GRADO EN INGENIERÍA DE ORGANIZACIÓN INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

|  |
| --- |
|  |

Imagen que contiene dibujo

Descripción generada automáticamente

**Tutor** Juan Carlos Rubio Romero

**Cotutora** María Martínez Rojas

**Área de conocimiento** Organización de empresas  
Departamento de Economía y administración de empresas

MAXIMILIANO GÁMEZ LÓPEZ

JUNIO 2020

TRABAJO FIN DE GRADO

17

**Aplicación de técnicas de minería de datos sobre registros de operaciones del sistema de alquiler público de bicicletas de Madrid (BiciMAD)**

RESUMEN:

El empleo de técnicas emergentes como la minería de datos, que pretende la identificación y explotación de patrones en grandes conjuntos de datos («*Big Data*»), se encuentran cada vez más presentes en las organizaciones y sus procesos. Alentadas por la transformación digital y la denominada Cuarta Revolución Industrial, integran este tipo de herramientas en sus procesos, resultando en ventajas competitivas. Este Trabajo Fin de Grado documenta el proceso y la metodología empleada para el desarrollo – adquisición, preprocesamiento y entrenamiento – de un modelo de predicción de la demanda del servicio de bicicleta compartida BiciMAD, en la ciudad de Madrid. Se han empleado metalgoritmos de aprendizaje como los árboles extremadamente aleatorios, entrenados a partir de conjuntos de datos abiertos publicados por EMT Madrid, resultando en modelo predictivo con un desempeño de , considerando solamente variables meteorológicas y de estacionalidad. De igual forma se recogen lecciones aprendidas y posibles puntos de mejora en un proceso de implantación real en la organización.

PALABRAS CLAVE:

Minería de datos, Big Data, aprendizaje automático, estadística predictiva, bicicleta compartida, BiciMAD, pronóstico de demanda, árboles de decisión.

ABSTRACT:

Emerging usage of tools such as data mining, which is intended to identify and exploit hidden patterns in massive datasets (Big Data), is increasingly present in organizations and their processes. Encouraged by digital transformation and the current Fourth Industrial Revolution, they incorporate these tools into value chain in order to achieve competitive advantages. Present Final Degree Project documents the process and methodology used to develop (data acquisition, preprocessing and training) a model which forecast demand of bike sharing system BiciMAD, in Madrid city. For this, learning metalgorithms, as extremely randomized trees, have been trained from open datasets released by Madrid municipal transport company, resulting in a predictive model which performs with , only considering weather and seasonality issues. Furthermore, a set of lessons learnt as well as potential points of improvement, in case of real deployment, have been recorded.

KEYWORDS:

Data mining, big data, automatized learning, predictive statistics, bike sharing, BiciMAD, demand forecasting, decision trees.

**ÍNDICE DE CONTENIDOS:** Página

[1. Objetivos y alcance 11](#_Toc42012016)

[2. Introducción 12](#_Toc42012017)

[3. BiciMAD 14](#_Toc42012018)

[3.1 Aceptación del servicio y tarifas 16](#_Toc42012019)

[3.2 Cómo usarlo 19](#_Toc42012020)

[4. Minería de datos 21](#_Toc42012021)

[5. Datos empleados 32](#_Toc42012022)

[5.1 Adquisición 32](#_Toc42012023)

[5.2 Preprocesamiento 38](#_Toc42012024)

[5.3 Unión en un único conjunto de datos 41](#_Toc42012025)

[6. Análisis de datos disponibles 45](#_Toc42012026)

[7. Líneas de trabajo con minería de datos 54](#_Toc42012027)

[7.1 Procesamiento de datos: Imputación de valores omitidos 55](#_Toc42012028)

[7.2 Aprendizaje no supervisado: reducción dimensional 59](#_Toc42012029)

[7.3 Selección de Modelo 65](#_Toc42012030)

[7.4 Definición de parámetros 70](#_Toc42012031)

[7.5 Validación de árboles extremadamente aleatorios y bosque aleatorio 80](#_Toc42012032)

[7.6 Líneas de trabajo para la mejora del modelo 86](#_Toc42012033)

[8. Conclusiones 89](#_Toc42012034)

[9. BIBLIOGRAFÍA 93](#_Toc42012035)

[10. ANEXOS 101](#_Toc42012036)

[Anexo I: Código Python de consultas de trayectos a API de Openroute Service 102](#_Toc42012037)

[Anexo II: Código Python correspondiente a conversión a CSV, filtrado y unificación de datos estáticos de desplazamientos de BiciMAD 104](#_Toc42012038)

[Anexo III: Código Python correspondiente a asociación de datos en un solo conjunto (desplazamientos + distancia y duración + meteo) 106](#_Toc42012039)

[Anexo IV: Código Python correspondiente a generación de gráficos para el análisis descriptivo de los datos empleados 108](#_Toc42012040)

[Anexo V: Código Python correspondiente a imputación de valores omitidos en conjunto de registros meteorológicos 112](#_Toc42012041)

[Anexo VI: Código Python correspondiente a generación de conjunto de datos correspondiente a demanda del sistema BiciMAD. 117](#_Toc42012042)

[Anexo VII: Código Python correspondiente a número de variables a considerar en análisis PCA 119](#_Toc42012043)

[Anexo VIII: Código Python empleado en la selección de modelo (sin PCA) 121](#_Toc42012044)

[Anexo IX: Código Python empleado en la definición de hiper-parámetros del modelo de árboles extremadamente aleatorios (ExtraTrees) 125](#_Toc42012045)

[Anexo X: Código Python empleado en la definición de hiper-parámetros del modelo de bosque aleatorio (Random Forest) 127](#_Toc42012046)

[Anexo XI: Código Python empleado en la validación del modelo de árboles extremadamente aleatorios (ExtraTrees) obtenido. 129](#_Toc42012047)

[Anexo XII: Código Python empleado en la validación del modelo de bosque aleatorio (RandomForest) obtenido. 131](#_Toc42012048)

**ÍNDICE DE ILUSTRACIONES:**

[Ilustración 3.3.1. Variación de la población por coronas en el periodo 2004-2014. Fuente: Encuesta Sintética de Movilidad en la Comunidad de Madrid ESM14 (diciembre 2014). Tabla nº7, p25. 13](#_Toc42012049)

[Ilustración 3.3.2 Motivos de no uso del transporte público (ESM14). Fuente: Gráfico nº21 de la Encuesta Sintética de Movilidad en la Comunidad de Madrid. 2014. 14](#_Toc42012050)

[Ilustración 3.3. Porcentaje de frecuencia de uso de la bicicleta pública en Madrid. Elaboración propia a partir de los datos del estudio del Observatorio de la Bicicleta Pública en España (2018) realizado por Alberto Castro y Esther Anaya. 16](#_Toc42012051)

[Ilustración 3.4 Porcentaje de razón principal (respuesta única) de uso de la bicicleta pública entre usuarios en Madrid. Elaboración propia a partir de los datos de la encuesta del Observatorio de la Bicicleta Pública en España realizado por Alberto Castro y Esther Anaya. 16](#_Toc42012052)

[Ilustración 3.5. Distribución de los viajes por motivo prioritario. Encuesta Sintética de Movilidad de 2014. Comunidad de Madrid. (N=4.899 encuestados) 17](#_Toc42012053)

