
| **count** | 245529.000000 | 245529.000000 |
| **mean** | 38.918056 | -6.350364 |
| **std** | 0.011826 | 0.016242 |
| **min** | 38.894306 | -6.393291 |
| **25%** | 38.912486 | -6.359937 |
| **50%** | 38.917666 | -6.348064 |
| **75%** | 38.922264 | -6.337858 |
| **max** | 38.973696 | -6.311123 |

**Ilustración 22: Estadísticas de los campos del dataset de partida**

```
In [32]:  datos210421['IT_REGISTRO'].mean()

Out[32]: Timestamp('2021-04-21 13:00:06.234274304')
```

**Ilustración 23: Media horaria de los dataset de partida**

Hay que recordar que el resto de *datasets* que se utilizarán en esta fase presentan la misma estructura mostrada anteriormente, únicamente variando el día de la muestra.

Hay que destacar que la media del campo fecha y hora llamado *IT_REGISTRO* es exactamente el punto medio. Se puede apreciar detalles importantes como los puntos máximos y mínimos de localización (coordenadas de latitud y longitud), el número total de registros de entrada y el tipo de campos que lo forman.

### 3.2.2 Descripción del proceso

La siguiente parte va a consistir en representar los diferentes puntos geográficos en un mapa, analizarlos e intentar sacar conclusiones.

Como tenemos cuatro *datasets* diferentes cada uno con diferentes días, se va a proceder a su análisis por parejas. En un primer momento se estudiarán los que contengan los dos días intersemanales, ambos miércoles en diferentes meses, uno en abril y otro en mayo. La principal diferencia entre ambos es que en el primero aún estaba vigente el Estado de Alarma y en el segundo no.

Para ello se empieza agrupando los datos de entrada cargados en el *dataframe* por los campos *LATITUD y LONGITUD.* Una vez agrupados, se realiza un conteo del número de registros en función de cada par de coordenadas y se crea una lista con el resultado.

En la siguiente ilustración se puede apreciar este conteo, concretamente el número de registros en el tramo horario seleccionado del miércoles 21 de abril de 2021.

```
In [47]: lista210421

Out[47]: [[38.89430584, -6.39329144, 10147.0],
          [38.90453959, -6.33899008, 11891.0],
          [38.90546652, -6.36194695, 8263.0],
          [38.90593161, -6.36698004, 6494.0],
          [38.90929506, -6.35993707, 16430.0],
          [38.91248564, -6.36357801, 11488.0],
          [38.91377623, -6.33750317, 17542.0],
          [38.91391664, -6.34445155, 8350.0],
          [38.9140993, -6.33271694, 14255.0],
          [38.91464693, -6.36114547, 4716.0],
          [38.91661155, -6.34806436, 7644.0],
          [38.91720914, -6.34423645, 2047.0],
          [38.91766563, -6.34647975, 4820.0],
          [38.91781186, -6.35836074, 2177.0],
          [38.91785431, -6.35836258, 3963.0],
          [38.91839464, -6.34922036, 2034.0],
          [38.91859642, -6.34304296, 21392.0],
          [38.91909658, -6.34997843, 10405.0],
          [38.91944515, -6.35144163, 3679.0],
          [38.92074416, -6.37983324, 13287.0],
          [38.92226409, -6.33124609, 7739.0],
          [38.9237298, -6.31112321, 4939.0],
          [38.92484099, -6.34863421, 9465.0],
          [38.92661517, -6.33477369, 1466.0],
          [38.92742437, -6.33785753, 4927.0],
          [38.927756, -6.33462715, 14890.0],
          [38.9297495, -6.33276464, 2.0],
          [38.934875, -6.34691111, 7185.0],
          [38.95059637, -6.35925768, 12780.0],
          [38.97369625, -6.36643186, 1112.0]]
```

**Ilustración 24: Conteo de los registros agrupados**

En la lista anterior aparecen ordenados de menor a mayor los pares de coordenadas de latitud y longitud. El tercer campo será por tanto la suma de los registros totales agrupados. Se realiza el mismo proceso para el resto de los días que intervienen en el análisis.

En total existen cuatro días que hay que analizar detenidamente. El objetivo es poder ver las diferencias que se dan entre los meses de abril y mayo. Como se ha mencionado anteriormente, la gran diferencia radica en la existencia del

Estado de Alarma y el conjunto de restricciones de movilidad que desencadena.

Las restricciones de movilidad afectaron considerablemente al turismo de fin de semana. Mérida es uno de los lugares más visitados dado su gran patrimonio histórico, por lo que el levantamiento del cierre perimetral de la comunidad autónoma de Extremadura permitía de nuevo la llegada de turistas.

Por otra parte, al tratarse de la capital, alberga gran cantidad de organismos públicos y la movilidad durante la semana (lunes a viernes) siempre está garantizada. No obstante, se decide comparar los días intersemanales por una parte y el fin de semana por otra para ver claramente las diferencias en ambos casos.

A continuación, se muestra el proceso que se ha seguido para comparar los dos miércoles de abril y mayo paso por paso. Hay que recalcar que es el mismo procedimiento que se ha usado para la comparación de los sábados, pero solo se mostrará una vez a fin de no ser repetitivos. Ahora bien, el proceso completo con todo el código aparecerá en el anexo 1.

```
#MIERCOLES DE ABRIL VS MIERCOLES DE MAYO
#CONTEO DE NUMERO DE REGISTROS TRAFICO MOVIL POR PAR LATITUD LONGITUD

grouped_data210421 = datos210421.groupby(["LATITUD", "LONGITUD"])

lista210421 = grouped_data210421.count().reset_index().values.tolist()

grouped_data260521 = datos260521.groupby(["LATITUD", "LONGITUD"])

lista260521 = grouped_data260521.count().reset_index().values.tolist()

#PORCENTAJE DE AUMENTO DE REGISTROS DIA SIN RESTRICCIONES VS DIA CON RESTRICCIONES MIERCOLES
#lista_aumento_mi.clear()
lista_aumento_mi = list()
for i in range(len(lista210421)):
    j = 0
    while (lista260521[j][0] != lista210421[i][0]):
        j += 1
        if j == len(lista260521):
            break
    if j != len(lista260521):
        lista_aumento_mi.append([lista260521[j][0], lista260521[j][1], ((lista260521[j][2]/lista210421[i][2])-1)*100])
```

**Ilustración 25: Cálculo de la variación de registros entre dos días**

Primero se ha realizado la agrupación y suma de los registros de ambos días mencionada anteriormente y se ha almacenado dicha información en dos listas llamadas *lista210421* y *lista260521*.

Seguidamente se quiere obtener el porcentaje de aumento de un día con respecto al otro. Como el dato que interesa es ver cómo ha variado la movilidad del mes de mayo en comparación con el mes de abril se dividen los registros de la *lista260521* entre los registros de la *lista210421.* Para ello se crea una nueva lista llamada *lista_aumento_mi* y se recorren las listas anteriores. Para los valores coincidentes de latitud y longitud se inserta un nuevo elemento realizando el cálculo de porcentaje oportuno.

Hay que destacar que el cálculo se realiza solo para los pares de latitud y longitud presentes en ambas listas para así ver el aumento o disminución de

los registros. Así, si existe un nuevo punto geográfico que presenta registros en un mes, pero no en el otro no se tendrá en cuenta para el estudio, ya que se considera nueva estación base.

En la siguiente porción de código se crea una nueva lista llamada *lista_aumento_mi_color*, basada en *lista_aumento_mi,* pero marcando con colores la variación de los registros. El rojo se utilizará para variaciones negativas superiores al 3%, el gris para variaciones estables entre el -3% y el 3% y el verde para variaciones positivas superiores al 3%. La utilización de estos colores permitirá ver con claridad en que puntos han variado los registros.

```
#len(lista_aumento_mi)
#lista_aumento_mi

#lista_aumento_mi_color.clear()
lista_aumento_mi_color = list()
for i in range(len(lista_aumento_mi)):
    if lista_aumento_mi[i][2] > 3:
        color = 'green'
    elif -3 <= lista_aumento_mi[i][2] <= 3:
        color = 'gray'
    else:
        color = 'red'

    lista_aumento_mi_color.append([lista_aumento_mi[i][0], lista_aumento_mi[i][1], lista_aumento_mi[i][2], color])
```

**Ilustración 26: Categorización de la variación de aumento**

En la siguiente imagen se ve el resultado de lo comentado anteriormente. La *lista_aumento_mi_*color contendrá los campos de latitud y longitud, el porcentaje de aumento o disminución de los registros de mayo frente a los de abril y el color que categoriza dicho aumento.

```
In [49]: lista_aumento_mi_color

Out[49]: [[38.89430584, -6.39329144, 8.554252488420232, 'green'],
          [38.90453959, -6.33899008, -0.428895803548901, 'gray'],
          [38.90546652, -6.36194695, -1.8032191697930577, 'gray'],
          [38.90593161, -6.36698004, 10.455805358792736, 'green'],
          [38.90929506, -6.35993707, 0.7486305538648708, 'gray'],
          [38.91248564, -6.36357801, -3.0292479108635084, 'red'],
          [38.91377623, -6.33750317, -0.8208870140234859, 'gray'],
          [38.91391664, -6.34445155, -1.1137724550898231, 'gray'],
          [38.9140993, -6.33271694, -2.006313574184493, 'gray'],
          [38.91464693, -6.36114547, -5.04664970313825, 'red'],
          [38.91661155, -6.34806436, -4.408686551543695, 'red'],
          [38.91720914, -6.34423645, 7.523204689789931, 'green'],
          [38.91766563, -6.34647975, 1.535269709543563, 'gray'],
          [38.91785431, -6.35836258, 42.92202876608631, 'green'],
          [38.91839464, -6.34922036, -17.994100294985248, 'red'],
          [38.91859642, -6.34304296, -1.0143979057591568, 'gray'],
          [38.91909658, -6.34997843, -18.058625660740024, 'red'],
          [38.91944515, -6.35144163, -5.952704539276976, 'red'],
          [38.92074416, -6.37983324, 9.43779634228945, 'green'],
          [38.92226409, -6.33124609, 16.849722186328986, 'green'],
          [38.9237298, -6.31112321, 12.411419315650951, 'green'],
          [38.92484099, -6.34863421, -9.403063919704168, 'red'],
          [38.92661517, -6.33477369, 9.481582537517053, 'green'],
          [38.92742437, -6.33785753, 11.731276638928346, 'green'],
          [38.927756, -6.33462715, -15.063801208865012, 'red'],
          [38.9297495, -6.33276464, 0.0, 'gray'],
          [38.934875, -6.34691111, 142.83924843423802, 'green'],
          [38.95059637, -6.35925768, 8.161189358372468, 'green'],
          [38.97369625, -6.36643186, 46.76258992805755, 'green']]
```

**Ilustración 27: Lista con las variaciones categorizadas**

Por último, habrá que representar los puntos geográficos en un mapa. Para ello se calcula la media de latitud y longitud de todos los registros presentes para poder pintar el mapa centrado.

Una vez creado el mapa, se introducen los diferentes marcadores con la información de la *lista_aumento_mi_*color.

```
##MIERCOLES 26 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO VS MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
lats = datos210421['LATITUD']
longs = datos210421['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong = statistics.mean(longs)
meanLat = statistics.mean(lats)


# crear un objeto mapa base usando Map()
mapObj = folium.Map(location=[meanLat, meanLong], zoom_start = 14)

for i in range(len(lista_aumento_mi_color)):
    marcador1 = folium.Marker(location=(lista_aumento_mi_color[i][0], lista_aumento_mi_color[i][1]),
                              popup=lista_aumento_mi[i][2], icon=folium.Icon(lista_aumento_mi_color[i][3]))
    marcador1.add_to(mapObj)


mapObj
```

**Ilustración 28: Elaboración del mapa con marcadores de variaciones**

Además, se representarán los mapas de calor correspondientes a los días que intervienen en el proceso, con el objetivo de ver las zonas más transitadas y sus diferencias. Mediante el siguiente fragmento de código se elabora el mapa de calor del miércoles 21 de abril.

```
##MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
lats210421 = datos210421['LATITUD']
longs210421 = datos210421['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong210421 = statistics.mean(longs210421)
meanLat210421 = statistics.mean(lats210421)


# crear un objeto mapa base usando Map()
mapObj210421 = folium.Map(location=[meanLat210421, meanLong210421], zoom_start = 14)


# crear capa de mapa de calor
heatmap210421 = HeatMap( list(zip(lats210421, longs210421)),
                    min_opacity=0.2,
                    radius=50, blur=50,
                    max_zoom=1)
# añadir capa de mapa de calor al mapa base
heatmap210421.add_to(mapObj210421)
mapObj210421
```

**Ilustración 29: Proceso de creación del mapa de calor**

En el siguiente apartado se mostrarán los resultados de este proceso explicando cada caso.

### 3.2.3 Representación de los resultados

En primer lugar, se analiza el impacto entre los dos días intersemanales. A continuación, se muestra en un mapa la localización de las distintas estaciones base y como han variado los registros en mayo tras el fin del Estado de Alarma, siendo el verde un aumento del 3% y el rojo una disminución del mismo valor, quedando el gris para valores intermedios.

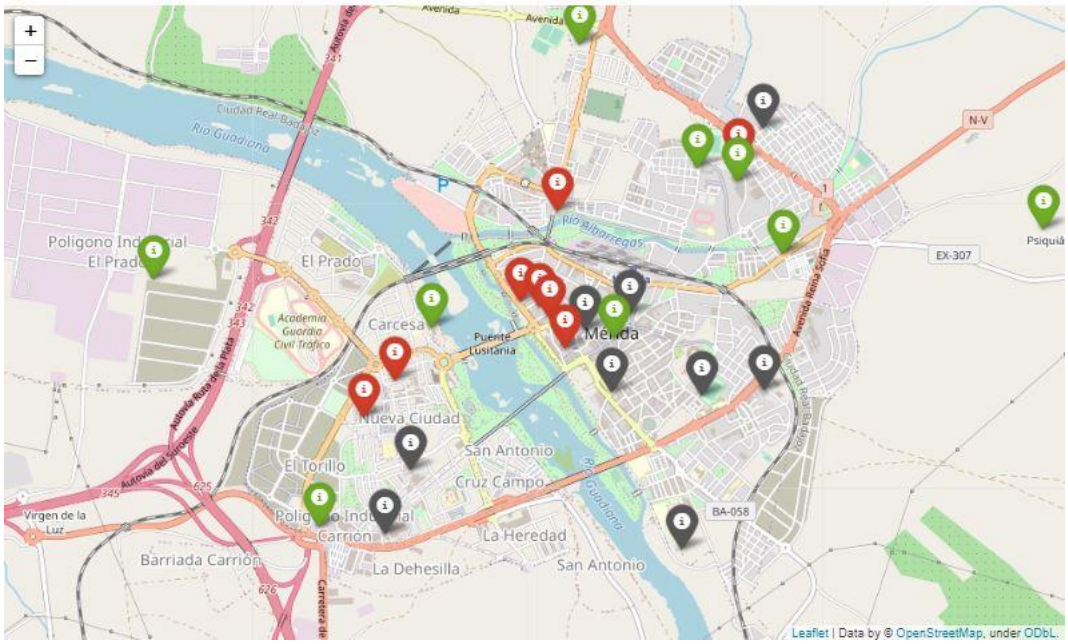


**Ilustración 30: miércoles 26 de mayo Vs miércoles 21 de abril**

Aunque sí que es verdad que en abril existían restricciones que afectaban a la movilidad, restringiendo la entrada o salida de personas de la comunidad autónoma, al tratarse de un día laboral no se aprecian diferencias significativas. Esto es debido principalmente, tal y como se ha comentado anteriormente, a que el centro de la ciudad de Mérida presenta una gran concurrencia al ser la capital de la comunidad autónoma de Extremadura y albergar los principales organismos de esta.

Pese a ello se puede ver que en las afueras de la ciudad sí que se detecta un cierto aumento de los registros y por lo tanto de la movilidad. Tiene sentido, ya que con el fin de las restricciones se suprimieron medidas como las limitaciones de aforo, cierre de espacios interiores o actividades deportivas. Además, los puntos donde aumenta la movilidad en la periferia coinciden con centros comerciales y espacios deportivos lo que refuerza este planteamiento.

Sin embargo, no se ha reflejado un gran impacto en la movilidad a nivel general. Era de suponer una mayor variación en la afluencia de personas tras la vuelta a la normalidad, tras meses de continuas restricciones.

Todo lo contrario, a lo que se puede apreciar en el siguiente mapa, en el que se muestra la variación entre dos días de fin de semana.