[Ilustración 3.6. Estación 145 ubicada en Calle Ortega y Gasset 87. Vista general de la estación (izquierda). Bicicleta disponible para su uso con detalle de control de asistencia al pedaleo en manillar, lector de tarjeta y señal lumínica en verde (centro). Detalle de base de estacionamiento con señal lumínica en rojo indicando que se encuentra disponible para depositar una bicicleta (derecha). Elaboración por terceros con cesión de derechos. 18](#_Toc42012054)

[Ilustración 4.1. Esquema de los pasos que constituyen el proceso KDD. Fuente: extraído de The KDD Process for Extracting Useful Knowledge from Volumes of Data [26]. 20](#_Toc42012055)

[Ilustración 4.2. Cadena DIKW que representa la jerarquía del conocimiento según J.Rowley. Fuente: Elaboración propia. 21](#_Toc42012056)

[Ilustración 4.3. Tráfico global de volumen de datos en comunicaciones máquina a máquina. Elaboración propia a partir de los datos disponibles en el pronóstico Cisco VNI Global IP Traffic Forecast 2017-2022 [29] 21](#_Toc42012057)

[*Ilustración 4.4. Captura extracto de conjunto de datos (entidad routes del GTFS de la Empresa Malagueña de Transportes) en formato CSV. Fuente: Portal de Datos Abiertos del Ayuntamiento de Málaga [30]*. 22](#_Toc42012058)

[*Ilustración 4.5. Captura extracto de conjunto de datos (ocupación de aparcamientos públicos en la ciudad de Málaga) en formato JSON. Fuente: Portal de Datos Abiertos del Ayuntamiento de Málaga [30].* 22](#_Toc42012059)

[Ilustración 4.6. Diagrama de secuencia de procesos para el análisis de datos en movimiento (mientras se generan). Fuente: elaboración propia a partir de notas de Cisco Networking Academy [31]. 24](#_Toc42012060)

[Ilustración 4.7. Diagrama de secuencia de procesos para el análisis de datos almacenados o estáticos. Fuente: elaboración propia a partir de notas de Cisco Networking Academy [31] 24](#_Toc42012061)

[Ilustración 4.8. Diagrama del ciclo de vida del proceso de minería de datos según el estándar CRISP-DM. Recurso elaborado por K. Jensen a partir de la guía IBM SPSS Modeler CRISP-DM. 25](#_Toc42012062)

[Ilustración 4.9. Representaciones gráficas del sub-ajuste (A), ajuste adecuado (B) y sobre-ajuste (C) de una frontera entre datos agrupados. Fuente: Recurso de Prof. Kichun Lee [33] 25](#_Toc42012063)

[Ilustración 4.10. Esquema de partición de un conjunto de datos en dos subconjuntos para la validación cruzada. Recurso de Prof. Kichun Lee [32] 26](#_Toc42012064)

[Ilustración 4.11. Esquema explicativo de validación cruzada en k-iteraciones de los modelos A, B y C a través de los indicadores 1, 2 y 3. Fuente: elaboración propia. 27](#_Toc42012065)

[Ilustración 4.12. Diagrama del ejemplo del resorte (izquierda) y representación gráfica del movimiento simulado de la partícula roja en el plano grabado por la Cámara A (derecha). Recurso original de J. Shlens [36] 28](#_Toc42012066)

[Ilustración 4.13. Diagrama explicativo de diferencias entre modelos de aprendizaje supervisados y de aprendizaje no supervisado, así como una posible integración de los mismos para obtener una variable objetivo Y (regresión o clasificación) a partir de atributos contenidos en X. Fuente: elaboración propia. 29](#_Toc42012067)

[Ilustración 5.1. Captura de texto plano contenido en archivo ‘Datos de uso de Mayo 2019’ en formato JSON. 31](#_Toc42012068)

[Ilustración 5.2. Imágenes referidas a la estación 145 «Ortega y Gasset 87». A la izquierda fotografía del tótem de la estación donde se indica que es la estación número 145. A la derecha captura del estado de dicha estación (8 de noviembre de 2018 a las 12:28 horas) en formato JSON. 33](#_Toc42012069)

[Ilustración 5.3.Esquema del proceso de adquisición de datos llevado a cabo empleando diferentes fuentes. Fuente: elaboración propia. 34](#_Toc42012070)

[Ilustración 5.4. Captura de pantalla de consulta semiautomática de valores climatológicos en la API de AEMET. 35](#_Toc42012071)

[Ilustración 5.5. Captura de pantalla de resultado de la consulta (izquierda) y de extracto de datos provistos por la consulta en formato JSON (derecha). 35](#_Toc42012072)

[Ilustración 5.6. Captura de extracto del código en Python empleado en las consultas a OpenrouteService. Fuente: elaboración propia a partir de ejemplos expuestos en Openroute Service [59]. 36](#_Toc42012073)

[Ilustración 5.7. Captura de archivo CSV confeccionado para la contención de datos resultantes de consulta a la API de OpenrouteService. Fuente: elaboración propia. 36](#_Toc42012074)

[Ilustración 5.8. Captura de extracto del código de transformación de JSON a CSV y supresión del campo tracks. Fuente: elaboración propia. 38](#_Toc42012075)

[Ilustración 5.9. Captura de pantalla de la visualización, con Pandas, de los diez primeros registros de los datos de uso de BiciMAD. Fuente: elaboración propia. 39](#_Toc42012076)

[Ilustración 5.10. Captura registros de valores meteorológicos. Fuente: elaboración propia a partir de los datos obtenidos de la Agencia Estatal de Meteorología. 40](#_Toc42012077)

[Ilustración 5.11. Extracto del conjunto de datos de Openroute Service con campos definidos referentes a distancia y duración prevista. Fuente: Elaboración propia a partir de los datos consultados a Openroute Service. 40](#_Toc42012078)

[Ilustración 5.12. Captura de extracto del código asocia datos provistos por Openroute Service con los registros de datos de uso de BiciMAD. Fuente: elaboración propia. 41](#_Toc42012079)

[Ilustración 5.13. Captura de pantalla de la visualización, con Pandas, de los diez primeros registros de los datos de uso tras la incorporación de datos de Openroute Service. Fuente: elaboración propia. 41](#_Toc42012080)

[Ilustración 5.14. Captura de pantalla de extracto del código de conversión del campo ‘unplug\_hourTime’ de formato str a datetime. 42](#_Toc42012081)

[Ilustración 5.15. Captura de extracto de visualización de diez registros del conjunto de datos tras la creación de columnas específicas para año, mes, día, día de la semana y hora. Fuente: elaboración propia. 42](#_Toc42012082)

[Ilustración 5.16. Captura de pantalla de extracto de visualización del conjunto de datos resultante de la unión de los datos adquiridos. 43](#_Toc42012083)

[Ilustración 5.17. Captura de subsanación anomalía en el registro 807759, el cual muestra valores en campos que no corresponden (izquierda) y tras su corrección (derecha). Fuente: elaboración propia. 43](#_Toc42012084)

[Ilustración 6.1. Captura de resultado de la ejecución de la función .describe() de Pandas donde se muestran estadísticos de las variables numéricas del conjunto de datos objeto de estudio. Fuente: elaboración propia. 45](#_Toc42012085)

[Ilustración 6.2. Diagrama de barras de desplazamientos registrados por rangos de edad (‘ageRange’), basado en una muestra aleatoria del conjunto compuesta por 100.000 registros. Fuente: elaboración propia. 45](#_Toc42012086)

[Ilustración 6.3. Diagrama de caja y bigote de las variables travel\_time (izquierda) y duration (derecha). Fuente: elaboración propia. 46](#_Toc42012087)