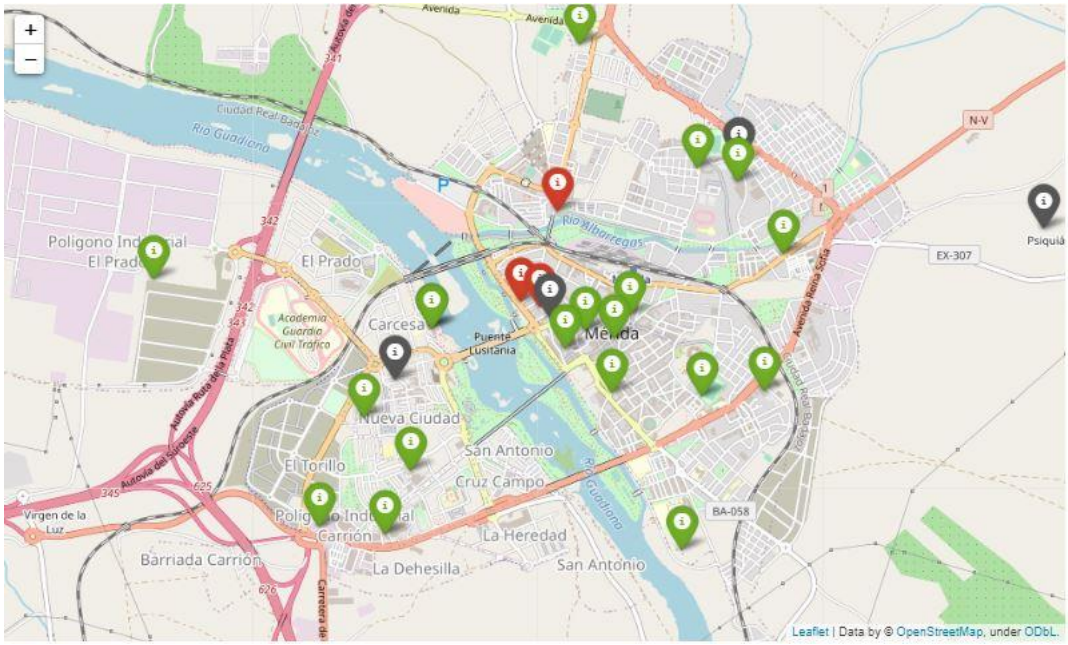


**Ilustración 31: sábado 22 de mayo Vs sábado 24 de abril**

A diferencia del primer mapa, aquí ya se puede observar el gran impacto en la movilidad. Salvo algún caso aislado todos los puntos registran un aumento considerable de los registros, tanto en el centro histórico como en la periferia. Esto se debe principalmente a la reactivación del turismo y a la eliminación de los cierres perimetrales de la comunidad lo que hizo de nuevo posible la movilidad entre comunidades autónomas y las provincias.

El turismo nacional en España aumentó considerablemente debido a la dificultad de salir al extranjero. Mérida cuenta con un gran atractivo turístico gracias al gran patrimonio histórico que posee. En la siguiente gráfica se puede ver la diferencia en la variación de la movilidad que se produce entre el fin de semana y un día intersemanal.

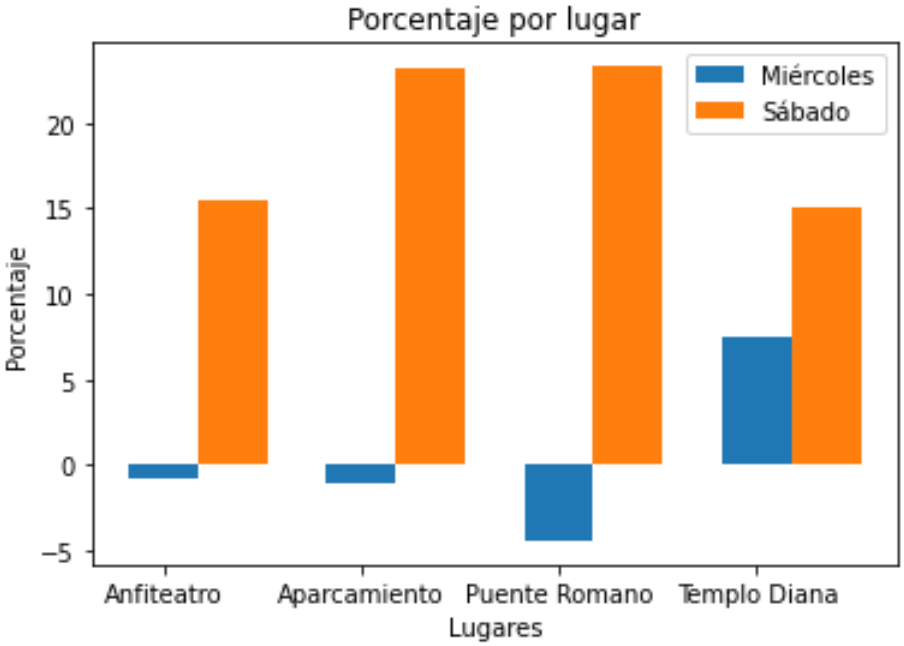


**Ilustración 32: Gráfica variación principales puntos geográficos**

Los principales puntos más importantes del centro histórico de la ciudad presentan un aumento entre el 15% y el 25% en la movilidad durante el fin de semana mientras que apenas se nota la diferencia en un día entre semana. El código con el que se ha elaborado la gráfica se encuentra en el anexo 1.

Lo comentado anteriormente se puede ver en los siguientes mapas de calor.

**Ilustración 33: Mapa de calor miércoles 21 de abril**

**Ilustración 34: Mapa de calor miércoles 26 de mayo**

En las dos imágenes anteriores apenas se aprecia una diferencia significativa, salvo algún incremento ligero en la periferia.

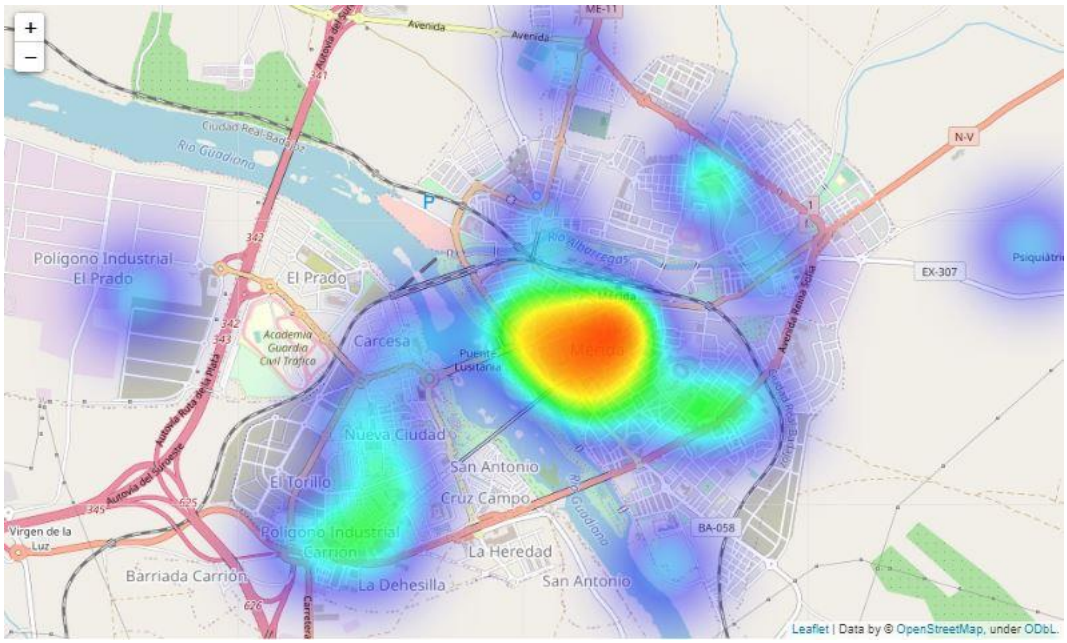

**Ilustración 35: Mapa de calor sábado 24 de abril**

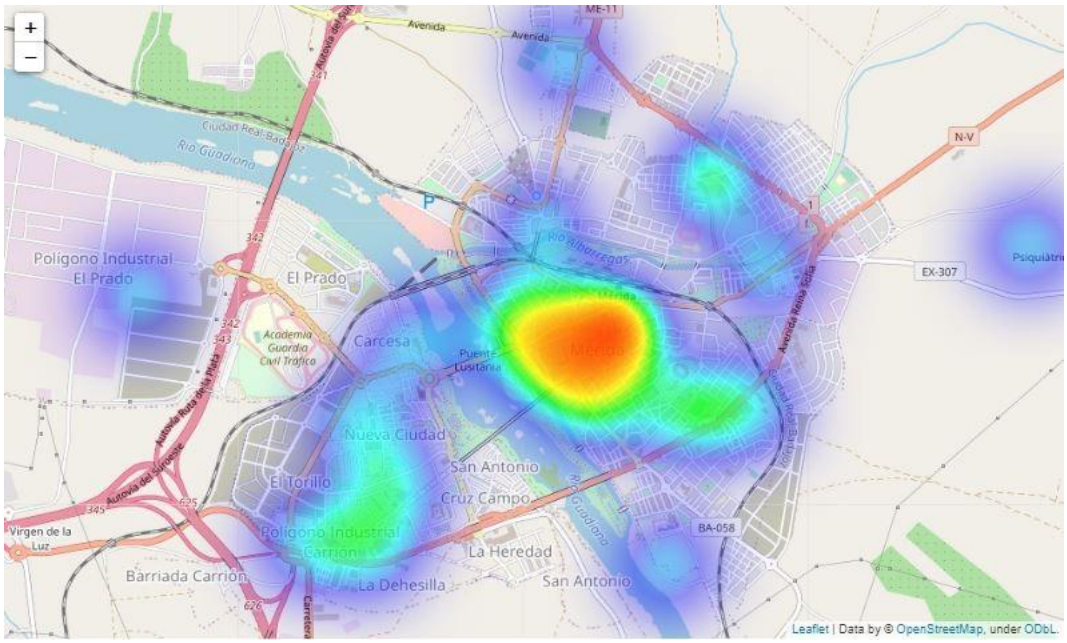

**Ilustración 36: Mapa de calor sábado 22 de mayo**

Por el contrario, en estos dos mapas de calor se puede contemplar diversas variaciones en la intensidad de color de las principales zonas de la ciudad. Tal y como se ha analizado anteriormente, las más afectadas son el centro de la ciudad y parte de las afueras.

### 3.3 Algoritmos predictivos de Machine Learning

Como una de las variables principales de los datos es el tiempo, se decide tratar y analizar su conjunto como una serie temporal.

Una serie temporal se compone de observaciones cuantitativas sobre atributos medibles tomadas en diversos puntos a lo largo del tiempo. Las series temporales suelen tener tres puntos comunes: el tiempo es el eje primario, los datos aparecen por orden temporal y se establece la misma distancia entre los diferentes puntos de la serie.

Los datos en una serie temporal pueden representarse en forma de tabla, siendo la primera columna el tiempo y la segunda el valor de la variable en ese instante. Dependiendo del número de variables observadas se denominan series univariables o multivariantes.

Una serie temporal se puede definir como el conjunto de cuatro factores:

- Tendencia general: comportamiento general de los datos que indican si una serie es estable, se incrementa o disminuye al cabo de un largo periodo de tiempo.

- Variaciones estacionales: son patrones que se observan de manera regular y periódica. Estas variaciones se deben normalmente al comportamiento del ser humano (periodos vacacionales, consumo, movimiento, calendario laboral) o a las condiciones climáticas (diferentes estaciones a lo largo del año o la temperatura ambiental). Normalmente tienen una periodicidad anual.

- Variaciones cíclicas: se trata de oscilaciones con una periodicidad superior a la anual. Se divide en ciclos y no tienen un periodo de tiempo establecido.

- Variaciones aleatorias: se trata de alteraciones que no se pueden controlar.

Una serie temporal es estacionaria si sus propiedades como la media, varianza o correlación no cambian a lo largo del tiempo. Esta propiedad es importante ya que la mayoría de las técnicas de análisis estadísticas sobre series parten del hecho de que el conjunto de datos sobre el que se trabaja es estacionario. La estacionariedad en las series temporales facilita su predicción al mantener constantes sus propiedades.

A partir de una serie temporal se podrán realizar los siguientes estudios:

- Un análisis de la serie temporal con el objetivo de ver el comportamiento de las diferentes medidas a lo largo del tiempo.

- La predicción de series temporales (*time series forecasting*). Se intenta predecir el comportamiento de observaciones futuras a partir de datos

históricos de la serie temporal. Para ello se crean modelos que identifiquen patrones y así pronosticar los datos futuros.

Primero llevaremos a cabo una transformación de los datos para que puedan ser tratados como una serie temporal.

### 3.3.1 Preparación y análisis de las series temporales

El objetivo de este apartado es preparar la información para que pueda ser leída adecuadamente. Cabe recordar que el conjunto de datos presenta cientos de registros por segundo y para establecer una serie temporal es necesario tener un único valor por cada punto temporal.

Si bien durante el análisis de datos se utilizaban días concretos durante un tramo horario específico, esta parte recogerá semanas completas abarcando todas las horas del día. Es decir, cada punto en la serie temporal se corresponderá con una hora diferente de la semana.

Las semanas analizadas corresponden a los días del 19 al 23 de abril de 2021 y del 24 al 28 de mayo.

Se elaboran dos series temporales distintas. La primera basada en el conteo de registros totales de las conexiones móviles de la ciudad por hora a lo largo de la semana. Con ello se pretender ver las horas con mayor o menor actividad del día.

La segunda contiene la media de las posiciones geográficas de las estaciones base donde se recogen las diferentes conexiones por hora a lo largo de la semana. Esto aportará un valor aproximado de las zonas de la ciudad que presentan mayor ocupación a lo largo del día y su variación.

A continuación, se describe el proceso de creación de las series temporales paso a paso.

```
#CARGAMOS LOS DATOS DE TRAFICO MOVIL POR HORA
#SEMANA 19-23 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file = '../DATOS\datos19230421.csv'
datos_dias = pd.read_csv(file, sep=';', engine='python')
datos_dias['IT_REGISTRO'] = datos_dias['IT_REGISTRO'].str[:11]
datos_dias['IT_REGISTRO'] = pd.to_datetime(datos_dias['IT_REGISTRO'])
```

```
grouped_data = datos_dias.groupby(["IT_REGISTRO"])
lista = grouped_data.count().reset_index().values
dataframe_lista = pd.DataFrame(lista)
dataframe_lista = dataframe_lista.drop(dataframe_lista.columns[2], axis='columns')
dataframe_lista.columns = ['ts', 'registros']
dataframe_lista = dataframe_lista.set_index('ts')
dataframe_lista["registros"] = pd.to_numeric(dataframe_lista["registros"])
```

**Ilustración 37: Elaboración de la serie temporal con el número de registros totales.**

Primero se carga en un *dataframe* los datos correspondientes a la semana de abril. Se decide acortar el campo con la fecha de la conexión para que solo incluya día y hora, excluyendo los minutos y segundos. Después, se realiza un conteo de los registros totales agrupados por cada hora de la semana. Lo que interesa es quedarse con este número, por lo que se eliminan el resto de los campos estableciéndose como índice la hora de la conexión.

El resultado es el siguiente:

```
dataframe_lista
```

| ts | registros |
|---|---|
| 2021-04-19 00:00:00 | 20424 |
| 2021-04-19 01:00:00 | 14074 |
| 2021-04-19 02:00:00 | 11491 |
| 2021-04-19 03:00:00 | 10293 |
| 2021-04-19 04:00:00 | 10136 |

**Ilustración 38: Registros totales por hora**

Para cada hora de la semana existe un valor único con el total de conexiones que se producen en ese periodo de tiempo. A partir de estos datos se puede representar la serie temporal correctamente.

<u>Series temporales de la semana del 19-23 de abril</u>

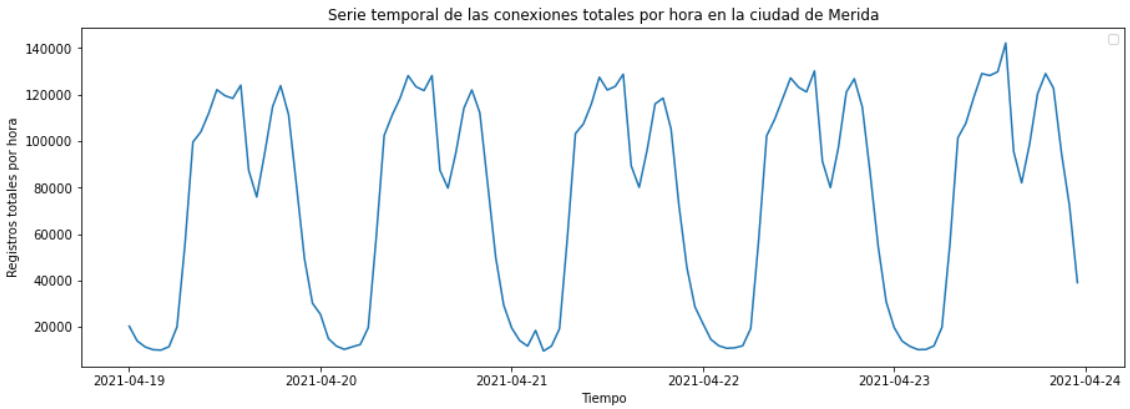


**Ilustración 39: Serie temporal de las conexiones totales por hora en una semana (19-23 abril)**

Se puede observar un patrón claro que se repite diariamente durante todos los días de la semana. El número de registros no es exactamente el mismo pero la forma sí. De primeras podría decirse que no presenta tendencia, pero si tiene estacionalidad diaria.