[Ilustración 6.4. Vista ampliada de diagramas de caja y bigote de las variables travel\_time (izquierda) y duration (derecha). Fuente: elaboración propia. 47](#_Toc42012088)

[Ilustración 6.5. Nube de puntos relativa a correspondencia entre tiempo predicho y empleado por cada desplazamiento, a la cual se superpone una recta de pendiente 1. Vista ampliada del gráfico basado en una muestra n=10.000 registros. Fuente: elaboración propia. Fuente: elaboración propia. 47](#_Toc42012089)

[Ilustración 6.6. Histograma de variable ‘Travel\_time’ basado en muestra aleatoria n=100.000 registros. Fuente: elaboración propia. 48](#_Toc42012090)

[Ilustración 6.7. Histograma de valores de variable ‘Travel\_time’ menores a 60 segundos. Fuente: elaboración propia. 48](#_Toc42012091)

[Ilustración 6.8. Diagrama de barras de frecuencia de uso del servicio BiciMAD por meses basado en una muestra de n= 100.000 registros pertenecientes al periodo junio 2017 – junio 2019. Fuente: elaboración propia. 49](#_Toc42012092)

[Ilustración 6.9. Diagrama de barras de frecuencia de uso del servicio BiciMAD por días de la semana basado en una muestra de n= 100.000. Fuente: elaboración propia. 49](#_Toc42012093)

[Ilustración 6.10. Temperatura máxima (rojo) y mínimas (azul) registradas diariamente en la estación meteorológica del Parque del Retiro, Madrid. Fuente: elaboración propia a partir de datos de AEMET. 50](#_Toc42012094)

[Ilustración 6.11. Gráficos correspondientes al número de desplazamientos registrados por día y hora (arriba) y al número de desplazamientos por día en el servicio BiciMAD entre abril de 2017 y junio de 2019. Fuente: elaboración propia a partir de datos de EMT Madrid. 51](#_Toc42012095)

[Ilustración 6.12. Gráfico de duración mediana de los desplazamientos en el servicio BiciMAD por días en el periodo abril 2017 a junio de 2019 (desechados valores relativos al 25, 29 y 30 de mayo de 2018 para facilitar la visualización de la tendencia). Fuente: elaboración propia a partir de los datos de EMT Madrid. 52](#_Toc42012096)

[Ilustración 6.13. Flujo neto (diferencia entre desplazamientos iniciados y finalizados) en la estación 1 (Puerta del Sol) de Bicimad por día y hora en el periodo octubre de 2017 - diciembre 2018. Fuente: elaboración propia a partir de los datos de EMT Madrid. 52](#_Toc42012097)

[Ilustración 7.1. Captura de pantalla del resultado de ‘df5.meteo’ donde se aprecian valores nulos en registros de temperatura, precipitaciones, dirección y velocidad del viento y de presión atmosférica. Fuente: elaboración propia. 54](#_Toc42012098)

[Ilustración 7.2. Capturas de pantalla de código correspondiente a las imputaciones de valores nulos en los registros 141 y 154 de variables de temperatura (arriba), así como la salida de la instrucción ‘meteo.info()’ (abajo) tras la ejecución del código anterior . Fuente: elaboración propia. 55](#_Toc42012099)

[Ilustración 7.3. Capturas de pantalla del código correspondiente imputación de valores nulos de variable ‘meteo\_presMax’ mediante regresión lineal (arriba) y de la salida de la instrucción ‘meteo.info()’ (abajo) tras la ejecución del código anterior. Fuente: elaboración propia. 56](#_Toc42012100)

[Ilustración 7.4. Capturas de pantalla del código correspondiente imputación de valores nulos de variable ‘meteo\_presMin’ mediante regresión lineal (arriba) y de la salida de la instrucción ‘meteo.info()’ (abajo) tras la ejecución del código anterior. Fuente: elaboración propia. 56](#_Toc42012101)

[Ilustración 7.5. Captura de pantalla del código de imputación de valores de presión mínimas para aquellos que resulten, tras la regresión lineal, mayores a la presión máxima. Fuente: elaboración propia. 57](#_Toc42012102)

[Ilustración 7.6. Capturas de pantalla de relación de variables y cantidad de valores nulos que contienen antes de asociar el conjunto de registros meteorológicos sin valores nulos (izquierda) y después de ello (derecha). Fuente: elaboración propia. 58](#_Toc42012103)

[Ilustración 7.7. Esquema explicativo sobre la segregación del conjunto de datos de uso del servicio por días en variable objetivo (Y) y variables de entrenamiento (X). Fuente: elaboración propia. 59](#_Toc42012104)

[Ilustración 7.8. Capturas de pantalla que muestran los diez primeros registros del subconjunto de entrenamiento X antes (arriba) y después (abajo) del proceso de normalización. 59](#_Toc42012105)

[Ilustración 7.9. Captura de pantalla de la matriz de covarianza, acompañada de leyenda en el lateral izquierdo en relación a la variable a la que corresponde el dígito numérico (de 0 a 12) dispuestos tanto en filas como en columnas. Fuente: elaboración propia. 60](#_Toc42012106)

[Ilustración 7.10. Gráfico de peso o importancias de las diferentes variables del conjunto de datos sobre las dos principales componentes PC1 y PC2. Fuente: elaboración propia inspirado en código del usuario «seralouk» en el repositorio Stack Overflow.[73]. 61](#_Toc42012107)

[Ilustración 7.11. Tabla de coeficientes que definen los 12 componentes principales (columnas) como combinación lineal de las variables del conjunto de datos (filas). Fuente: elaboración propia. 61](#_Toc42012108)

[Ilustración 7.12. Gráfico que representa la proporción de varianza explicada por cada componente principal así como el valor acumulado por la misma y anteriores. Fuente: elaboración propia a partir de líneas de código (modificadas) de L. Lui [74]. 62](#_Toc42012109)

[Ilustración 7.13. Captura de pantalla de matriz de cambio de base W. Fuente: elaboración propia. 63](#_Toc42012110)

[Ilustración 7.14. Representación gráfica de valores predichos por el modelo de regresión lineal ‘LinearRegression()’ de Scikit Learn frente a valores reales de la variable ‘\_id’. Fuente: elaboración propia. 65](#_Toc42012111)

[Ilustración 7.15. Captura de pantalla del resultado del proceso iterativo de evaluación inicial de diferentes modelos ante el conjunto de datos sin reducción dimensional previa. Fuente: elaboración propia a partir de líneas de código inspiradas en la documentación disponible en scikit-learn.org [86]. 66](#_Toc42012112)

[Ilustración 7.16. Histogramas de valores de error absoluto registrados en los diferentes modelos. Fuente: elaboración propia. 67](#_Toc42012113)

[Ilustración 7.17. Captura de pantalla del resultado del proceso iterativo de evaluación inicial de diferentes modelos ante el conjunto de datos tras aplicar reducción dimensional empleando las nueve componentes principales. Fuente: elaboración propia a partir de líneas de código inspiradas en la documentación disponible en Scikit-learn.org [86] 68](#_Toc42012114)

[Ilustración 7.18. Histogramas de valores de error absoluto registrados en los diferentes modelos tras la aplicación de PCA a través del uso de las nueve componentes principales. Fuente: elaboración propia. 68](#_Toc42012115)

[Ilustración 7.19. Captura de pantalla del resultado de la función ‘.describe()’ que muestra la cuenta, media, desviación estándar, mínimo, máximo, así como primer, segundo y tercer cuartil de los valores predichos por cada modelo así como los valores reales de la variable objetivo en el subconjunto de los registros evaluados. Fuente: elaboración propia. 69](#_Toc42012116)

[Ilustración 7.20. Captura de pantalla del resultado de la función ‘.describe()’ que muestra la cuenta, media, desviación estándar, mínimo, máximo así como primer, segundo y tercer cuartil de los valores predichos por cada modelo así como los valores reales de la variable objetivo en el subconjunto de los registros evaluados. Fuente: elaboración propia. 69](#_Toc42012117)

[Ilustración 7.21. Ejemplo de árbol de clasificación aplicado en el conjunto de datos flor iris, empleando funciones de la librería Skearn. Fuente: scikit-learn.org [90]. 70](#_Toc42012118)

[Ilustración 7.22. Ejemplo de árbol de regresión aplicado a un conjunto de datos de precios de la vivienda en Boston [91]. Fuente: recurso obtenido en explained.ai [92]. 72](#_Toc42012119)

[Ilustración 7.23. Captura de pantalla del código donde se establecen los hiper-parámetros sobre los cuales se establece la búsqueda secuenciar con la función GridSearchCV() para el modelo de árboles extremadamente aleatorios. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96] 75](#_Toc42012120)

[Ilustración 7.24. Captura de pantalla que muestra los resultados de la búsqueda secuencial mostrada en la Ilustración 7.23. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96]. 77](#_Toc42012121)

[Ilustración 7.25. Captura de pantalla del código donde se establecen los hiper-parámetros sobre los cuales se realiza la búsqueda secuencial con la función GridSearchCV() para el modelo de bosque aleatorio. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96] 78](#_Toc42012122)

[Ilustración 7.26. Captura de pantalla que muestra los resultados de la búsqueda secuencial mostrada en la Imagen 6.25. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96] 78](#_Toc42012123)

[Ilustración 7.27. Diagrama del proceso de validación cruzada y subconjuntos de registros empleados durante las fases de definición de parámetros y validación de modelos. Fuente: elaboración propia. 79](#_Toc42012124)

[Ilustración 7.28. Representación gráfica de valores predichos por el modelo de árboles extremadamente aleatorios, tras la definición de hiper-parámetros, frente a los reales del conjunto ensayado. Fuente: elaboración propia. 80](#_Toc42012125)

[Ilustración 7.29. Distribución de registros de error absoluto obtenido por el modelo de árboles extremadamente aleatorios, tras la definición de hiper-parámetros, en su representación gráfica en forma de histograma (arriba) y cuantificada (abajo). Fuente: elaboración propia. 81](#_Toc42012126)

[Ilustración 7.30. Representación gráfica de valores predichos por el modelo de bosque aleatorio, tras la definición de hiper-parámetros, frente a los reales del conjunto ensayado. Fuente: elaboración propia 82](#_Toc42012127)

[Ilustración 7.31. Distribución de registros de error absoluto obtenido por el modelo de árboles extremadamente aleatorios, tras la definición de hiper-parámetros, en su representación gráfica en forma de histograma (arriba) y cuantificada (abajo). Fuente: elaboración propia. 82](#_Toc42012128)

[Ilustración 7.32. Comparativa de distribuciones de errores absolutos obtenidos por los modelos de árboles extremadamente aleatorio (izquierda) y bosque aleatorio (derecha). Fuente: elaboración propia. 83](#_Toc42012129)

[Ilustración 7.33. Diagrama explicativo de precisión y exactitud. Fuente: recurso propio de Óscar Moreno (INTEF). 84](#_Toc42012130)

[Ilustración 7.34. Captura de pantalla de líneas de códigos correspondientes al almacenamiento de un modelo una vez ha sido entrenado (parte superior), carga del mismo para su uso posterior (centro) y empleo del mismo para predecir unos valores dada una entrada ‘X’ (parte inferior). Fuente: elaboración propia a partir de la documentación disponible en Scikit-learn.org [86]. 84](#_Toc42012131)

**ÍNDICE DE TABLAS:**

[Tabla 3.1. Variación de la población 14-80 años de la Comunidad de Madrid desde 1996 a 2014 (ESM14). Fuente: Encuesta Sintética de Movilidad en la Comunidad de Madrid ESM14 (diciembre 2014). Tabla nº7, p25 13](#_Toc42012132)

[Tabla 3.2 Tarifas del servicio BiciMAD en diciembre de 2019. Fuente: BiciMAD (EMT Madrid). [25] 18](#_Toc42012133)

[Tabla 4.1. Indicadores considerados en la Imagen 4.7 para los modelos A,B y C. Elaboración propia. 27](#_Toc42012134)

[Tabla 7.1. Tabla de proporción de varianza explicada por el número de componentes principales usadas. Fuente: elaboración propia. 62](#_Toc42012135)

[Tabla 7.2. Hiper-parámetros definidos en la documentación de SciKit Learn para los modelos de bosque aleatorio (izquierda) y árboles extremadamente aleatorios (derecha). Fuente: elaboración propia a partir de la información disponible en la documentación de SciKit Learn[94] [95]. 73](#_Toc42012136)

[Tabla 7.3. Comparación de indicadores significativos de la calidad de los modelos validados. De izquierda a derecha se representa el coeficiente de determinación (R^2), raíz de error cuadrático medio (RECM) y distribución del error absoluto obtenido (medio, primer segundo y tercer cuartil y desviación estándar). Fuente: Elaboración propia. 83](#_Toc42012137)

[Tabla 7.4. Análisis de fechas en las que el modelo ha registrado una raíz de error cuadrático superior a los 3.000 trayectos (en el subconjunto de validación), acompañado de posibles casusas y efectos que han podido afectar a la demanda del servicio. Fuente: elaboración propia. 86](#_Toc42012138)

[Tabla 7.5. Fecha y número de desplazamientos de los registros contenidos en el conjunto de datos con menos de 2.000 registros diarios. Fuente: elaboración propia. 86](#_Toc42012139)

**ÍNDICE DE ECUACIONES:**

[Ecuación 7.1. Expresión empleada por ‘StandardScaler()’ dentro de la librería Scikit Learn. Fuente: elaboración propia a partir de recurso propio de Scikit Learn. 59](#_Toc42012140)

[Ecuación 7.2. Función que evalúa el componente principal PC1 obtenido. Fuente: elaboración propia. 61](#_Toc42012141)

[Ecuación 7.3. Producto matricial correspondiente al cambio de base constituido por las componentes principales empleadas. Fuente: elaboración propia. 63](#_Toc42012142)

[Ecuación 7.4. Definición de error absoluto como la diferencia entre el valor predicho por el modelo y el valor realmente registrado. Fuente: The Data Science Design Manual [87]. 67](#_Toc42012143)

[Ecuación 7.5. Expresión que define el coeficiente de determinación . 83](#_Toc42012144)

# Objetivos y alcance

El presente Trabajo Fin de Grado tiene como objetivo la aplicación de técnicas de minerías de datos con fin de analizar registros de datos de uso de servicio, así como de obtener información de utilidad para la operación del servicio.

A su vez, en el contexto de aplicación probado, se desea estudiar si el uso del servicio se ve afectado por factores climatológicos cuantificables como temperatura, viento o precipitaciones registradas.