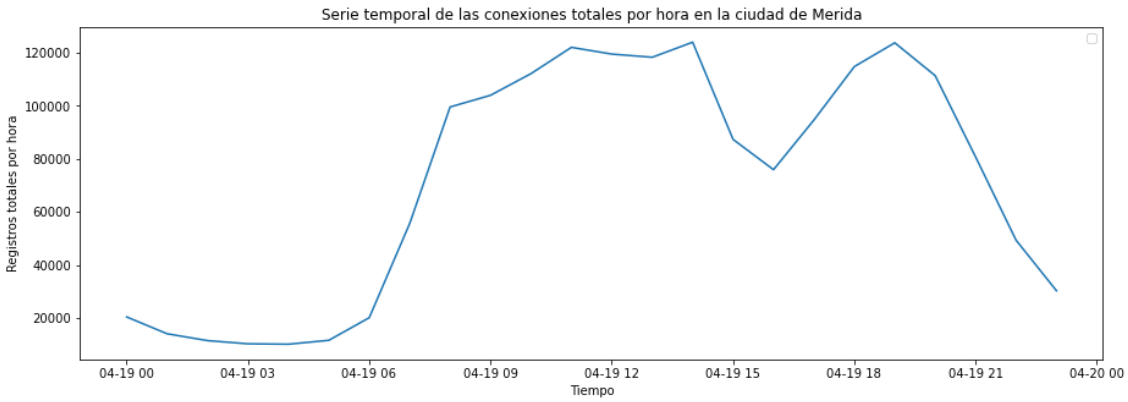


**Ilustración 40: Serie temporal de las conexiones totales por hora en un día (19 de abril)**

Analizando la secuencia temporal de un día completo se puede apreciar como el mayor número de conexiones se producen durante las horas centrales del día (7:00-14:00) y la tarde (17:00-20:00). Tiene lógica al tratarse del tramo horario con mayor actividad laboral.

La segunda secuencia temporal se realiza de la siguiente manera:

```
#CARGAMOS LOS DATOS DE TRAFICO MOVIL
#MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file = '../DATOS\datos19230421.csv'
datosdias = pd.read_csv(file, sep=';', engine='python')
datosdias['IT_REGISTRO'] = datosdias['IT_REGISTRO'].str[:11]
datosdias['IT_REGISTRO'] = pd.to_datetime(datosdias['IT_REGISTRO'])

timestamps = list(datosdias.drop_duplicates('IT_REGISTRO').sort_values('IT_REGISTRO')['IT_REGISTRO'])
```

```
lat_mean = []
long_mean = []

for ts in tqdm(timestamps):

    data_ts = datosdias[datosdias["IT_REGISTRO"] == ts]
    lat_mean.append(np.mean(data_ts["LATITUD"]))
    long_mean.append(np.mean(data_ts["LONGITUD"]))

new_dataframe = pd.DataFrame(data={"lat":lat_mean, "long":long_mean, "ts":timestamps})
new_dataframe = new_dataframe.set_index('ts')
new_dataframe_lat =  new_dataframe["lat"]
new_dataframe_long =  new_dataframe["long"]
```

**Ilustración 41: : Elaboración de la serie temporal con la media de las coordenadas por hora.**

De forma similar al primer caso, se carga en un *dataframe* todos los registros de las conexiones que abarcan del 19 al 24 de abril recortando el campo *IT_REGISTRO* para que se informe solamente de la hora. Luego, se extraen los diferentes valores únicos de las horas de la semana para así poder hacer la media de la latitud y longitud correctamente recorriendo todos los puntos de la serie temporal posibles. Una vez finalizado se establece el campo de fecha y hora como índice y se separa en latitud y longitud para poder analizarlas independientemente.

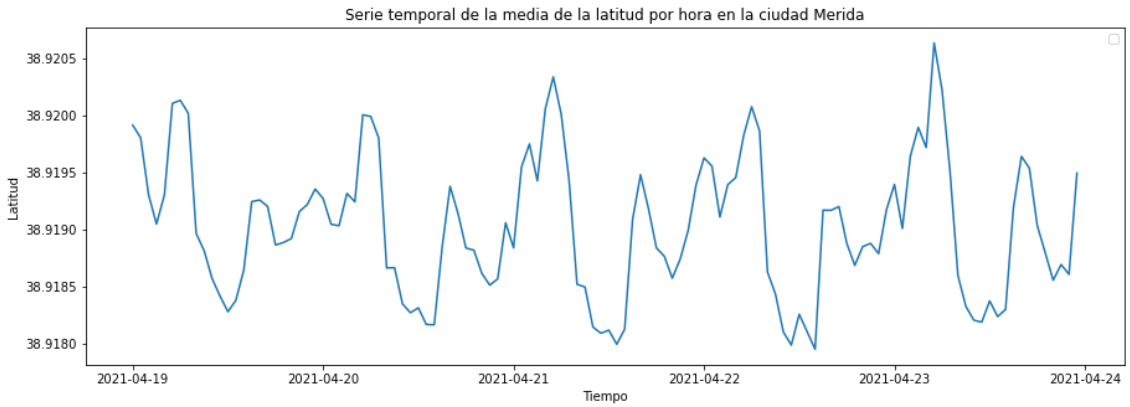


**Ilustración 42: Serie temporal de la media de latitud por hora en una semana (19-23 abril)**

Al igual que en el primer caso se observa un patrón que se repite diariamente. No presenta tampoco una tendencia clara.

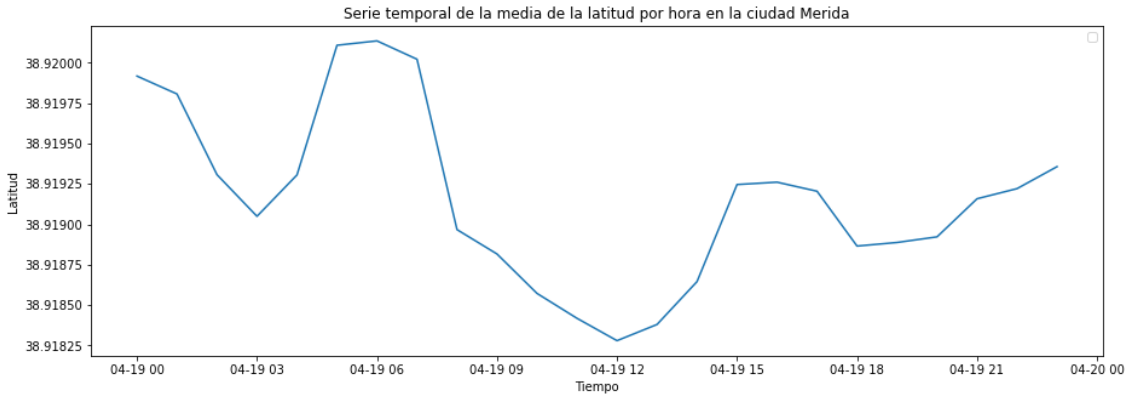


**Ilustración 43: Serie temporal de la media de latitud por hora en un día (19 de abril)**

En la imagen anterior se aprecia que la latitud a primeras horas de la mañana permanece en torno a los 38.92000 grados y a lo largo de la mañana se reduce notablemente hasta llegar a tocar los 38.91825 al mediodía. Durante la tarde sube ligeramente y se estabiliza alrededor de 38.91875.

En cuanto a la longitud se tiene el siguiente gráfico:

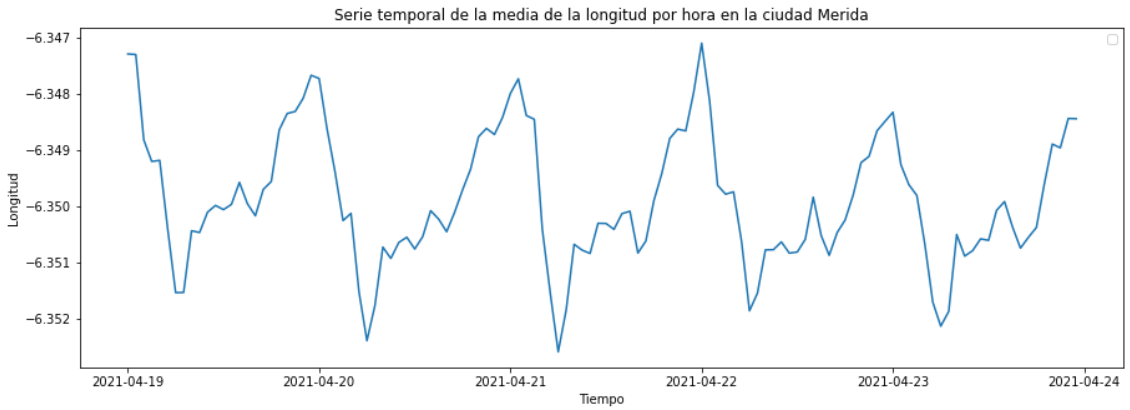


**Ilustración 44: Serie temporal de la media de longitud por hora en una semana (19- 23 abril)**

En la gráfica anterior se contempla también un patrón diario sin una tendencia clara. No obstante, el mínimo y el máximo tienden a reducirse ligeramente. Si se amplía a escala diaria, se puede ver con mayor precisión.

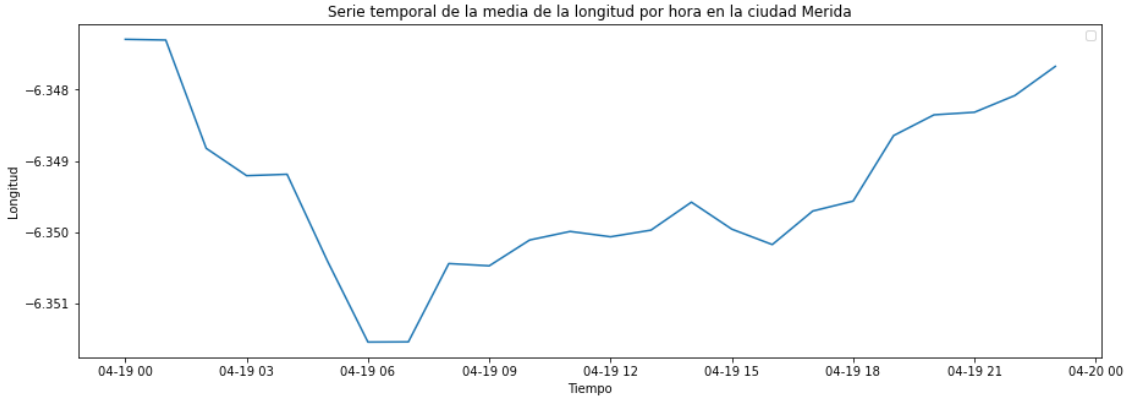


**Ilustración 45: Serie temporal de la media de longitud por hora en un día (19 abril)**

A primeras horas de la mañana la longitud se sitúa en el mínimo de -6.351 y a lo largo del día va subiendo ligeramente centrándose en el tramo 8:00-16:00 en los -6.350 para acabar subiendo durante la tarde y la noche a los valores máximos de -6.348.

<u>Series temporales de la semana del 24-28 de mayo</u>

A continuación, se realiza el mismo procedimiento anterior, pero con una semana diferente de mayo. Solo se mostrarán a modo de resumen las gráficas semanales comparándolas con la semana del 19-23 de abril.

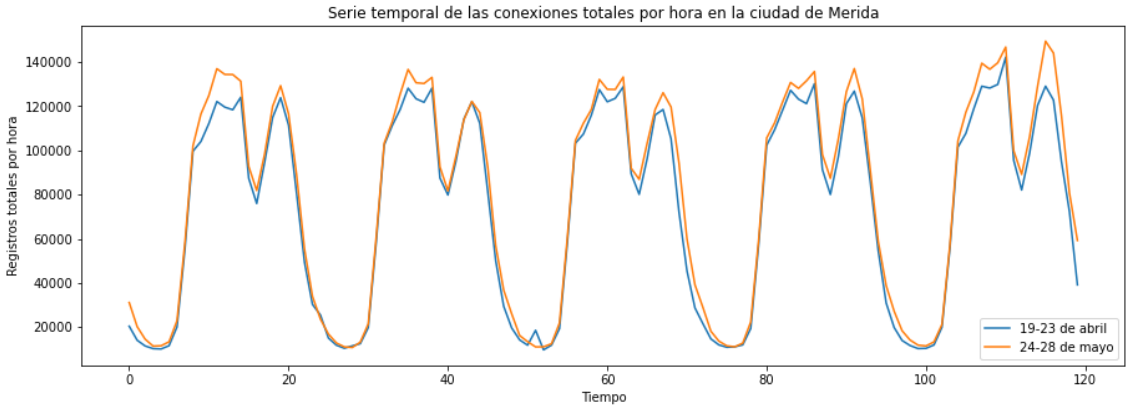


**Ilustración 46: Serie temporal de las conexiones totales por hora en una semana (24-28 mayo)**

La ilustración anterior presenta una comparación de las semanas de abril y mayo. Se puede apreciar que el máximo de conexiones en las horas centrales del día en mayo es ligeramente superior, al moverse en valores en torno a los 140.000 registros por hora. Esto tiene sentido, ya que como se ha comentado en capítulos anteriores en mayo se eliminó el Estado de Alarma y diversas restricciones lo que afectó directamente a la movilidad.

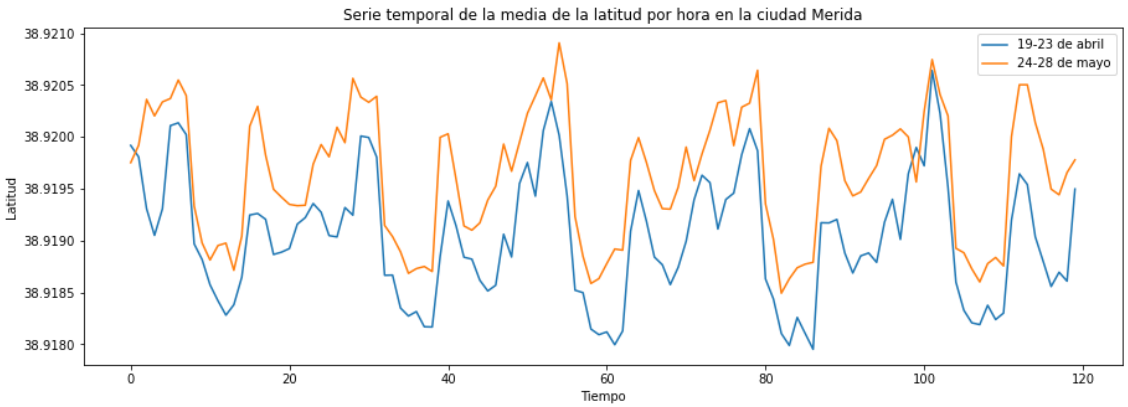


**Ilustración 47: Serie temporal de la media de latitud por hora en una semana (24-28 mayo)**

En cuanto a la localización media principal de las estaciones base, desde donde se registran las conexiones móviles, se observan unos valores de latitud en mayo algo diferentes a los de la semana de abril. Los valores máximos y mínimos son superiores acercándose a 38.9205 y 38.9190 respectivamente. Si se amplía para ver el comportamiento diario:

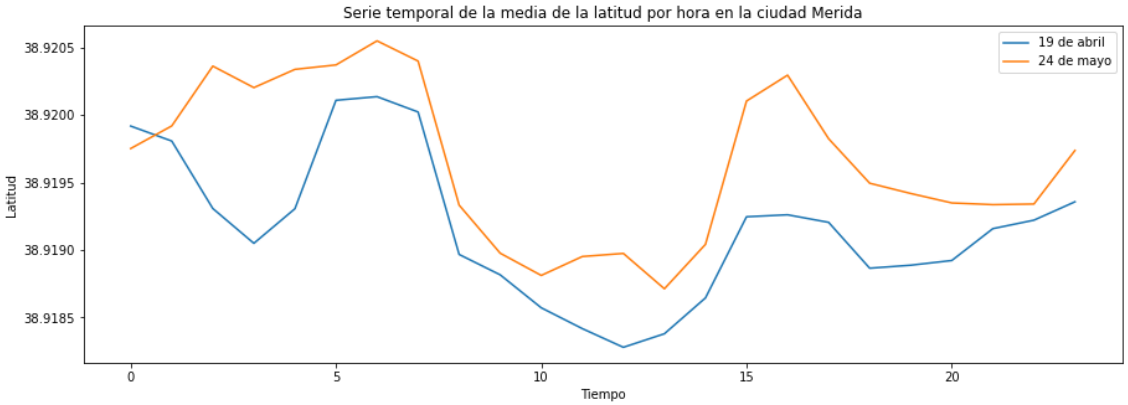


**Ilustración 48: Serie temporal de la media de latitud por hora en un día (24 de mayo)**

Aquí se ve claramente como los valores de la latitud, aunque presentan una distribución parecida aumentan ligeramente en todos los tramos del día. Estas diferencias también podrían deberse al levantamiento de restricciones de movilidad.

Por último, se realiza el mismo procedimiento, pero con la longitud.

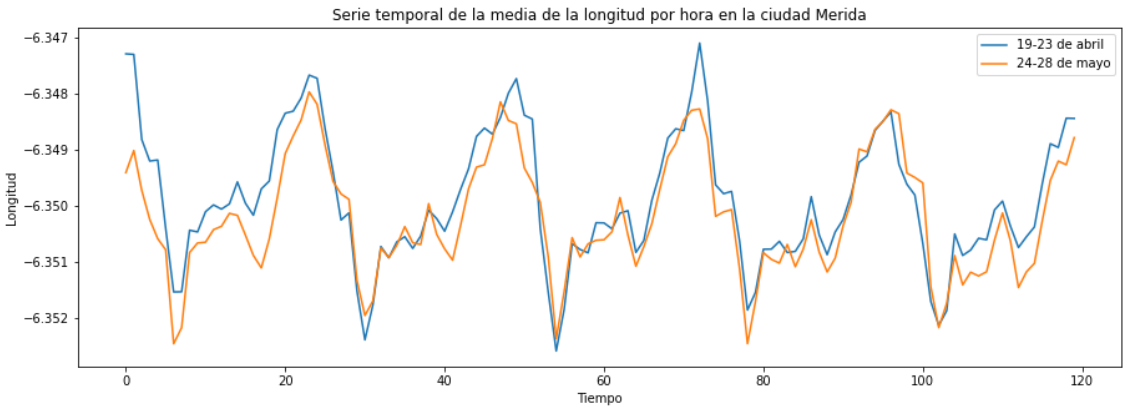


**Ilustración 49: Serie temporal de la media de longitud por hora en una semana (24-28 mayo)**

En la imagen anterior no se aprecian diferencias significativas en términos de longitud.