Para ello, se articulan secciones con los siguientes contenidos:

1. Definición de objetivos y alcance del trabajo.
2. Introducción y motivación de este.
3. Servicio BiciMAD y su funcionamiento.
4. Técnicas de minería de datos.
5. Explicación, adquisición y procesamiento inicial de los datos.
6. Análisis de los datos obtenidos, desde un punto de vista descriptivo.
7. Definición de líneas de trabajo posibles y documentación de la metodología empleada para la explotación de una de ellas.
8. Conclusiones obtenidas
9. Referencias bibliográficas empleadas.
10. Documentación anexada correspondiente a las líneas de código, en lenguaje Python, empleadas para el desarrollo del trabajo.

Se considera dentro del alcance del presente Trabajo Fin de Grado:

* Introducción y contextualización del objeto de estudio.
* Adquisición, preparación y procesamiento de datos.
* Aplicación de metodología, herramientas y técnicas propias de la minería de datos.
* Justificación de metodología empleada.
* Obtención de un modelo predictivo de la demanda del servicio BiciMAD.
* Definición de líneas de trabajo y mejora complementarias a la desarrollada.

Queda fuera del alcance del presente Trabajo Fin de Grado:

* Justificación y desarrollo matemático de las herramientas empleadas.
* Optimización del rendimiento de algoritmos usados a la arquitectura computacional disponible.
* Análisis y cálculo de costes asociados a la ejecución del presente estudio.
* Estudio de viabilidad y de recursos necesarios en una implantación de los procesos empleados.
* Integración del modelo resultante en un entorno de trabajo compatible con los procesos de operaciones del servicio.

# Introducción

La revolución producida en el marco de las tecnologías de la información y la comunicación (TIC), han transformado en los últimos años el mundo tal y como lo conocíamos. Aunque los cambios más visibles sean percibidos en el ámbito doméstico y repercutan principalmente en la forma de establecer relaciones interpersonales y hábitos de consumo, existen variaciones aún más significativas en el ámbito empresarial.

Las organizaciones, confluyentes en un mercado cada vez más concurrido y globalizado, han obtenido y basado ventajas competitivas en la gestión de los sistemas de información (SI). A su vez, los importantes avances tecnológicos que se producen hacen que cada vez más organizaciones apuesten e integren muchos dichos sistemas en sus procesos principales. Esta integración, conocida como transformación digital, ofrece grandes oportunidades para una mayor y mejor recolección y transformación de datos, así como explotación de la información resultante a través del conocimiento obtenido de todo ello.

En este sentido, resulta de gran utilidad el uso del conjunto de herramientas empleadas en la disciplina denominada «Minería de Datos». Consiste en la búsqueda de patrones, relaciones de asociación y agrupamientos - entre otros – entre grandes conjuntos de datos almacenados en bases u obtenidos en tiempo real. Con ello se posibilita la construcción de modelos predictivos y de aprendizaje con infinidad de usos en multitud de ámbitos. En algunos casos estos modelos están habilitados para la percepción del entorno y toma de decisiones, dando lugar a la inteligencia artificial. Sirven como ejemplos:

-En el ámbito sanitario, a partir de parámetros biomédicos, se ha logrado un diagnóstico preciso, rápido y económico, de enfermedades como el Alzheimer [1], habiéndose diagnosticado una década después casos considerados como falsos positivos.

- En el comercio, en grandes superficies, comercio electrónico o plataformas digitales, las aplicaciones de reglas de asociación y árboles de decisión son cada vez más frecuentes. Descuentos en base a compras recientes y perfil del cliente, previsión de demanda y sugerencias de productos y servicios, pueden encontrarse en casos como Carrefour [2], Amazon y Netflix [3].

- En la gestión de operaciones y del mantenimiento resulta de suma utilidad el gran volumen de datos generados. En el sector de la aviación pueden ser mencionados cómo modelos de redes neuronales se emplean para el cálculo y optimización de operaciones del espacio aéreo europeo [4]. Igualmente, aeronaves comerciales de última generación incorporan sistemas de recopilación de datos en sus propios sistemas. Por ejemplo el Airbus A350 XWB, que en 24 horas operando obtiene alrededor de 2,5 terabytes (se prevé triplicar esta cifra en próximas versiones) de información procedente de más de 6.000 sensores. Esta información sirve como punto de partida en la creación de modelos predictivos para detección de anomalías, gestión inteligente del combustible, programación del mantenimiento preventivo o anticipación de fallos que pudieran resultar fatales [5].

- En el reconocimiento de imágenes, audio y voz, la minería de datos y la inteligencia artificial actúan conjuntamente en la identificación de figuras, rostros, símbolos alfanuméricos [6], así como el reconocimiento de voz y piezas musicales [7].

De otro lado, una mayor conciencia social sobre los efectos de las emisiones contaminantes y polución del aire, la congestión de las redes de transporte en grandes ciudades, y la aparición de nuevos medios y formas de transporte, están transformando sustancialmente el modelo de movilidad, especialmente en el ámbito urbano. Resultan relevantes los papeles que juegan las redes de transporte público frente al privado, los medios con propulsión eléctrica, híbrida o autopropulsados, así como modelos de vehículos compartidos o *sharing*.

En este contexto, nace en 2014 el servicio BiciMAD [8], un sistema de alquiler público de bicicletas eléctricas de la que es titular el Ayuntamiento de Madrid. Cuenta con más de 2.000 bicicletas eléctricas, alrededor de 62.500 usuarios registrados y 165 estaciones repartidas a lo largo de la ciudad [9] [10].

La disponibilidad de los registros de operación de BiciMAD, bajo la política de datos abiertos, ha servido como punto de partida en la realización del presente Trabajo Fin de Grado. Consiste en el registro de los trayectos realizados entre abril de 2017 y junio de 2019 en dicho sistema, lo que resulta en un volumen de datos equivalente a unos 12 gigabytes de información en formato *JSON*.

El objeto de este Trabajo Fin de Grado es la explotación de dicho registro de datos, tras su preparación y aplicación sobre el mismo de técnicas de minería de datos, la obtención de información de utilidad para la planificación y el control de sus operaciones, así como el conocimiento de factores que pudieran resultar condicionantes.

# BiciMAD

Los flujos migratorios experimentados, especialmente durante el Siglo XX, de entornos rurales hacia las ciudades han contribuido en gran medida a la formación de extensas áreas metropolitanas. La concentración de grandes conjuntos de personas alrededor de un mismo núcleo urbano, así como la densidad de población con las que suelen formarse, plantean retos antes no vistos para satisfacer necesidades individuales y colectivas, tales como la planificación de servicios como el transporte y las vías de comunicación. La deslocalización y concentración de la población en grandes conjuntos residenciales, frecuentemente distantes de los centros de estancia durante gran parte del día - lugar de trabajo, estudios o desempeño de otras actividades-, supone un reto logístico la realización de millones de desplazamientos demandados diariamente con el menor tiempo y coste, así como con la mayor comodidad posible para el usuario.

En el caso de la Comunidad de Madrid, en la Tabla 3.1 se muestra el dato de población, por «Coronas» y su evolución a lo largo de las distintas Encuestas Domiciliadas de Movilidad en 1996, 2004 y 2014. Se aprecia evolución de crecimiento pronunciado en las Coronas más externas y un estancamiento, o ligera contracción, en las coronas más centrales (Ilustración 3.3.1).