Tras un primer análisis de las diferentes series temporales, se ve fácilmente que presentan un comportamiento similar diario. Esto será muy importante de cara a la predicción de valores futuros, ya que habrá que tener en cuenta que se trabaja con series con estacionalidad.

En el siguiente apartado se explica detalladamente el modelo de predicción que se aplicará cuando se tienen este tipo de series temporales y los diferentes parámetros que habrá que tener en cuenta.

### 3.3.1 Modelo SARIMA

Se trata de un modelo ARIMA con estacionalidad. SARIMA significa modelo de promedio móvil integrado autorregresivo y se especifica mediante tres parámetros de orden: (p, d, q) (P, D, Q, s) [45]. Las siglas significan lo siguiente:

- **AR(p) Auto regresión:** modelo de regresión basado en la relación que se da entre una observación actual y las observaciones durante un periodo anterior. El valor p indica cuantos valores históricos se utilizan para devolver el valor predicho.

  Hay que calcular una p inicial para entrenar el modelo. Para ello se utiliza la función de autocorrelación parcial (PACF). Establece la relación que se da entre una observación en una serie temporal con observaciones en pasos de tiempos anteriores con las relaciones de las observaciones intermedias eliminadas. Es decir, la autocorrelación parcial en el rezago k es la correlación que resulta después de eliminar el efecto de cualquier correlación debido a los términos en rezagos más cortos.

  En el siguiente capítulo, se explica que valores de p es necesario coger en función de las gráficas PACF.

- **I(d) Integración:** se aplica la diferencia entre observaciones para hacer estacionaria la serie. Es decir, la integración se produce porque primero diferenciamos la serie de tiempo d veces para que la serie sea estable.

- **Media móvil MA(q):** El modelo de promedio móvil es modelar el error de la serie de tiempo y asumir que el error actual depende del error con retraso. El valor de q inicial se calcula utilizando la función de autocorrelación llamada ACF. Esta función calcula la correlación de las observaciones de la serie temporal con valores de la misma serie, pero en tiempos anteriores.

- **S(s):** El modelo SARIMA añade el parámetro s que indica la duración de la estacionalidad. Es decir, si la estacionalidad es semanal y cada punto es un día el valor sería de siete.

### 3.3.2 Elaboración del modelo

Una vez explicado en que consiste el modelo SARIMA y los diferentes parámetros que intervienen en él, llega el momento de elaborar el modelo.

Modelo de predicción de las conexiones totales por hora

De las series temporales mostradas anteriormente, se utilizará para la explicación del modelo la semana del 19 al 23 de abril.

En primer lugar, hay que dividir los datos en dos tipos: datos de entrenamiento y datos de prueba. Normalmente, los datos de entrenamiento constituyen un 70% del total, mientras que el restante 30% serían para los datos de prueba.

```
date_min_test = "22/04/21 12:00:00"
TEST_SIZE = len(dataframe_lista192304[dataframe_lista192304.index>=date_min_test])
# train and test are dataframes that contain the values and a datetime index:
train, test = dataframe_lista192304.iloc[:-TEST_SIZE], dataframe_lista192304.iloc[-TEST_SIZE:]
# x_train and x_test only contain the values:
x_train, x_test = np.array(range(train.shape[0])), np.array(range(train.shape[0], dataframe_lista192304.shape[0]))
train.shape, x_train.shape, test.shape, x_test.shape
```

```
((84, 1), (84,), (36, 1), (36,))
```

**Ilustración 50: Selección de los datos de entrenamiento y de prueba de las conexiones diarias por hora (19-23 de abril)**

De las 120 horas que componen la serie temporal, 84 componen los datos de entrenamiento y 36 componen los datos de prueba.

A continuación, se representa cada conjunto en la misma gráfica.

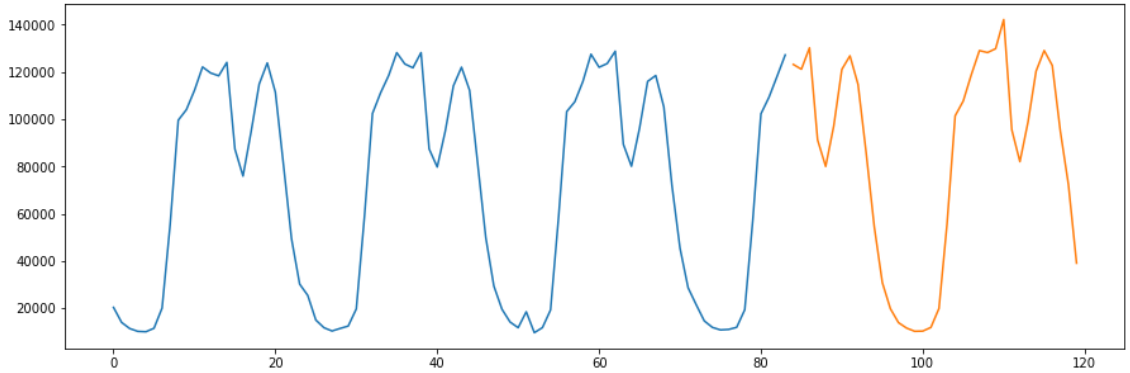


**Ilustración 51: Representación de los datos de entrenamiento y prueba de las conexiones diarias por hora (19-23 de abril).**

En el gráfico anterior, se puede ver que los datos presentan una fuerte estacionalidad, con 24 horas al día como duración de la repetición. Sin embargo, no existe una tendencia clara ni ascendente ni descendente.

Como se ha comentado anteriormente, para trabajar con el modelo SARIMA y seleccionar los parámetros adecuadamente, hay que intentar eliminar la estacionalidad. Con este fin, se lleva a cabo una diferenciación de la serie temporal un total de d veces para conseguir que sea estable. El parámetro d es por lo tanto el componente I.

Mediante el siguiente código se lleva a cabo la diferenciación de la serie temporal. Se realiza una diferencia entre la serie temporal y la serie temporal desplazada 24 registros. Así, el valor de la componente estacional D será de valor uno.

```
dataframe_shifted = dataframe_lista192304 - dataframe_lista192304.shift(24)
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
#ax.plot( dataframe_lista192304.iloc[:24])
ax.plot( dataframe_shifted[24:])
plt.xlabel('Tiempo')
plt.ylabel('Registros totales por hora')
plt.title('Serie temporal de las conexiones totales por hora en la ciudad de Merida')
ax.legend();
```

```
No handles with labels found to put in legend.
```

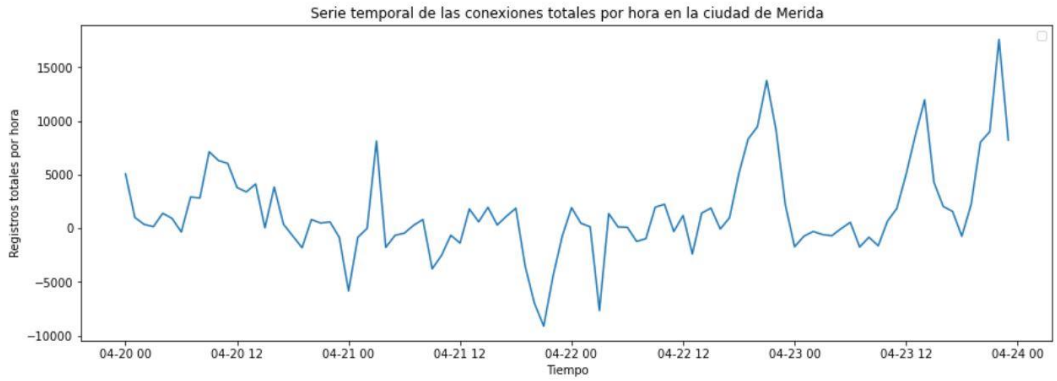


**Ilustración 52: Diferenciación de la serie temporal de las conexiones diarias por hora (19-23 de abril).**

Ahora la serie temporal es estacionaria, ya que no existe una componente estacional aparente. A partir de esta gráfica, se puede empezar a elegir los parámetros necesarios para la ejecución del modelo SARIMA.

Antes que nada, tal y como se ha comentado previamente, el valor de la componente I, será la duración del periodo de estacionalidad. Asique s tendrá un valor de 24.

Para el cálculo de los parámetros de la componente AR, habrá que representar la gráfica de la autocorrelación parcial (PACF) de los dos primeros ciclos estacionales (primeros 48 *lags*).

```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
plot_pacf(dataframe_shifted[24:], ax, lags = 47)
plt.show()
```

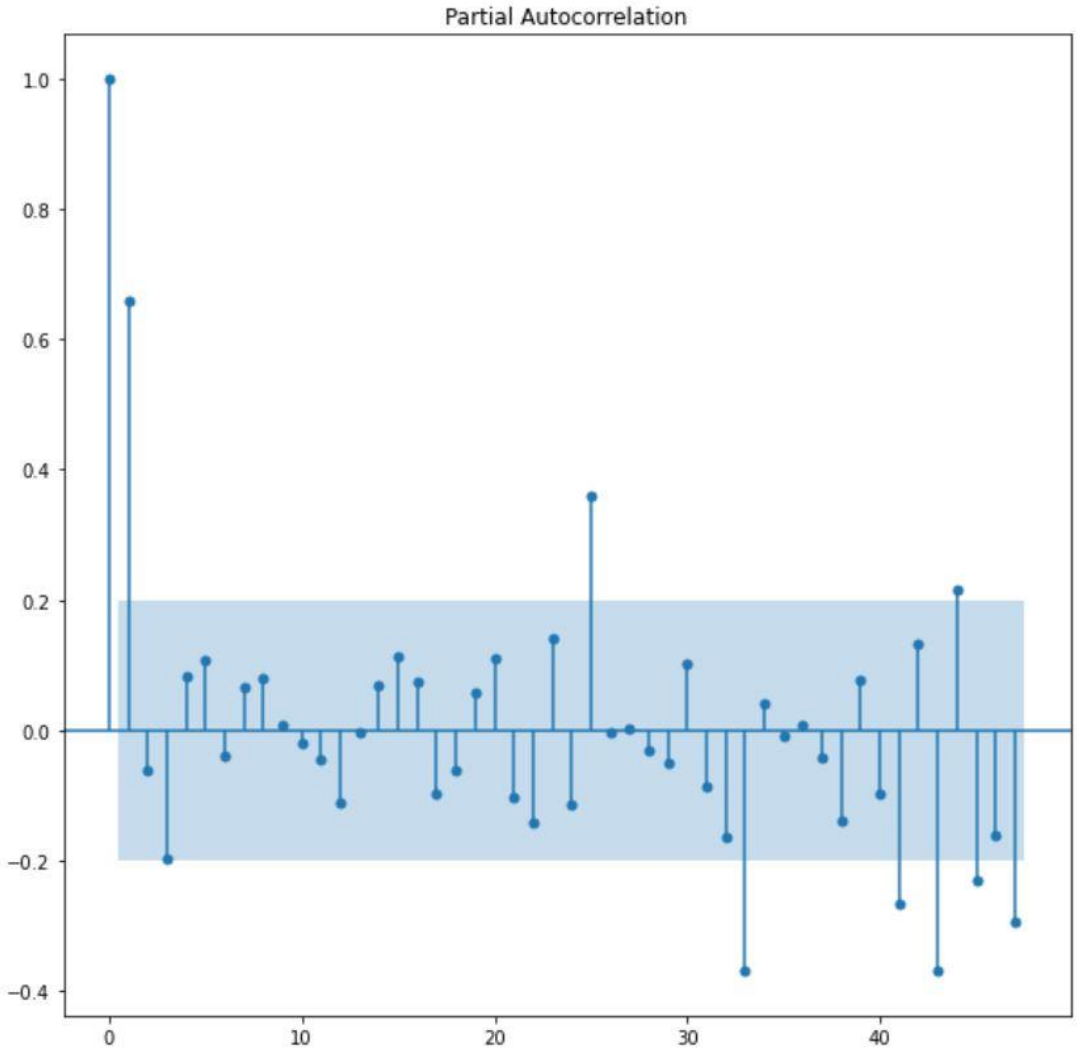


**Ilustración 53: PACF de la serie temporal de las conexiones diarias por hora (19-23 de abril).**

Observando la gráfica anterior, el número de puntos que se salen del contorno azul, constituyen el número de rezagos de las observaciones actuales con relación a las observaciones pasadas. El valor de p será por lo tanto el número

de rezagos significativos que se dan en el primer periodo (primeros 24 *lags*). En consecuencia, su valor será de dos, ya que el primer *lag* no se cuenta ya que la autocorrelación de una observación con ella misma será siempre de uno.

Para el cálculo de la componente estacional, hay que fijarse en los rezagos separados un total de 24 *lags,* que es lo que dura un ciclo (periodo que comprendido del *lag* 24 al 47). El valor de P será de siete.

Por último, hay que calcular los parámetros de la componente MA. De la misma forma que antes, se representa la gráfica de la autocorrelación (ACF).

```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
plot_acf(dataframe_shifted[24:], ax, lags = 47)
plt.show()
```

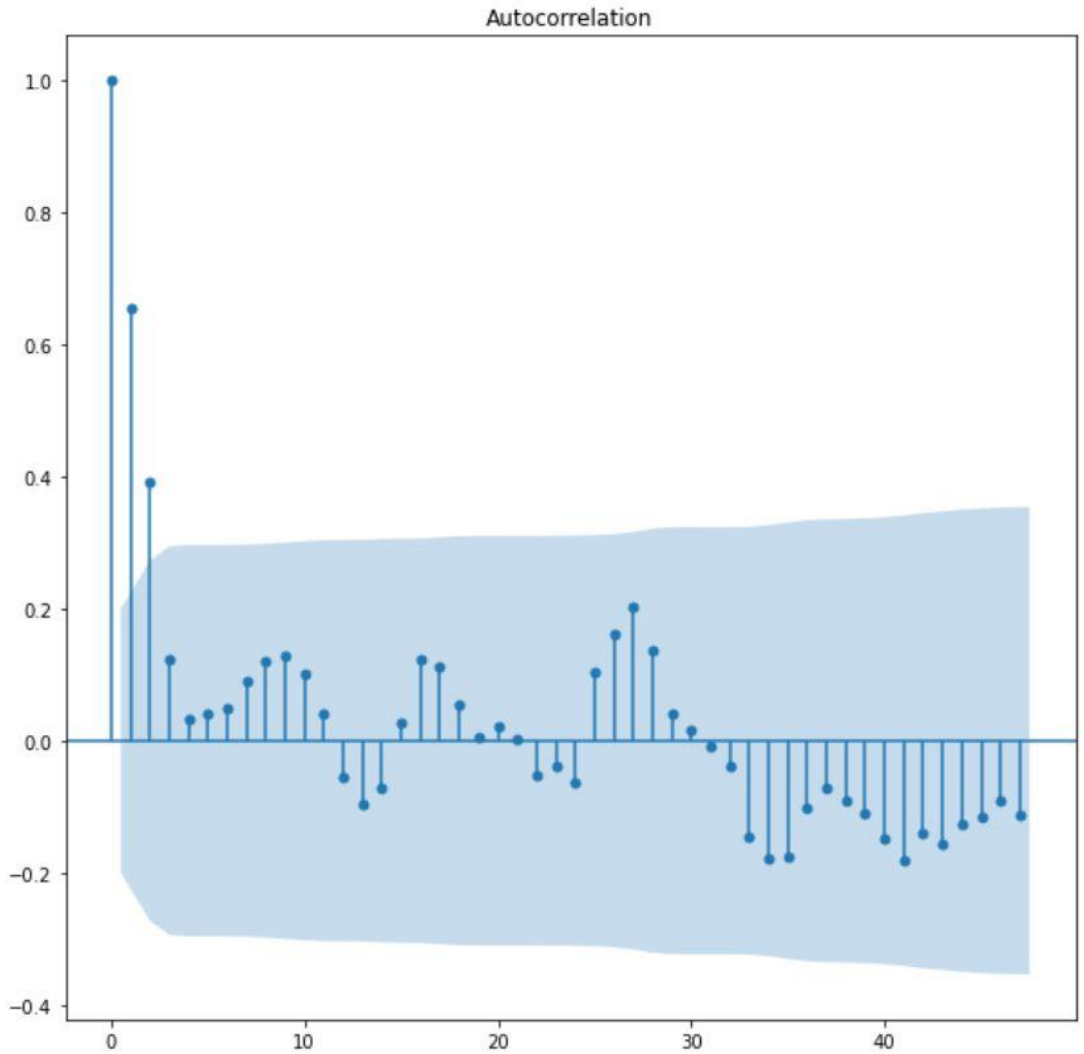


**Ilustración 54: ACF de la serie temporal de las conexiones diarias por hora (19-23 de abril).**

A partir de la gráfica anterior, el número de puntos que no aparecen dentro de los límites establecidos en azul, constituyen los rezagos de las observaciones actuales en función de los errores pasados. Para encontrar el valor de q, hay que contemplar el primer ciclo, lo que nos da un total de dos. Como antes, el

primer valor no se cuenta, ya que la autocorrelación de un valor con si misma siempre será de uno. La componente Q tendrá un valor de cero.

A modo de resumen, los posibles valores posibles de los parámetros que constituyen el modelo SARIMA, son los siguientes:

- P = 7 ó 6.
- p = 1 ó 2.
- d = 0.
- D = 1.
- q = 2 ó 1.
- Q = 0.
- s = 24.

A partir de los valores anteriores, entrenamos el modelo SARIMA empleando la función SARIMAX y los parámetros anteriores.

```
sarima = SARIMAX(train,order=(2,1,2),seasonal_order=(7,1,0,24),).fit();
sarima.summary()
```

**Ilustración 55: Entrenamiento modelo SARIMA**

Una vez finalizado el entrenamiento, es hora de ver los resultados del modelo. Se intentan predecir los datos de prueba para poder compararlos entre sí.

```
pred = sarima.predict('22/04/21 12:00:00','23/04/21 23:00:00')
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train, train, label='Entrenado')
ax.plot(x_test, pred, label='Test estimado')
ax.plot(x_test, test, label='Test real')

plt.xlabel('Tiempo')
plt.ylabel('Registros totales por hora')
plt.title('Predicción de los registros totales por hora en la ciudad de Merida')
ax.legend();
```

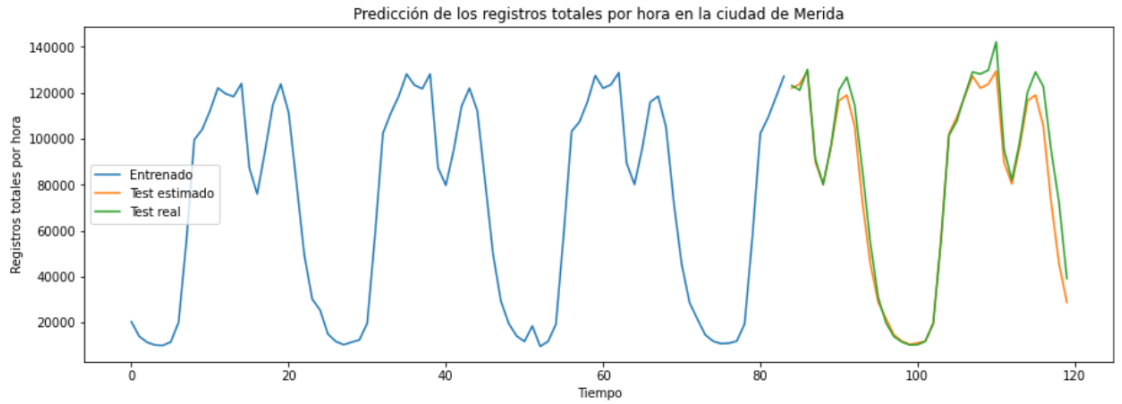


**Ilustración 56: Predicción de los datos de prueba de las conexiones diarias por hora (19-23 de abril).**

Se puede apreciar en la imagen anterior, que la predicción es bastante fiable y se ajusta notablemente a los datos de prueba. Las principales diferencias radican en la presencia de un mayor número de conexiones en comparación

con días anteriores de los datos de prueba. No obstante, se puede dar por válido el modelo.
Este modelo, se empleará en el siguiente capítulo para la obtención de predicciones de diferentes días. A tal efecto se analizarán diferentes semanas.

Modelo de predicción de la media de la latitud por hora

En este apartado se seguirán los mismos pasos que en el modelo anterior, pero analizando la serie temporal de la media de la latitud de las estaciones base desde donde se registran las diferentes conexiones.

Con el objetivo de ser lo más escuetos posibles, solo se aportará la información final del modelo, mostrando el valor de los parámetros elegidos y la predicción de los datos de prueba.

Como se ha podido ver anteriormente, la gráfica de la latitud al igual que el resto presenta una fuerte estacionalidad. Siguiendo los pasos explicados, primero se realiza una diferenciación de la serie para eliminar la estacionalidad y poder calcular los parámetros correctamente.

A partir de las funcione PACF y ACF se obtienen los siguientes parámetros.

- P = 9 ó 8.
- p = 4 ó 3.
- d = 0
- D = 1.
- q = 1.
- Q = 2.
- s = 24

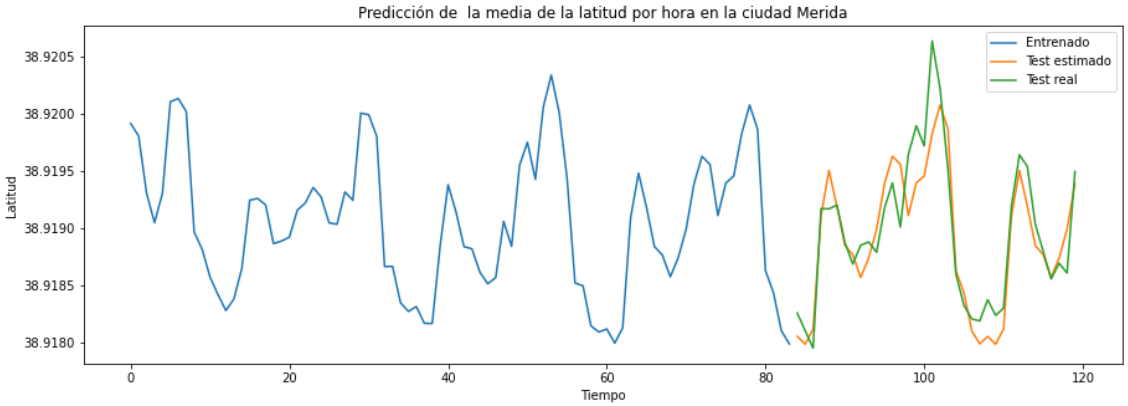


**Ilustración 57: Predicción de los datos de prueba de la media de la longitud por hora (19-23 de abril).**

Hay que fijarse que esta predicción no es lo suficientemente buena. Se han ido probando diferentes valores, pero las irregularidades del último día de la serie temporal hacen que la predicción no se tan fiable. No obstante, sí que se puede observar que mantiene la forma, aunque los máximos de los datos de prueba sean mayores que los del modelo de predicción. Para resolver este problema, en el siguiente apartado se utilizarán datos que comprendan semanas enteras.

Modelo de predicción de la media de la longitud por hora

Por el contrario, en esta sección se estudiará la serie temporal de la media de la longitud por hora de las estaciones base desde donde se registran las diferentes conexiones.

Primero se realiza la diferenciación de la serie al igual que el resto de los casos anteriores. Una vez representadas las funciones de autocorrelación se seleccionan los parámetros correspondientes. En este caso, los parámetros óptimos son los siguientes:

- P = 9 ó 8.
- p = 4 ó 3.
- d = 0
- D = 1.
- q = 1.
- Q = 1.
- s = 24

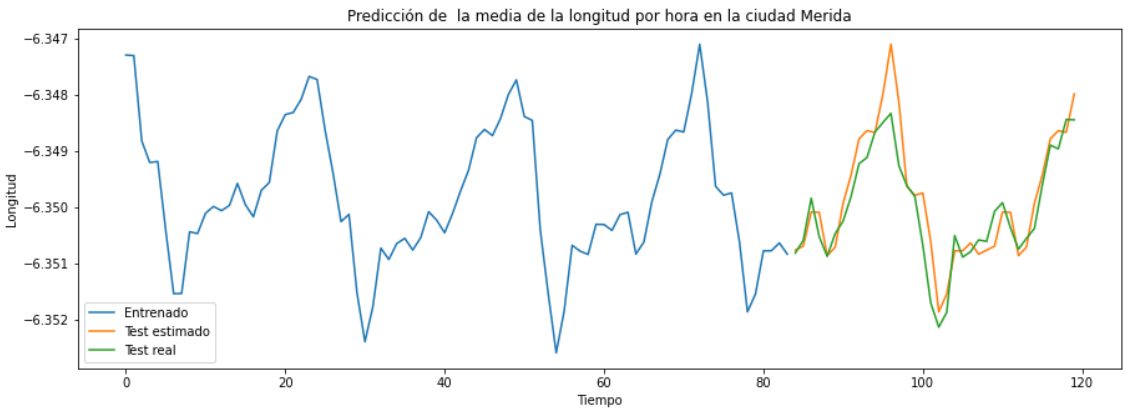


**Ilustración 58: Predicción de los datos de prueba de la media de la longitud por hora (19-23 de abril).**

Ambas predicciones son válidas, por lo que el modelo nos servirá para predecir coordenadas aproximadas de días, lo que nos dará una información relevante sobre las zonas de la ciudad que presentarán mayor concentración de gente y su distribución a lo largo del día.

A continuación, se emplearán un mayor conjunto de datos para asegurar una mayor precisión en las predicciones.

### 3.3.3 Resultados

Para realizar las predicciones de este apartado, se ha decidido utilizar conjuntos de datos mayores que en apartados anteriores. En concreto periodos de quince días. Esto aportará un mayor sentido a las predicciones, pero por contrapartida aumentará los tiempos de ejecución, haciéndolo más costoso.

Siguiendo el modelo descrito en el capítulo anterior, se decide por una parte predecir una semana de siete días de abril y otra de mayo.

<u>Conexiones totales diarias por hora</u>

Los datos de entrenamiento de abril de 2021 abarcan del día 15 al 24, mientras que los datos de prueba del día 24 al 29.

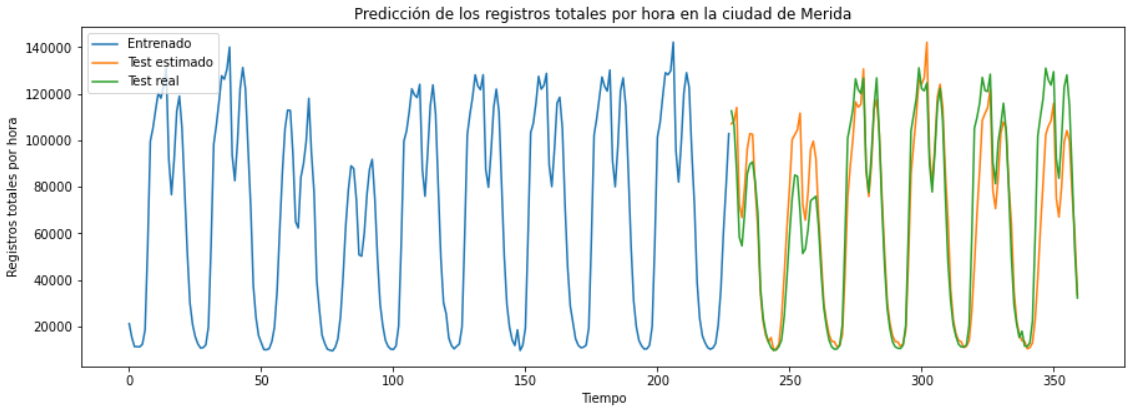


**Ilustración 59: Predicción de las conexiones diarias por hora de abril (datos de prueba)**

Se puede apreciar como la predicción de los datos de prueba capta una disminución de registros en el fin de semana y un aumento en el resto de los días.

Los datos de entrenamiento de mayo de 2021 abarcan del día 13 al 22, mientras que los datos de prueba del 22 al 27.

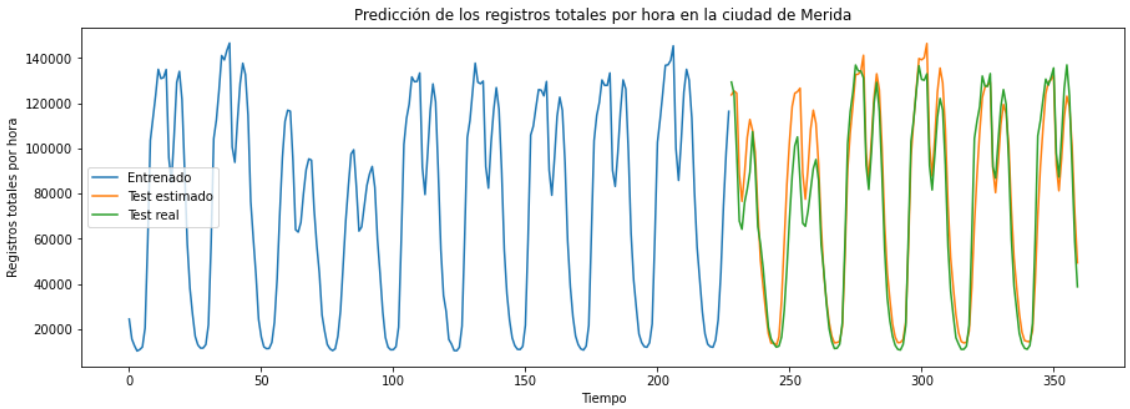


**Ilustración 60: Predicción de las conexiones diarias por hora de mayo (datos de prueba)**

Al igual que en el caso anterior, se observa el decremento de registros durante el sábado y el domingo, y un incremento para los días entre semana.

A partir de estos resultados, se puede demostrar que las predicciones son bastante satisfactorias. Por lo tanto, el siguiente paso será predecir semanas enteras de 7 días.

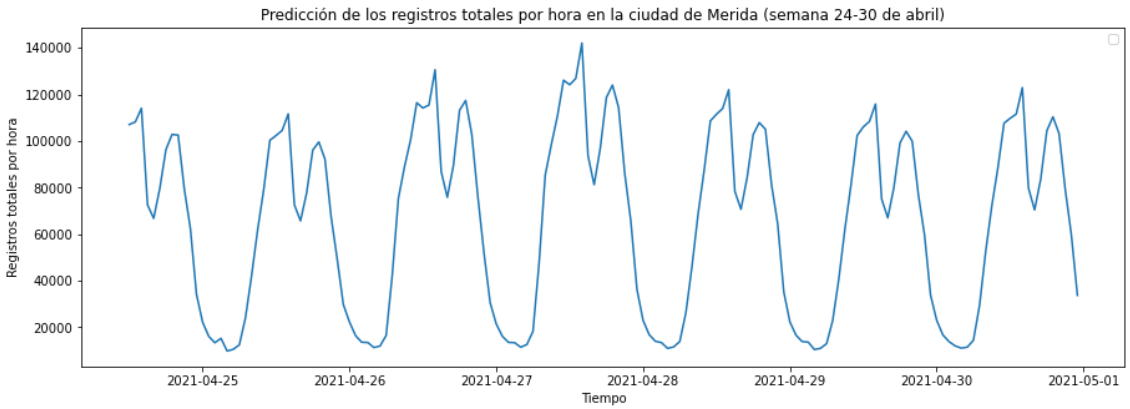


**Ilustración 61: Predicción de las conexiones diarias por hora de abril (semana 24-30 de abril)**

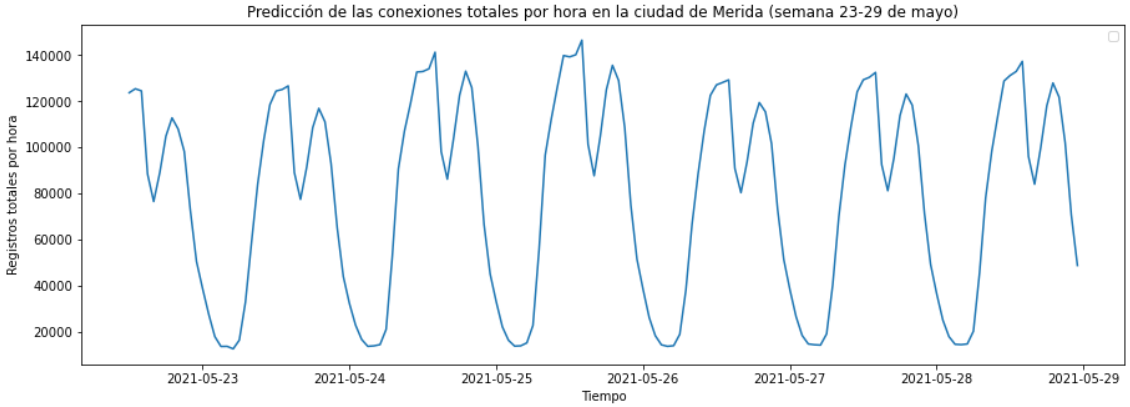


**Ilustración 62: Predicción de las conexiones diarias por hora de mayo (semana 22-28 de mayo)**

A simple vista, es difícil poder apreciar diferencias significativas en las gráficas anteriores. Por eso, se procede a representarlas en la misma gráfica.

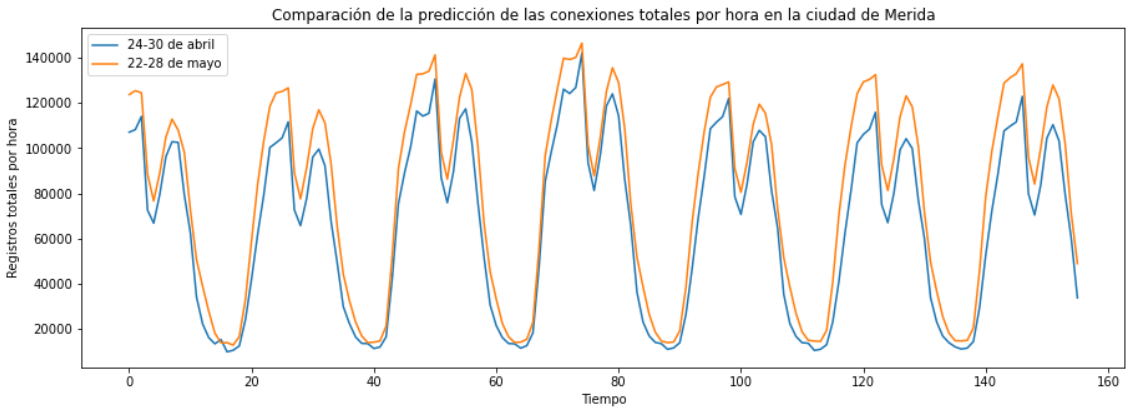


**Ilustración 63: Comparación de las predicciones de conexiones diarias por hora**

Con esta comparativa se ve claramente como el número de registros aumenta en la semana de mayo, independientemente de la hora. Aquí se demuestra una vez más el incremento de la movilidad que supuso la eliminación del Estado de Alarma y el cese de restricciones.