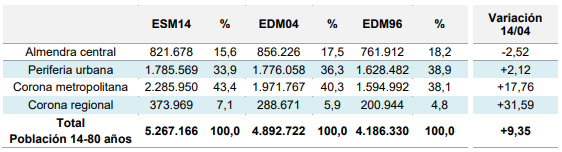


Tabla 3.1. Variación de la población 14-80 años de la Comunidad de Madrid desde 1996 a 2014 (ESM14). Fuente: Encuesta Sintética de Movilidad en la Comunidad de Madrid ESM14 (diciembre 2014). Tabla nº7, p25

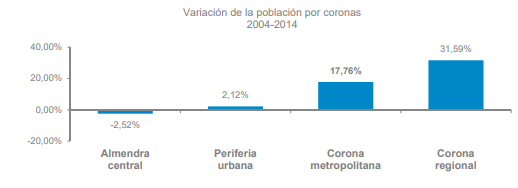


Ilustración 3.3.1. Variación de la población por coronas en el periodo 2004-2014. Fuente: Encuesta Sintética de Movilidad en la Comunidad de Madrid ESM14 (diciembre 2014). Tabla nº7, p25.

De otro lado la calidad del aire, en términos medioambientales, y la alta contribución de las actividades de transporte a su empobrecimiento - responsable del 41% de las emisiones de gases invernadero en Madrid para el año 2015 según el informe CyPT Madrid [11] – no hace más que aumentar el grado de complejidad que requiere la planificación de infraestructuras y servicios de transporte en grandes áreas metropolitanas.

En el caso de Madrid, una oferta amplia y de diversa modalidad en los transportes públicos tratan de hacer frente al transporte privado – responsable de alrededor del 85% de las emisiones de gases de efecto invernadero en Madrid durante el 2015 según el ya citado informe CyPT Madrid. Pese a estar cada vez más desincentivado, en 2014 alrededor del 49% de los desplazamientos mecanizados en Madrid se realizaron en transporte privado, frente al 46,22% en transporte público [12].

Según esta misma fuente, los principales motivos que disuadieron a los encuestados a optar por el transporte público son los mostrados en la Ilustración 3.3.2. Razones principales como ‘Cercanía/prefiero andar/hacer ejercicio’, ‘Ahorro de tiempo’, ‘Mala combinación con Transporte Público’ o ‘No hay servicio’ pueden ser paliados con la alternativa que es objeto de estudio en el presente Trabajo Fin de Grado. BiciMAD, y los sistemas de bicicleta compartida en general, son presentados como modos de trasporte económicos, saludables y respetuosos con el medio ambiente. Con capacidad de servir trayectos punta a punta – siempre condicionado a la disponibilidad y proximidad de estaciones, así como disponibilidad en ellas de bicicletas o puntos de enganche, y servicio disponible las 24 horas – resulta una alternativa válida en desplazamientos cortos y medios cortos para determinados grupos de población. El transporte compartido, cuyos servicios ofertados van más allá de la bicicleta (motocicletas en el caso de *Muving* [13]o vehículos turismos en el de *Car2Go* [14]), es una modalidad de transporte emergente que ofrece flexibilidad y autonomía propias del transporte privado, sin estar el usuario a expensas de horarios y servicios planificados –aunque si a su disponibilidad-, y más económico que un vehículo particular. El pago por uso no requiere inversión inicial y el precio incluye combustible, seguros de responsabilidad civil, mantenimiento y amortización del vehículo.

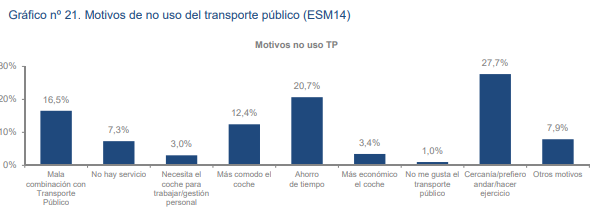


Ilustración 3.3.2 Motivos de no uso del transporte público (ESM14). Fuente: Gráfico nº21 de la Encuesta Sintética de Movilidad en la Comunidad de Madrid. 2014.

Los primeros servicios de bicicleta compartida aparecen en las décadas de los sesenta y setenta – Amsterdam (1964) o La Rochelle (1974)- aunque su desarrollo parece impedido hasta la implantación de tecnologías como la electrónica o la geolocalización en el sistema que permite disuadir o controlar el uso fraudulento, vandalismo y robo de las bicicletas. Se plantean como modos alternativos de movilidad para desplazamientos cortos, de hasta 6-7 kilómetros dentro del núcleo urbano, o en combinación intermodal junto a otros sistemas de transporte colectivos [15]. Dichos sistemas presentan beneficios como una generación de CO2 por viajero y kilómetro, unas 130 veces menos que la de un vehículo particular [16], menor tasa de ocupación de suelo respecto el vehículo particular o reducción de la contaminación acústica respecto a otros medios convencionales.

BiciMAD se crea en 2014 por el Ayuntamiento de Madrid, presentándose con la finalidad de «proporcionar un elemento alternativo de transporte limpio y saludable al ciudadano y fomento el uso de la bicicleta en la ciudad». Inicialmente ofertado mediante la fórmula de concesión a la empresa privada *BonoPark* [17], el servicio de BiciMAD fue municipalizado en octubre de 2016, cuya gestión depende de la Empresa Municipal de Transportes (EMT) de Madrid hasta la fecha. Actualmente cuenta con una infraestructura de 165 estaciones, con 4116 bases de estacionamiento, y una flota compuesta por 2028 bicicletas eléctricas [9].

Además de la disponibilidad de asistencia eléctrica al pedaleo en el 100% de su flota, destacan también como ventajas de BiciMAD la disponibilidad de una amplia red de vías ciclistas en el núcleo urbano de Madrid y la integración de los sistemas y tecnologías de información y comunicación - mejora la experiencia del usuario a través de una visibilidad a tiempo real de la disponibilidad de servicios en varias plataformas. Estas tecnologías de información, junto a la gestión pública del servicio, han posibilitado la disponibilidad de registros de operación que han inspirado y hecho posible la realización del presente Trabajo Fin de Grado. Dichos registros se encuentran disponibles, como datos abiertos en formato JSON, en la sección «*OPEN DATA*» de la sede electrónica de la Empresa Municipal de Transportes (EMT) de Madrid [18].

### 3.1 Aceptación del servicio y tarifas

Respecto al uso y aceptación del sistema, una muestra de 234 de los alrededor de 65.000 usuarios abonados al sistema [19] – también se permite uso de viajes ocasionales – han sido objeto de estudio por parte del Observatorio de la Bicicleta Pública en España [20]. A partir de dicho estudio se obtienen los gráficos representados en Ilustración 3.3 e Ilustración 3.4. En relación a la frecuencia de uso de la bicicleta pública en Madrid se presenta la Ilustración 3.3, donde destaca el uso ocasional y esporádico por la mayor parte de los encuestados – solo un 26% de los encuestados afirman emplear la bicicleta más de 3 veces a la semana.



Ilustración 3.3. Porcentaje de frecuencia de uso de la bicicleta pública en Madrid. Elaboración propia a partir de los datos del estudio del Observatorio de la Bicicleta Pública en España (2018) realizado por Alberto Castro y Esther Anaya.