Media de la latitud diaria por hora

Es el momento de realizar los pronósticos relacionados con las coordenadas geográficas. Tras fijar los parámetros más adecuados, la predicción de los datos de prueba del periodo comprendido entre el 15 y el 29 de abril de 2021 es la siguiente.

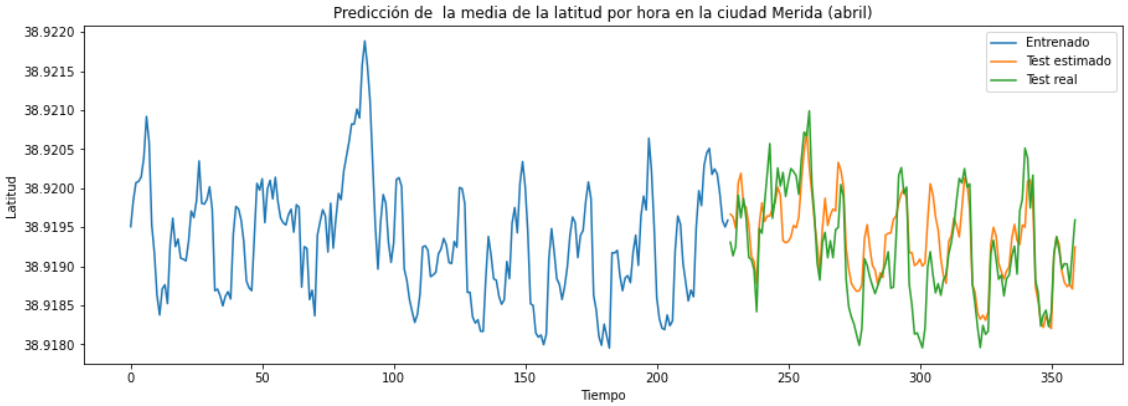


**Ilustración 64: Predicción de la media de la latitud de abril (datos de prueba)**

De la misma forma se procede a realizar el mismo procedimiento para los datos comprendidos entre el 13 y el 27 de mayo de 2021.

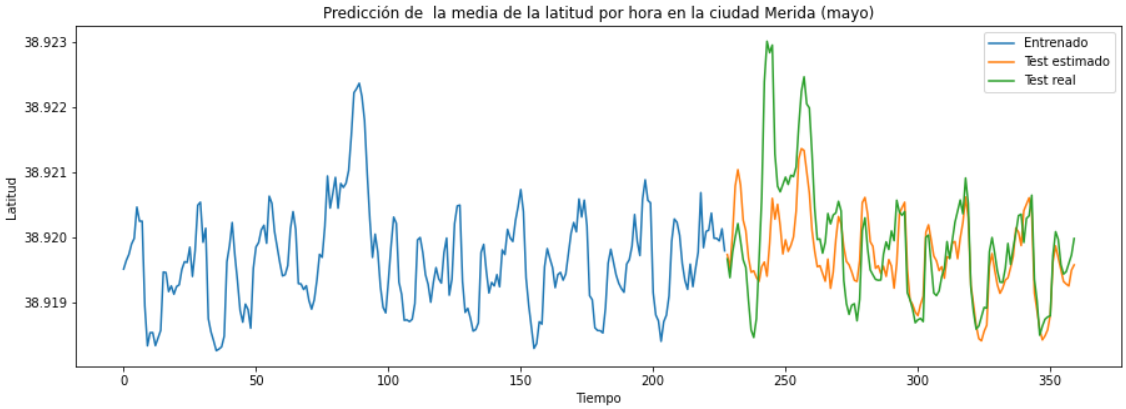


**Ilustración 65: Predicción de la media de la latitud de mayo (datos de prueba)**

Las predicciones se ajustan en un alto porcentaje a los datos de prueba en ambos casos, salvo algunas irregularidades. Esto es normal, ya que es prácticamente imposible acertar plenamente con la predicción debido a la presencia de anomalías.

Dado el modelo por válido, es la hora de aplicar el proceso a semanas enteras de siete días. En las siguientes ilustraciones se pueden ver la previsión para la semana del 24 al 30 de abril y la del 22 al 28 de mayo.

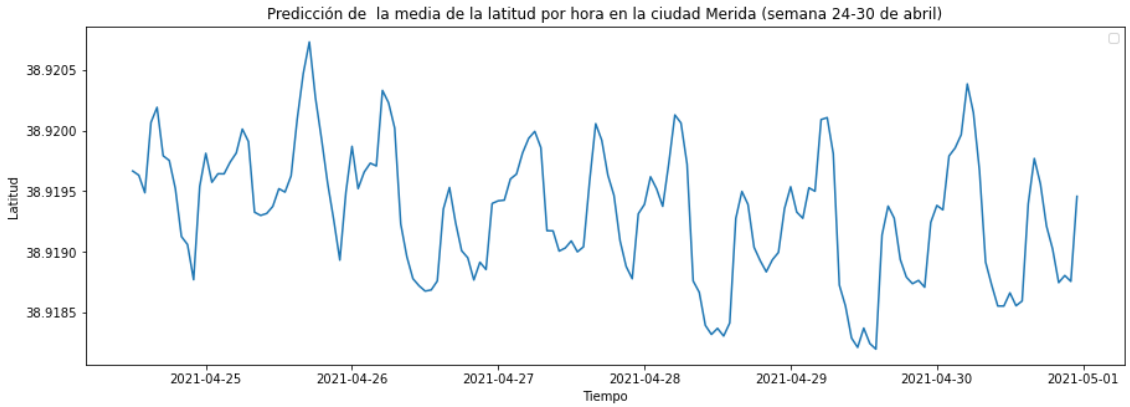


**Ilustración 66: Predicción de la media de la latitud por hora (semana 24-30 de abril)**

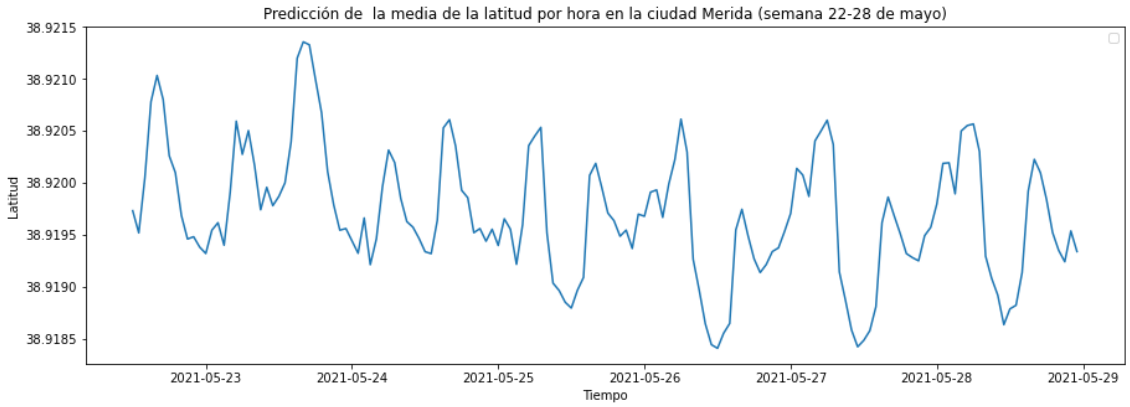


**Ilustración 67: Predicción de la media de la latitud por hora (semana 22-28 de mayo)**

Si juntamos las imágenes anteriores en una misma gráfica, se puede observar que las latitudes de mayo son siempre superiores a las de abril. Exactamente igual que en el apartado de análisis, lo que tiene bastante sentido.

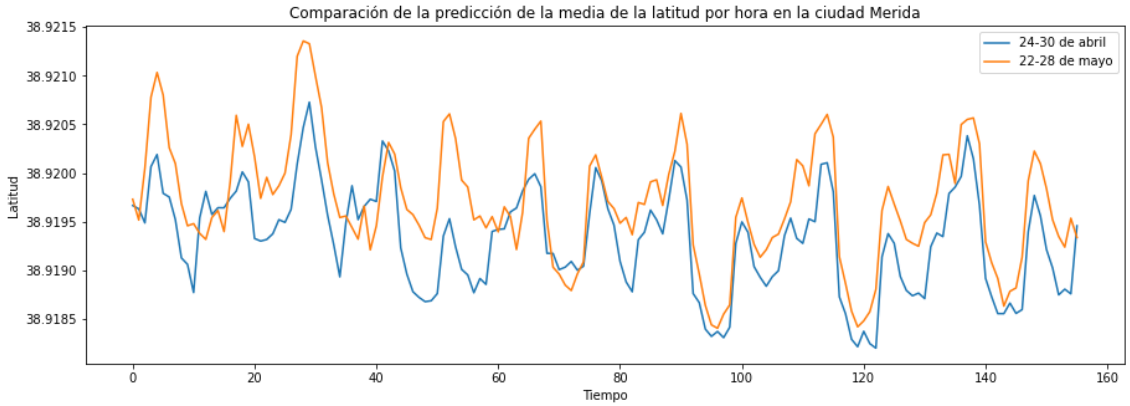


**Ilustración 68: Comparación de las predicciones de latitud por hora**

Media de la longitud diaria por hora

Anteriormente, se han realizado las predicciones necesarias para la latitud. Ahora, es el turno de la longitud.

La predicción de los datos de prueba correspondiente al periodo del 15 al 29 de abril de 2021 se puede apreciar en la siguiente imagen.

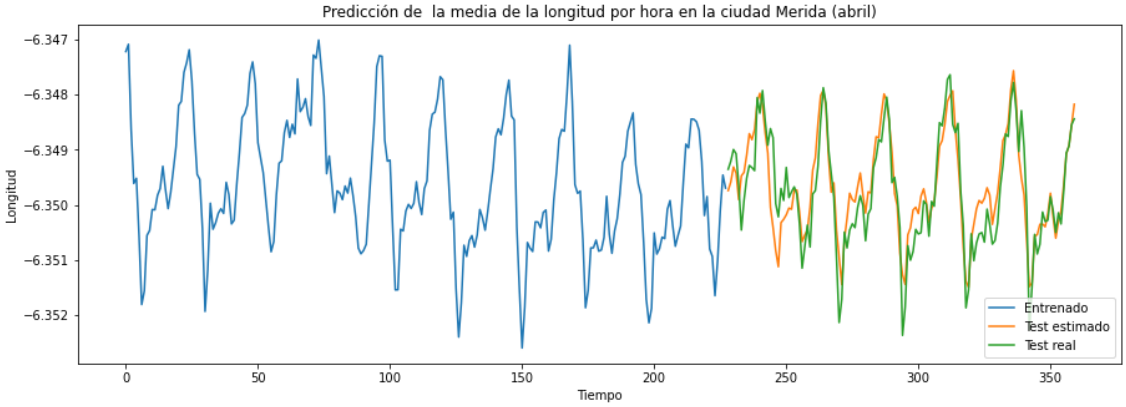


**Ilustración 69: Predicción de la media de longitud de abril (datos de prueba)**

Seguidamente, se realiza el mismo proceso con el periodo del 13 al 27 de mayo de 2021.

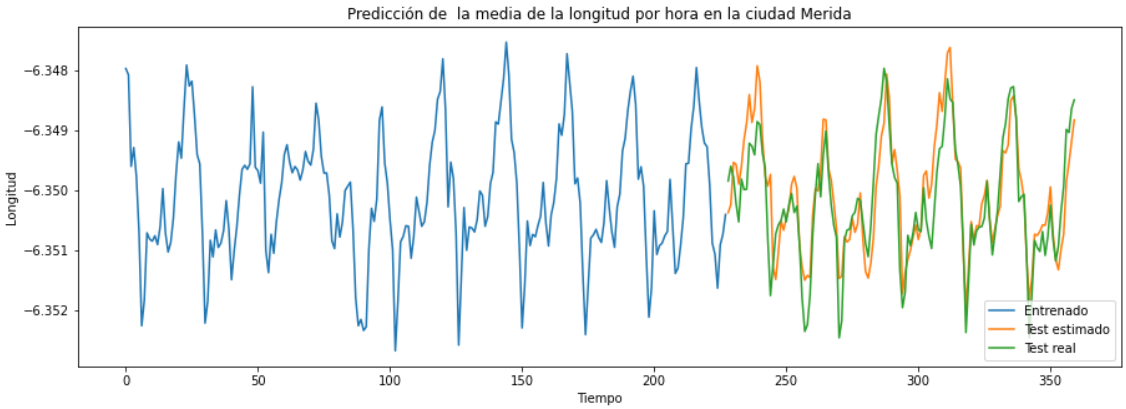


**Ilustración 70: Predicción de la media de longitud de abril (datos de prueba)**

En ambos casos, las predicciones son bastante fiables, por lo que se procede a la predicción de semanas únicas utilizando la misma configuración.

Al igual que se ha realizado en pasos anteriores, primero se mostrarán las predicciones de cada mes y posteriormente una comparativa.

El pronóstico de la longitud media por hora para la semana comprendida entre el 24 y el 30 de abril es la siguiente.

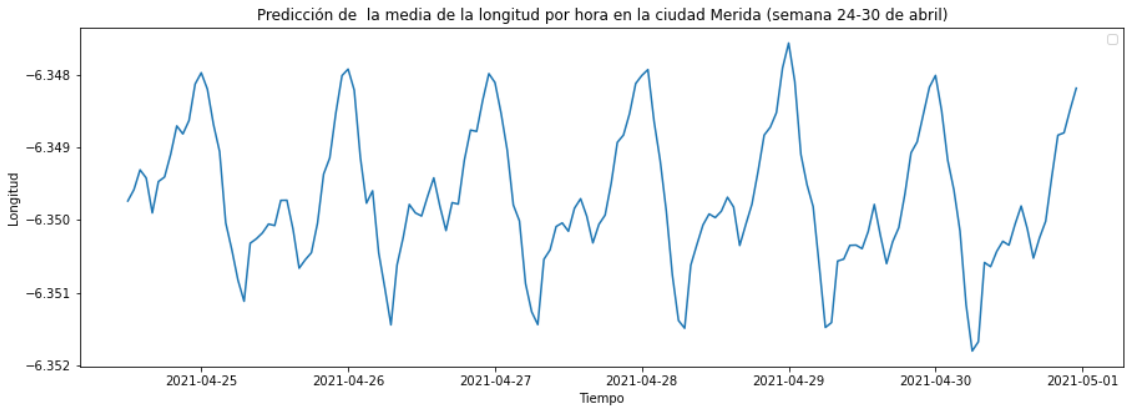


**Ilustración 71: Predicción de la media de la longitud por hora (semana 24-30 de abril)**

Asimismo, la previsión de la longitud media por hora para la semana comprendida entre el 22 y el 28 de mayo es la siguiente.

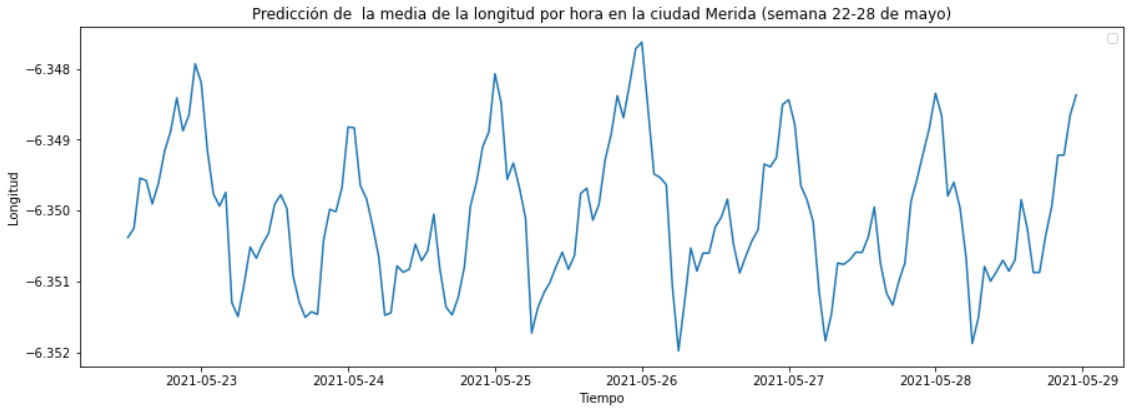


**Ilustración 72: Predicción de la media de la longitud por hora (semana 22-28 de mayo)**

Comparando ambas gráficas, se tiene un comportamiento similar al que se obtuvo en la ilustración 49. Los valores son muy parecidos en términos de longitud.

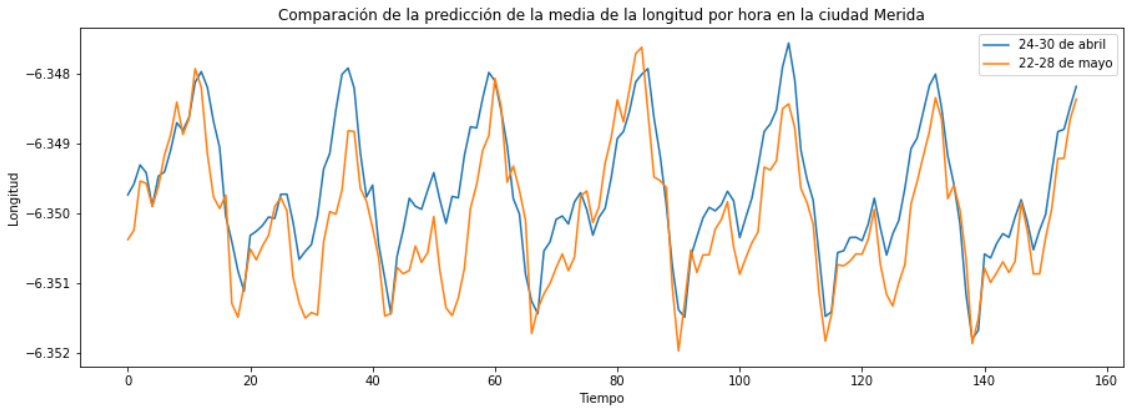


**Ilustración 73: Comparación de las predicciones de longitud por hora**

Para ver todo esto más claro, se va a realizar una representación geográfica de los puntos más relevantes de las predicciones.

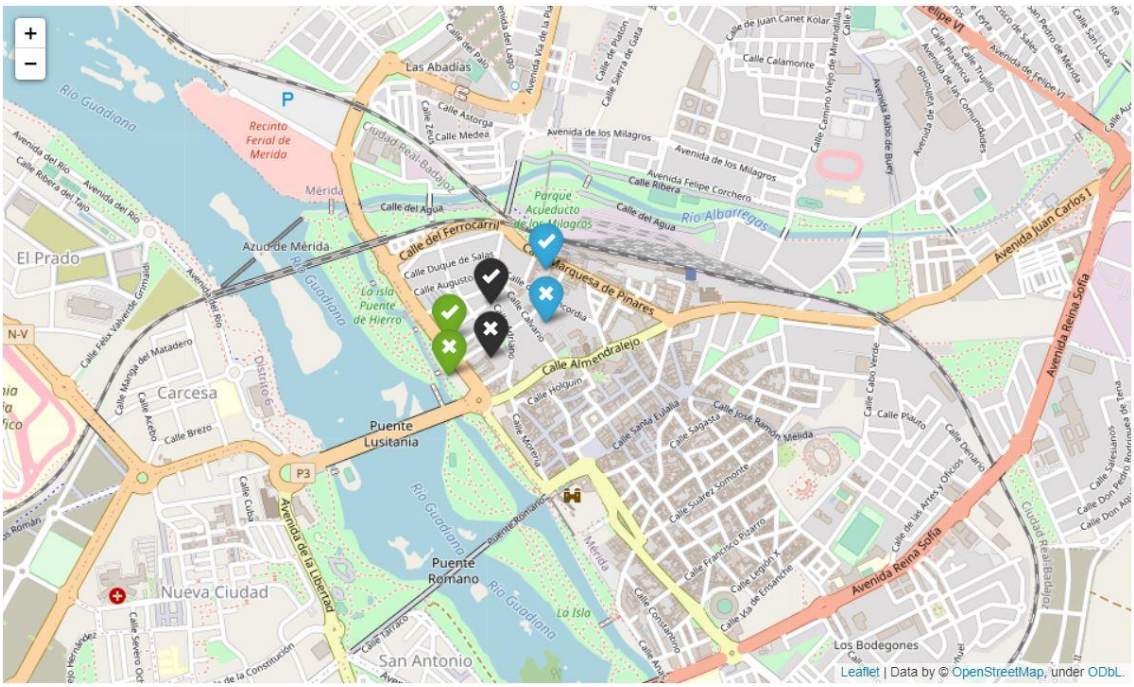

**Ilustración 74: Representación puntos geográficos de la predicción**

Los iconos que contienen una x representan al mes de abril que tenía restricciones, mientras que por el contrario el tic al mes de mayo. El color azul corresponde a primera hora de la mañana (5-8:00), el color verde a las horas centrales del día (10-17:00) y el negro a la tarde-noche (20:00-23:00). Los puntos corresponden a la media de las coordenadas de latitud y longitud de las predicciones realizadas anteriormente.

Se aprecia claramente un incremento de latitud para los días sin restricciones (mayo) con respecto a los días con restricciones (abril) en los tres intervalos principales del día. Sin embargo, la longitud permanece constante. Todo esto tiene sentido dado que el norte de la ciudad contiene numerosas instalaciones deportivas, empresas, parques y sedes autonómicas. Gracias a la eliminación de medidas como el toque de queda, la reducción de aforos en interiores y exteriores y vuelta a la organización de eventos, entre otras, se ve reflejada una ligera variación en la posición geográfica de las conexiones móviles en todos los tramos del día.

# 4. Conclusiones

A lo largo del proyecto, se ha ido viendo como las restricciones por COVID-19 afectaron directamente a la movilidad de las personas en la ciudad.

En el capítulo de análisis de datos, se realizó un exhaustivo estudio de las diferentes zonas de la ciudad y las variaciones de las conexiones móviles que se dieron entre el mes de abril de 2021 (Estado de Alarma vigente) y el mes de mayo (sin Estado de Alarma).

A partir de los diferentes mapas mostrados, se pudo observar que los fines de semana de mayo de 2021, presentaban una mayor afluencia que los de abril en la mayor parte de la ciudad. También, haciendo uso de los mapas de calor, se apreciaron cambios significativos. Se llega a la conclusión, que estas variaciones están justificadas en la reactivación del turismo como consecuencia del levantamiento de los cierres perimetrales de las provincias y comunidades autónomas.

Después, en el capítulo de *Machine Learning*, se vio que al trabajar con datos que contenían días y hora exactos, lo mejor era realizar series temporales. Por ello, se empezó haciendo un análisis de las diferentes series temporales más importantes como el número de conexiones diarias por hora y la media de la latitud y longitud de las estaciones base que recogen estas conexiones.

Para elegir el algoritmo más adecuado, se tuvo en cuenta la presencia de componentes estacionales presentes en las diferentes series temporales.

Por esta razón, se estudiaron diferentes modelos, siendo el SARIMA el más adecuado. Se trata de un modelo estadístico que utiliza variaciones y regresiones de los datos con la finalidad de encontrar patrones y así realizar predicciones a futuro. Para establecer el rango de parámetros más adecuados del modelo, se usaron tanto la función parcial de autocorrelación (PACF) como la función de autocorrelación (ACF). Además, el modelo SARIMA implica que la serie temporal ha de ser estacionaria, por lo que se realizó un desplazamiento de esta para eliminar la estacionalidad.

A partir de los diferentes rangos de parámetros, se fueron realizando las predicciones de las series temporales. Para establecer los indicadores de éxito o de fracaso, se trabaja con un dataset de prueba y otro de entrenamiento con el propósito de comparar las predicciones resultantes con los datos de prueba. Los datos de entrenamiento servirán por tanto para entrenar el modelo. Las diferencias entre la predicción y los datos de prueba se estudian a partir del error cuadrático medio (MSE), con el objetivo de reducirlo lo máximo posible. Así, cuanto menor sea el error más se parecerá la gráfica de la predicción a la de los datos de prueba.

Gracias a estos datos, se pudo ver con claridad los periodos de mayor actividad a lo largo de la semana y la distribución geográfica aproximada de las conexiones diarias en la ciudad. Se demostró que, tanto en mayo como en

abril, las series temporales seguían el mismo patrón, pero con un claro aumento de conexiones en todos los tramos del día para el mes de mayo. Esto tal y como hemos comentado varias veces, tiene su lógica. En cuanto a las coordenadas geográficas, también se argumentaron variaciones destacadas, especialmente en la latitud.

Por otra parte, se han logrado todos los objetivos que se plantearon en la PEC1. La obtención del dataset a partir de una base de datos externa, el diseño e implementación del sistema de tratamiento de los datos para su manipulación y análisis y la representación de estos usando diferentes herramientas de visualización.

Además, también se ha logrado el aprendizaje y la utilización del lenguaje de programación Python y un estudio de las diferentes técnicas de *Machine Learning* para la aplicación de modelos predictivos sobre los datos.

Gracias a ello, se pudieron implementar los modelos necesarios para la predicción de las series temporales de los datos.

En cuanto a la planificación del proyecto, se ha llevado a cabo cumpliendo cada hito programado. Tras una primera PEC, donde se estableció el primer capítulo con los objetivos y alcance del proyecto y una segunda PEC en la que se desarrolló el Estado del Arte, la tercera PEC se dividió en tres partes: el proceso de ETL, el análisis de datos y los algoritmos predictivos de Machine Learning.

Las aplicaciones del presente proyecto están ligadas directamente a la movilidad y al desarrollo sostenible. Teniendo en cuenta los principales tramos horarios en los que se produce un mayor o menor número de conexiones, se pueden aumentar o disminuir los recursos necesarios, tales como iluminación o servicios primarios en las zonas que presenten una mayor o menor incidencia respectivamente. Esto permitirá un ahorro o refuerzo de recursos a determinadas horas del día si fuera necesario.

En cuanto a las limitaciones del proyecto, al tratar con grandes cantidades de datos y al uso de un ordenador personal, analizar más de dos semanas era prácticamente imposible. Por eso, se decidió realizar el análisis y las predicciones basándose en un grupo reducido de días.

Con respecto a las líneas de futuro a seguir del presente proyecto, sería conveniente hacer hincapié en las zonas de la ciudad en las que se ha producido un mayor aumento de conexiones y por lo tanto de la afluencia de gente tras la caída del Estado de Alarma.

Así, se podrán reforzar los diferentes servicios de la ciudad en función de cada tramo horario para promover la movilidad inteligente.

# 5. Glosario

| | |
|---|---|
| GPRS | General Packet Radio Service |
| EDGE | Enhanced Data Rates for GSM Evolution |
| TDMA | Time Division Multiple Access |
| W-CDMA | Wideband Code Division Multiple Access |
| UMTS | Universal Mobile Telecommunication System |
| LTE | Long Term Evolution |
| 5G NSA | 5G Non-Standalone |
| 5G SA | 5G Standalone |
| SQL | Structured Query Language |
| NoSQL | Bases de datos que no son relacionales. |
| ETL | Extract, Transform y Load. |

# 6. Bibliografía

[1] https://es.statista.com/temas/6510/impacto-del-coronavirus-en-el-sector-turistico-en-espana/

[2] https://www.diariodemallorca.es/mallorca/2021/01/13/aeropuerto-palma-cierra-desplome-79-28061391.html

[3] https://www.geotab.com/es/blog/movilidad-inteligente/

[4] https://blogdatlas.wordpress.com/2020/02/16/4-metodologias-para-proyectos-de-data-science-datlas-research/

[5] https://www.lamoncloa.gob.es/covid-19/Paginas/estado-de-alarma.aspx

[6] https://www.elespanol.com/reportajes/20210326/poder-semana-santa-comunidad-cierre-movilidad-aforos/568694886_0.html

[7] https://radio-waves.orange.com/es/como-funciona-una-red-movil/

[8] http://intotally.com/tot4blog/2013/07/11/lte-for-beginners-day-2-brief-description-about-lte-network-architecture/?lang=es

[9] https://www.adslzone.net/reportajes/tecnologia/tecnologia-telefonia-movil/

[10] https://dasci.es/es/investigacion/unidades/ciencia-de-datos-y-big-data/

[11] https://datos.gob.es/es/blog/ciencia-de-datos-machine-learning-y-deep-learning

[12] https://netmind.net/es/analisis-de-datos-en-big-data/

[13] https://netmind.net/es/analisis-de-datos-en-big-data/

[14] A. Silberschatz, H. F. Horth y S. Sudarshan, (2014) “Fundamentos de Bases de Datos” (6ª Edición). McGraw-Hill.

[15] Oracle. - https://www.oracle.com

[16] Microsoft SQL server.- https://www.microsoft.com/es-es/sql-server

[17] MySql. - https://www.mysql.com/

[18] SQL en Wikipedia http://es.wikipedia.org/wiki/SQL

[19] Python. - https://www.python.org/

[20] https://hardzone.es/tutoriales/rendimiento/diferencias-ia-deep-machine-learning/

[21] https://www.apd.es/algoritmos-del-machine-learning/

[22] https://forraxa.github.io/Rcaride/algoritmos_de_regresion_r.html

[23] https://www.tokioschool.com/noticias/redes-neuronales-machine-mearning/

[24] https://www.datahack.es/blog/big-data/algoritmos-analisis-predictivo/

[25] https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/

[26] https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/#

[27] https://es.wikipedia.org/wiki/Teorema_de_Bayes

[28] https://www.grapheverywhere.com/algoritmos-de-machine-learning/

[29] https://www.apd.es/algoritmos-del-machine-learning/

[30] Folium.- https://mappinggis.com/2018/10/folium-utilizando-leaflet-con-python/

[31] Leafset. - https://en.wikipedia.org/wiki/Leaflet_(software)

[32] JavaScript. - https://www.javascript.com/

[33] https://www.openstreetmap.org

[34] https://www.ine.es/

[35] https://administracion.gob.es/pag_Home/espanaAdmon/comoSeOrganizaEstado/Administracion_Gral_Estado.html

[36] https://portal.mineco.gob.es/es-es/Paginas/default.aspx

[37] https://www.ine.es/covid/exp_movilidad_covid_proyecto.pdf

[38] https://experience.arcgis.com/experience/5e442514cc604efc87ff11c0c3fff8ed/

[39] https://en.wikipedia.org/wiki/Oracle_SQL_Developer

[40] https://ingenieriadesoftware.es/tipos-sql-join-guia-referencia/

[41] https://es.climate-data.org/

[42] https://es.wikipedia.org/wiki/SQL*Plus

[43] https://es.wikipedia.org/wiki/Archivo_batch

[44] https://jupyter.org/

[45] https://es.acervolima.com/2021/02/09/python-modelo-arima-para-el-pronostico-de-series-de-tiempo/

# 7. Anexos

## 7.1 Código Python análisis de datos

IMPORTAMOS TODO LO NECESARIO PARA TRABAJAR

```
import matplotlib.pyplot as plt
import matplotlib.cm
import pandas as pd
import numpy as np
import folium
from folium.plugins import HeatMap
# importar el módulo geopy
import geopy
# crear un objeto de servicio
service = geopy.Nominatim(user_agent = "myGeocoder")
import statistics


# Algunas librerias extra que usaremos
from matplotlib.colors import Normalize
import matplotlib.colors as colors
from numpy import array
from numpy import max
```

CARGAMOS LOS FICHEROS EN PYTHON PARA TRABAJAR CON ELLOS Y CONVERTIMOS EL CAMPO DE TEXTO A FECHA

```
#CARGAMOS LOS DATOS DE TRAFICO MOVIL
#MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file210421 = '../DATOS\datos210421.csv'
datos210421 = pd.read_csv(file210421, sep=';', engine='python')
datos210421['IT_REGISTRO'] = pd.to_datetime(datos210421['IT_REGISTRO'])

#SABADO 24 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file240421 = '../DATOS\datos240421.csv'
datos240421 = pd.read_csv(file240421, sep=';', engine='python')
datos240421['IT_REGISTRO'] = pd.to_datetime(datos240421['IT_REGISTRO'])

#SABADO 22 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO
file220521 = '../DATOS\datos220521.csv'
datos220521 = pd.read_csv(file220521, sep=';', engine='python')
datos220521['IT_REGISTRO'] = pd.to_datetime(datos220521['IT_REGISTRO'])

#MIERCOLES 26 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO
file260521 = '../DATOS\datos260521.csv'
datos260521 = pd.read_csv(file260521, sep=';', engine='python')
datos260521['IT_REGISTRO'] = pd.to_datetime(datos260521['IT_REGISTRO'])
```

TÍTULO: COMPARACIÓN DE DOS MIÉRCOLES: 21/04/2021 Y 26/05/2021

RESUMEN: SE REALIZA UN CONTEO AGRUPADO DE LOS REGISTROS QUE APARECEN POR CADA LOCALIZACIÓN Y SE INTRODUCEN LOS VALORES EN UNA LISTA. DESPUÉS SE COMPARAN LAS DOS LISTAS DE LOS DOS MIÉRCOLES CALCULANDO LOS PORCENTAJES ENTRE MAYO Y ABRIL Y SE CREA OTRA LISTA CON LOS VALORES. LUEGO SE CREA UNA NUEVA LISTA QUE CONTENGA ADEMÁS DE LO ANTERIOR LOS