Acerca de la naturaleza de los desplazamientos para los que se emplea el servicio, se presenta la Ilustración 3.4, donde predomina el uso para ir o volver a un lugar de ocio (41%), seguido de ir o volver del trabajo (29%). Estos resultados contrastan con los presentados, correspondientes a la motivación de los desplazamientos, en general, según la Encuesta Sintética de Movilidad de 2014 (ESM14) [12] en la Comunidad de Madrid, disponibles en la Ilustración 3.5. De esto puede deducirse cierta afinidad de los usuarios hacia el servicio de bicicleta compartida para satisfacer desplazamientos motivados, por ejemplo, por ocio frente al trabajo. A su vez evidencia que, de considerarse la muestra del estudio como significativa de la población de usuarios de BiciMAD, esta no es representativa de la población general del área metropolitana de Madrid (estudiada en la ESM14), siendo susceptible de estudio el perfil del usuario del servicio BiciMAD.



Ilustración 3.4 Porcentaje de razón principal (respuesta única) de uso de la bicicleta pública entre usuarios en Madrid. Elaboración propia a partir de los datos de la encuesta del Observatorio de la Bicicleta Pública en España realizado por Alberto Castro y Esther Anaya.

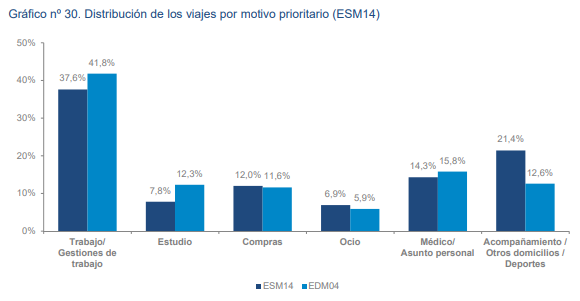


Ilustración 3.5. Distribución de los viajes por motivo prioritario. Encuesta Sintética de Movilidad de 2014. Comunidad de Madrid. (N=4.899 encuestados)

El precio de uso del servicio puede considerarse uno de los factores más influyentes en la forma de uso del servicio de bicicleta compartida. A diferencia de la mayoría de sistemas de bicicletas compartidas del país, que ofrecen servicio de tarifa plana para usos desplazamiento inferiores a 30 minutos (Bicing [21], Sevici [22], Bizi [23] o Málagabici [24]) – aunque ninguno de ellos para bicicletas con ayuda eléctrica que caracteriza BiciMAD-. Con ello se pretende evitar uso para desplazamientos muy cortos y la transferencia masiva de usuarios del transporte público [15].

Las tarifas de BiciMAD, en sus dos modalidades (con abono anual o con tarjeta ocasional) se muestran en la Tabla 3.2.



Tabla 3.2 Tarifas del servicio BiciMAD en diciembre de 2019. Fuente: BiciMAD (EMT Madrid). [25]

Tal y como se aprecia en la Tabla 3.2, existen bonificaciones de 0,10€ si se retira la bicicleta en una estación excedentaria, anclaje en una deficitaria o reserva de punto de estacionamiento en el destino (las dos últimas bonificaciones no son acumulables).

### 3.2 Cómo usarlo

Los usuarios del sistema interactúan por un lado con elementos tangibles, como las bicicletas y estaciones (y sus respectivas bases) de estacionamiento, e intangibles como los sistemas de información a través del cual se gestionan abonos, reservas y pagos.



Ilustración 3.6. Estación 145 ubicada en Calle Ortega y Gasset 87. Vista general de la estación (izquierda). Bicicleta disponible para su uso con detalle de control de asistencia al pedaleo en manillar, lector de tarjeta y señal lumínica en verde (centro). Detalle de base de estacionamiento con señal lumínica en rojo indicando que se encuentra disponible para depositar una bicicleta (derecha). Elaboración por terceros con cesión de derechos.

Para realizar un desplazamiento el usuario, abonado o con una tarjeta ocasional, se dirige a una estación cercana donde se encuentre, al menos, una bicicleta lista para su uso (Ilustración 3.6, izquierda). Con el fin de conocer las estaciones más cercanas, así como la disponibilidad de bicicletas en el origen y de puntos de anclaje en el destino, el usuario puede acceder a la información del servicio en la página web, la aplicación móvil o en la consola de cualquier estación de la red BiciMAD. De esta forma también podrá conocer las estaciones excedentarias y las deficitarias que permiten obtener bonificaciones en ese momento. Una vez allí, el usuario puede desbloquear una bicicleta presentando su tarjeta directamente en el lector dedicado a ello en cada base de estacionamiento (Ilustración 3.6, centro). Si hay saldo suficiente y tras desbloquearse el anclaje, la bicicleta queda liberada y puede comenzarse el desplazamiento. El usuario tiene libertad para hacer el recorrido que quiera hasta estacionar de nuevo la bicicleta en cualquier estación a su paso – incluyendo la de origen – o allí donde ha sido reservado un punto de anclaje al comienzo de la reserva (Ilustración 3.6, derecha). Una vez anclada de nuevo la bicicleta, se da por concluido el trayecto, aplicándose las normas de tarificación visibles en la Tabla 3.2. Si la bicicleta es devuelta en la misma estación donde se retiró antes de transcurrir 5 minutos, el sistema entenderá que ha sido devuelta por problemas técnicos y no cobrará importe alguno.

Los puntos de estacionamiento cuentan con una señal lumínica de colores que indican su disponibilidad a través de los colores, siendo verde si está disponible, rojo si está listo para un enganche o con una bicicleta no disponible, y azul si es una base de anclaje reservada. Si no hay ninguna luz encendida la base de estacionamiento no se encuentra disponible.

# Minería de datos

La Minería de Datos consiste en el proceso de extracción de información de valor, generalmente oculta a simple vista, a partir de grandes conjuntos de valor a través de técnicas, herramientas y algoritmos concretos. Este proceso se considera a su vez parte del proceso denominado Descubrimiento de Conocimiento en Bases de Datos [26]- «*Knowledge Discovery Database*» o KDD por sus siglas en inglés-. A la Minería de Datos preceden otros subprocesos del KDD como la propia adquisición de datos, selección y preparación de los mismos, así como otros procesos que le siguen como la interpretación y explotación de los mismos.

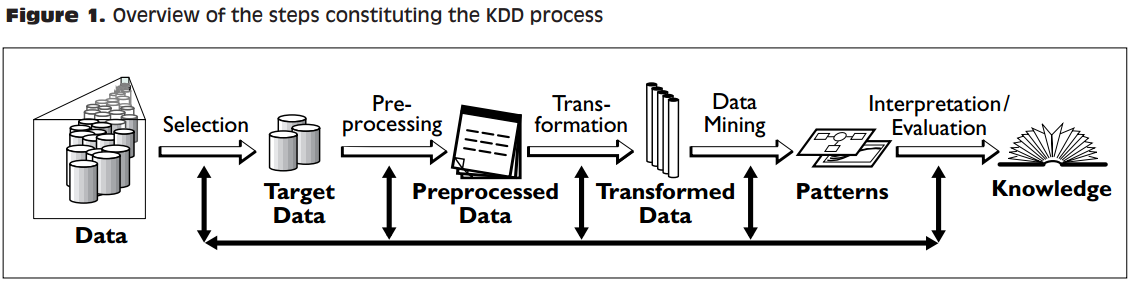


Ilustración 4.1. Esquema de los pasos que constituyen el proceso KDD. Fuente: extraído de The KDD Process for Extracting Useful Knowledge from Volumes of Data [26].