COLORES VERDE CUANDO HA AUMENTADO Y ROJO CUANDO HAYA DISMINUIDO PARA REPRESENTARLOS EN EL MAPA. POR ÚLTIMO, SE CREA EL MAPA REPRESENTANDO CADA LOCALIZACION CON LOS COLORES PARA VER SI HA AUMENTADO O DISMINUIDO

```
#MIERCOLES DE ABRIL VS MIERCOLES DE MAYO
#CONTEO DE NUMERO DE REGISTROS TRAFICO MOVIL POR PAR LATITUD LONGITUD

grouped_data210421 = datos210421.groupby(["LATITUD", "LONGITUD"])

lista210421 = grouped_data210421.count().reset_index().values.tolist()

grouped_data260521 = datos260521.groupby(["LATITUD", "LONGITUD"])

lista260521 = grouped_data260521.count().reset_index().values.tolist()

#PORCENTAJE DE AUMENTO DE REGISTROS DIA SIN RESTRICCIONES VS DIA CON RESTRICCIONES MIERCOLES

#lista_aumento_mi.clear()
lista_aumento_mi = list()
for i in range(len(lista210421)):
    j = 0
    while (lista260521[j][0] != lista210421[i][0]):
        j += 1
        if j == len(lista260521):
            break
    if j != len(lista260521):
        lista_aumento_mi.append([lista260521[j][0], lista260521[j][1], ((lista260521[j][2]/lista210421[i][2])-1)*100])

#lista_aumento_mi_color.clear()
lista_aumento_mi_color = list()
for i in range(len(lista_aumento_mi)):
    if lista_aumento_mi[i][2] > 3:
        color = 'green'
    elif -3 <= lista_aumento_mi[i][2] <= 3:
        color = 'gray'
    else:
        color = 'red'

    lista_aumento_mi_color.append([lista_aumento_mi[i][0], lista_aumento_mi[i][1], lista_aumento_mi[i][2], color])

##MIERCOLES 26 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO VS MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO

lats = datos210421['LATITUD']
longs = datos210421['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong = statistics.mean(longs)
meanLat = statistics.mean(lats)


# crear un objeto mapa base usando Map()
mapObj = folium.Map(location=[meanLat, meanLong], zoom_start = 14)

for i in range(len(lista_aumento_mi_color)):
```

```
    marcador1 = folium.Marker(location=(lista_aumento_mi_color[i][0], lista_aumento_mi_color[i][1]), popup=lista_aumento_mi[i][2], icon=folium.Icon(lista_aumento_mi_color[i][3]))
    marcador1.add_to(mapObj)


mapObj
```

TÍTULO: COMPARACIÓN DE DOS SÁBADOS: 22/05/2021 Y 24/04/2021

RESUMEN: SE REALIZA UN CONTEO AGRUPADO DE LOS REGISTROS QUE APARECEN POR CADA LOCALIZACIÓN Y SE INTRODUCEN LOS VALORES EN UNA LISTA. DESPUÉS SE COMPARAN LAS DOS LISTAS DE LOS DOS MIÉRCOLES CALCULANDO LOS PORCENTAJES ENTRE MAYO Y ABRIL Y SE CREA OTRA LISTA CON LOS VALORES. LUEGO SE CREA UNA NUEVA LISTA QUE CONTENGA ADEMÁS DE LO ANTERIOR LOS COLORES VERDE CUANDO HA AUMENTADO Y ROJO CUANDO HAYA DISMINUIDO PARA REPRESENTARLOS EN EL MAPA. POR ÚLTIMO, SE CREA EL MAPA REPRESENTANDO CADA LOCALIZACION CON LOS COLORES PARA VER SI HA AUMENTADO O DISMINUIDO

```
#CONTEO DE NUMERO DE REGISTROS TRAFICO MOVIL POR PAR LATITUD LONGITUD

#datos210421.groupby(["LATITUD", "LONGITUD"])["IT_REGISTRO"].count().sort_values(ascending=False)

grouped_data240421 = datos240421.groupby(["LATITUD", "LONGITUD"])

lista240421 = grouped_data240421.count().reset_index().values.tolist()

grouped_data220521 = datos220521.groupby(["LATITUD", "LONGITUD"])

lista220521 = grouped_data220521.count().reset_index().values.tolist()

#PORCENTAJE DE AUMENTO DE REGISTROS DIA SIN RESTRICCIONES VS DIA CON RESTRICCIONES MIERCOLES

#lista_aumento_sa.clear()
lista_aumento_sa = list()
for i in range(len(lista240421)):
    j = 0
    while (lista220521[j][0] != lista240421[i][0]):
        j += 1
        if j == len(lista220521):
            break
    if j != len(lista220521):
        lista_aumento_sa.append([lista220521[j][0], lista220521[j][1], ((lista220521[j][2]/lista240421[i][2])-1)*100])

#lista_aumento_sa_color.clear()
lista_aumento_sa_color = list()
for i in range(len(lista_aumento_sa)):
    if lista_aumento_sa[i][2] > 3:
        color = 'green'
    elif -3 <= lista_aumento_sa[i][2] <= 3:
        color = 'gray'
    else:
        color = 'red'

    lista_aumento_sa_color.append([lista_aumento_sa[i][0], lista_aumento_sa[i][1], lista_aumento_sa[i][2], color])
```

#SABADO 22 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO VS SABADO 24 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO

```
lats = datos210421['LATITUD']
longs = datos210421['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong = statistics.mean(longs)
meanLat = statistics.mean(lats)


# crear un objeto mapa base usando Map()
mapObj = folium.Map(location=[meanLat, meanLong], zoom_start = 14)

for i in range(len(lista_aumento_sa_color)):
    marcador1 = folium.Marker(location=(lista_aumento_sa_color[i][0], lista_aumento_sa_color[i][1]), popup=lista_aumento_sa_color[i][2], icon=folium.Icon(lista_aumento_sa_color[i][3]))
    marcador1.add_to(mapObj)
```

GRÁFICAS MOSTRANDO EL INCREMENTO

```
#lista_grafica_aux.clear()
lista_grafica_aux = list()

for i in range(len(lista_aumento_sa)):
    if lista_aumento_sa[i][0] == 38.91377623:
        lista_grafica_aux.append(['Anfiteatro', lista_aumento_sa[i][0], lista_aumento_sa[i][1], lista_aumento_sa[i][2]])
    elif lista_aumento_sa[i][0] == 38.91391664:
        lista_grafica_aux.append(['Aparcamiento', lista_aumento_sa[i][0], lista_aumento_sa[i][1], lista_aumento_sa[i][2]])
    elif lista_aumento_sa[i][0] == 38.91661155:
        lista_grafica_aux.append(['Puente Romano', lista_aumento_sa[i][0], lista_aumento_sa[i][1], lista_aumento_sa[i][2]])
    elif lista_aumento_sa[i][0] == 38.91720914:
        lista_grafica_aux.append(['Templo Diana', lista_aumento_sa[i][0], lista_aumento_sa[i][1], lista_aumento_sa[i][2]])


#lista_grafica.clear()
lista_grafica = list()


for i in range(len(lista_aumento_mi)):
    if lista_aumento_mi[i][0] == lista_grafica_aux[0][1]:
        lista_grafica.append([lista_grafica_aux[0][0], lista_grafica_aux[0][1], lista_grafica_aux[0][2],lista_grafica_aux[0][3], lista_aumento_mi[i][2]])
    elif lista_aumento_mi[i][0] == lista_grafica_aux[1][1]:
        lista_grafica.append([lista_grafica_aux[1][0], lista_grafica_aux[1][1], lista_grafica_aux[1][2],lista_grafica_aux[1][3], lista_aumento_mi[i][2]])
    elif lista_aumento_mi[i][0] == lista_grafica_aux[2][1]:
        lista_grafica.append([lista_grafica_aux[2][0], lista_grafica_aux[2][1], lista_grafica_aux[2][2],lista_grafica_aux[2][3], lista_aumento_mi[i][2]])
    elif lista_aumento_mi[i][0] == lista_grafica_aux[3][1]:
        lista_grafica.append([lista_grafica_aux[3][0], lista_grafica_aux[3][1], lista_grafica_aux[3][2],lista_grafica_aux[3][3], lista_aumento_mi[i][2]])

#lugares.clear()
```

```
#aumento_sabado.clear()
#aumento_miercoles.clear()
lugares = list()
aumento_sabado = list()
aumento_miercoles = list()

for i in range(len(lista_grafica)):
    lugares.append(lista_grafica[i][0])
    aumento_sabado.append(lista_grafica[i][3])
    aumento_miercoles.append(lista_grafica[i][4])

indice = np.arange(len(lugares))
ancho = 0.35

plt.bar(indice, aumento_miercoles, width=ancho, label='Miércoles')
plt.bar(indice + ancho, aumento_sabado, width=ancho, label='Sábado')

plt.legend(loc='best')

plt.xticks(indice, lugares)
plt.ylabel("Porcentaje")
plt.xlabel("Lugares")
plt.title('Porcentaje por lugar')

plt.show()
```

MAPA DE CALOR MIERCOLES 21 DE ABRIL

```
##MIERCOLES 21 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
lats210421 = datos210421['LATITUD']
longs210421 = datos210421['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong210421 = statistics.mean(longs210421)
meanLat210421 = statistics.mean(lats210421)


# crear un objeto mapa base usando Map()
mapObj210421 = folium.Map(location=[meanLat210421, meanLong210421], zoom_start = 14)


# crear capa de mapa de calor
heatmap210421 = HeatMap( list(zip(lats210421, longs210421)),
                min_opacity=0.6,
                radius=50, blur=50,
                max_zoom=1)
# añadir capa de mapa de calor al mapa base
heatmap210421.add_to(mapObj210421)
mapObj210421
```

MAPA DE CALOR MIERCOLES 26 DE MAYO

```
#MIERCOLES 26 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO
lats260521 = datos260521['LATITUD']
longs260521 = datos260521['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong260521 = statistics.mean(longs260521)
meanLat260521 = statistics.mean(lats260521)
```

```
# crear un objeto mapa base usando Map()
mapObj260521 = folium.Map(location=[meanLat260521, meanLong260521], zoom_start = 14)


# crear capa de mapa de calor
heatmap260521 = HeatMap( list(zip(lats260521, longs260521)),
                min_opacity=0.6,
                radius=50, blur=50,
                max_zoom=1)
# añadir capa de mapa de calor al mapa base
heatmap260521.add_to(mapObj260521)
mapObj260521
```

MAPA DE CALOR SÁBADO 24 DE ABRIL

```
#SABADO 24 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
lats240421 = datos240421['LATITUD']
longs240421 = datos240421['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong240421 = statistics.mean(longs240421)
meanLat240421 = statistics.mean(lats240421)


# crear un objeto mapa base usando Map()
mapObj240421 = folium.Map(location=[meanLat240421, meanLong240421], zoom_start = 14)


# crear capa de mapa de calor
heatmap240421 = HeatMap( list(zip(lats240421, longs240421)),
                min_opacity=0.6,
                radius=50, blur=50,
                max_zoom=1)
# añadir capa de mapa de calor al mapa base
heatmap240421.add_to(mapObj240421)
mapObj240421
```

MAPA DE CALOR SÁBADO 22 DE MAYO

```
#SABADO 22 DE MAYO MERIDA SIN RESTRICCIONES BUEN TIEMPO
lats220521 = datos220521['LATITUD']
longs220521 = datos220521['LONGITUD']

# calcular los valores medios de longitud y latitud
meanLong220521 = statistics.mean(longs220521)
meanLat220521 = statistics.mean(lats220521)


# crear un objeto mapa base usando Map()
mapObj220521 = folium.Map(location=[meanLat220521, meanLong220521], zoom_start = 14)


# crear capa de mapa de calor
heatmap220521 = HeatMap( list(zip(lats220521, longs220521)),
                min_opacity=0.6,
                radius=50, blur=50,
                max_zoom=1)
# añadir capa de mapa de calor al mapa base
```

```
heatmap220521.add_to(mapObj220521)
mapObj220521
```

## 7.2 Código predicción series temporales

### 7.2.1 Número de conexiones por hora (igual, tanto para abril como para mayo)

```
import pickle

import scipy.stats
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)
import matplotlib as mpl
from sklearn import svm
from sklearn import ensemble
from matplotlib import pyplot as plt
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score, train_test_split, validation_curve
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
from tqdm import tqdm
import seaborn as sbn

%matplotlib inline

#CARGAMOS LOS DATOS DE TRAFICO MOVIL POR HORA
#SEMANA 15-29 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file192304 = '../DATOS\datos15290421.csv'
datosdias192304 = pd.read_csv(file192304, sep=';', engine='python')
datosdias192304['IT_REGISTRO'] = datosdias192304['IT_REGISTRO'].str[:11]
datosdias192304['IT_REGISTRO'] = pd.to_datetime(datosdias192304['IT_REGISTRO'])

grouped_data192304 = datosdias192304.groupby(["IT_REGISTRO"])
lista192304 = grouped_data192304.count().reset_index().values
dataframe_lista192304 = pd.DataFrame(lista192304)
dataframe_lista192304 = dataframe_lista192304.drop(dataframe_lista192304.columns[2],
axis='columns')
dataframe_lista192304.columns = ['ts', 'registros']
dataframe_lista192304 = dataframe_lista192304.set_index('ts')
dataframe_lista192304["registros"] = pd.to_numeric(dataframe_lista192304["registros"])

date_min_test = "24/04/21 12:00:00"
TEST_SIZE = len(dataframe_lista192304[dataframe_lista192304.index>=date_min_test])
# train and test are dataframes that contain the values and a datetime index:
train, test = dataframe_lista192304.iloc[:-TEST_SIZE], dataframe_lista192304.iloc[-
TEST_SIZE:]
# x_train and x_test only contain the values:
x_train, x_test = np.array(range(train.shape[0])), np.array(range(train.shape[0],
dataframe_lista192304.shape[0]))
train.shape, x_train.shape, test.shape, x_test.shape

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
```

```
ax.plot(x_train, train)
ax.plot(x_test, test);
#train.plot(ax=ax, label='train')
#test.plot(ax=ax, label='test')

dataframe_shifted = dataframe_lista192304 - dataframe_lista192304.shift(24)

fig, ax = plt.subplots(1, 1, figsize=(10, 10))
plot_pacf(dataframe_shifted[24:], ax, lags = 47)
plt.show()

fig, ax = plt.subplots(1, 1, figsize=(10, 10))
plot_acf(dataframe_shifted[24:], ax, lags = 47)
plt.show()

sarima = SARIMAX(train,order=(6,0,8),seasonal_order=(10,1,3,24),).fit();
sarima.summary()

pred_abril = sarima.predict('24/04/21 12:00:00','30/04/21 23:00:00')

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train, train, label='Entrenado')
ax.plot(x_test, pred, label='Test estimado')
ax.plot(x_test, test, label='Test real')

plt.xlabel('Tiempo')
plt.ylabel('Registros totales por hora')
plt.title('Predicción de los registros totales por hora en la ciudad de Merida')
ax.legend();

pred = sarima.predict('24/04/21 12:00:00','30/04/21 23:00:00')

fig, ax = plt.subplots(1, 1, figsize=(15, 5))

ax.plot( pred_abril)
plt.xlabel('Tiempo')
plt.ylabel('Registros totales por hora')
plt.title('Predicción de los registros totales por hora en la ciudad de Merida (semana 24-30 de abril)')
ax.legend();
```

### 7.2.2 Latitud y longitud media por hora (igual, tanto para abril como para mayo)

```
#CARGAMOS LOS DATOS DE TRAFICO MOVIL
#SEMANA 15-29 DE ABRIL MERIDA CON RESTRICCIONES BUEN TIEMPO
file192304 = '../DATOS\datos15290421.csv'
datosdias192304 = pd.read_csv(file192304, sep=';', engine='python')
datosdias192304['IT_REGISTRO'] = datosdias192304['IT_REGISTRO'].str[:11]
datosdias192304['IT_REGISTRO'] = pd.to_datetime(datosdias192304['IT_REGISTRO'])

timestamps192304 =
list(datosdias192304.drop_duplicates('IT_REGISTRO').sort_values('IT_REGISTRO')['IT_REGISTRO'])

lat_mean = []
long_mean = []

for ts in tqdm(timestamps192304):
```

```
    data_ts = datosdias192304[datosdias192304["IT_REGISTRO"] == ts]
    lat_mean.append(np.mean(data_ts["LATITUD"]))
    long_mean.append(np.mean(data_ts["LONGITUD"]))

new_dataframe_192304 = pd.DataFrame(data={"lat":lat_mean, "long":long_mean,
"ts":timestamps192304})
new_dataframe_192304 = new_dataframe_192304.set_index('ts')
new_dataframe_lat_192304 =  new_dataframe_192304["lat"]
new_dataframe_long_192304 =  new_dataframe_192304["long"]

date_min_test = "24/04/21 12:00:00"
TEST_SIZE =
len(new_dataframe_lat_192304[new_dataframe_lat_192304.index>=date_min_test])
# train and test are dataframes that contain the values and a datetime index:
train, test = new_dataframe_lat_192304.iloc[:-TEST_SIZE], new_dataframe_lat_192304.iloc[-
TEST_SIZE:]
# x_train and x_test only contain the values:
x_train, x_test = np.array(range(train.shape[0])), np.array(range(train.shape[0],
new_dataframe_lat_192304.shape[0]))
train.shape, x_train.shape, test.shape, x_test.shape

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train, train)
ax.plot(x_test, test);
#train.plot(ax=ax, label='train')
#test.plot(ax=ax, label='test')

dataframe_shifted = new_dataframe_lat_192304 - new_dataframe_lat_192304.shift(24)

fig, ax = plt.subplots(1, 1, figsize=(10, 10))
plot_pacf(dataframe_shifted[24:], ax, lags = 47)
plt.show()

fig, ax = plt.subplots(1, 1, figsize=(10, 10))
plot_acf(dataframe_shifted[24:], ax, lags = 47)
plt.show()

sarima = SARIMAX(train,order=(3,0,8),seasonal_order=(6,1,1,24),).fit();
sarima.summary()

pred = sarima.predict('24/04/21 12:00:00','29/04/21 23:00:00')

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train, train, label='Entrenado')
ax.plot(x_test, pred, label='Test estimado')
ax.plot(x_test, test, label='Test real')

plt.xlabel('Tiempo')
plt.ylabel('Latitud')
plt.title('Predicción de la media de la latitud por hora en la ciudad Merida (abril)')
ax.legend();

pred = sarima.predict('24/04/21 12:00:00','30/04/21 23:00:00')

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot( pred)
plt.xlabel('Tiempo')
plt.ylabel('Latitud')
plt.title('Predicción de  la media de la latitud por hora en la ciudad Merida (semana 24-30 de
abril)')
```

```
ax.legend();
```