Las revoluciones tecnológicas experimentadas en las últimas décadas han contribuido a la generación masiva de datos y su almacenamiento. El creciente número de sensores y registros no solamente permiten un mayor control y conocimiento del desempeño producido o que se está realizando tiempo real en un sistema – destacan los sistemas de planificación de recursos empresariales (ERP por sus siglas en inglés) y la interconexión de objetos a través del «Internet de las Cosas» – sino que se abre un amplio abanico de posibilidades de obtención de conocimientos, patrones y asociaciones que permiten establecer modelos predictivos realmente útiles. Algunos casos se citan en la introducción del presente documento. La integración de estas nuevas herramientas de análisis de datos, junto a otras tendencias como la computación en la nube, está al frente del proceso de digitalización de los procesos industriales y de la denominada cuarta revolución industrial.

Es importante reseñar la diferencia existente entre la minería de datos y el uso masivo de datos (popularmente conocido por su nombre en inglés «*Big Data*») siendo este último la disponibilidad de grandes volúmenes de datos que pueden ser explotados para obtener información que a su vez se puede convertir en conocimiento. La estructura de este proceso, denominada «jerarquía del conocimiento» y representada a través de la pirámide o de la cadena *DIKW* [27], representada en la Ilustración 4.2, diferencia etapas desde que los datos en bruto son adquiridos, hasta que resultan en un valor añadido o diferencial obtenido a partir de los mismos – denominado por la autora como «sabiduría». La adquisición de información, conocimiento y sabiduría se corresponde a su vez con la denominada «Inteligencia de Negocios» o *BI* por sus siglas en inglés («*Business Intelligence*»).



Ilustración 4.2. Cadena DIKW que representa la jerarquía del conocimiento según J.Rowley. Fuente: Elaboración propia.

Respecto al carácter masivo de los datos, su generación y transferencia está experimentando un crecimiento exponencial. En este año 2020 se espera que el tráfico de Internet global sea 95 veces el volumen de todo el tráfico de Internet registrado durante 2005 – según señala la empresa estadounidense Cisco en su pronóstico VNI sobre el crecimiento de datos entre 2015 y 2020 [28]. Una parte significativa del volumen de tráfico global de internet se debe a conexiones entra máquinas («*Machine to machine*» o *M2M* por sus siglas en inglés) cuyo crecimiento previsto que – según el pronóstico VNI de Cisco para el periodo 2017-2022 – se produce de forma exponencial, tal y como puede apreciarse en la Ilustración 4.3.

Un ejemplo de datos de esta naturaleza (*M2M*) es el conjunto de datos objeto de estudio en el presente Trabajo Fin de Grado. Sensores propios de las bicicletas, como el geolocalizador, y de las estaciones, como los sensores de presencia de bicicletas o los lectores de tarjeta, recogen registros y transmiten a tiempo real – a través de controladores y dispositivos IP - con otros dispositivos, que forman parte del sistema de información de BiciMAD, de forma automática sin precisar para ello la mediación de ninguna persona en el transcurso normal de las operaciones. El detalle del conjunto de datos objeto de estudio será abordado más delante en el capítulo «Análisis de datos disponibles».



Ilustración 4.3. Tráfico global de volumen de datos en comunicaciones máquina a máquina. Elaboración propia a partir de los datos disponibles en el pronóstico Cisco VNI Global IP Traffic Forecast 2017-2022 [29]

En el estudio de los datos masivos se conoce como las “V” o “uves” a diferentes aspectos cuantitativos y cualitativos que caracterizan a un conjunto de datos. Aunque no existe unanimidad en el número de V que se han de considerar – algunos textos indican tres, cinco, siete, ocho, nueve o diez “uves”- pueden se citan las siguientes:

1. Volumen: número y extensión de registros disponibles.
2. Velocidad: en relación con el tiempo que tardan los registros en ser generados y procesados.
3. Variedad: orígenes o fuentes de los datos, su heterogeneidad, formatos y unidades de medida empleados, así como el carácter estructurado, semiestructurado o no estructurado del conjunto.
4. Valor: utilidad de la información que se obtiene a partir de dichos.
5. Variabilidad: la concentración o dispersión de los valores de cada variable.
6. Veracidad: mediciones confiables, fieles a la magnitudes y unidades consideradas, con la precisión, escala y sensibilidad adecuadas. Sin presencia de ruido.
7. Volatilidad: tiempo que el conjunto de datos es conservado. Pueden ser almacenados indefinidamente, solamente durante un periodo o desechados tras su procesamiento.

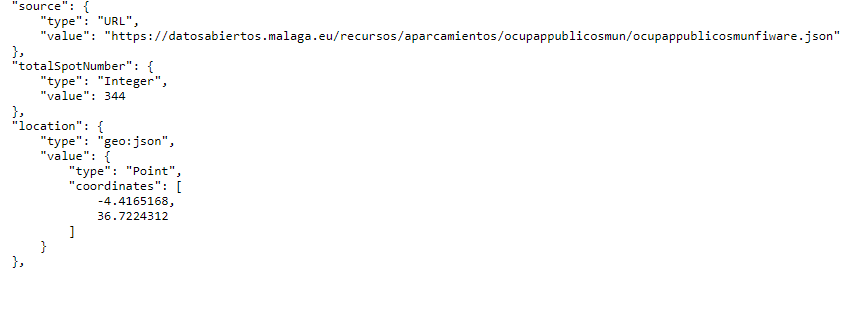
En cuanto a la ya citada estructuración de los datos, en base a ella se puede encontrar conjuntos de datos:

-Estructurados: estructura definida y homogénea, generalmente por campos. P.ej.: un archivo en formato CSV (conjunto ordenado de datos separados por comas), el cual contiene valores en campos definidos por filas y columnas. (Ilustración 4.4)



*Ilustración 4.4. Captura extracto de conjunto de datos (entidad routes del GTFS de la Empresa Malagueña de Transportes) en formato CSV. Fuente: Portal de Datos Abiertos del Ayuntamiento de Málaga [30]*.

-Semiestructurados: estructura definida no necesariamente homogénea: P.ej.: un archivo en formato JSON, donde los formatos son presentados precedidos de la declaración del campo al que se refieren. (Ilustración 4.5).



*Ilustración 4.5. Captura extracto de conjunto de datos (ocupación de aparcamientos públicos en la ciudad de Málaga) en formato JSON. Fuente: Portal de Datos Abiertos del Ayuntamiento de Málaga [30].*

-No estructurados: sin estructura definida. P.ej.: un documento manuscrito.

Otro factor que sin duda ha contribuido al desarrollo y aplicación de procesos minería de datos, es el aumento de la capacidad computacional en equipos convencionales, así como la creciente oferta de servicios de computación bajo el modelo en «la nube» – la carga computacional se ejecuta en grandes centros de cálculos donde se concentran máquinas virtualizadas que optimizan su rendimiento y aplican tarifas de pago por uso – y en «la niebla» - similar a en «la nube» pero más próximo al entorno de generación datos, beneficiando a la velocidad de respuesta frente a la capacidad de computación. Ello ha posibilitado la aplicación y desarrollo de nuevas técnicas y algoritmos, capaces de obtener más información procesada en menor tiempo y respondiendo a los crecientes volúmenes de datos que se generan. También ha supuesto un avance cualitativo materializado en el desarrollo del aprendizaje automatizado (más conocido como «*Machine Learning»*) – sistemas dinámicos, retroalimentados por sus propios resultados, cuyo desempeño mejora con la experiencia – y la Inteligencia Artificial – que además incluye la percepción del entorno, la toma de decisiones y el aprendizaje de forma autónoma.

En definitiva, como principal objeto del presente Trabajo Fin de Grado se presenta el ensayo de un modelo predictivo, a través de herramientas de minería de datos, que pueda resultar de utilidad para la gestión de las operaciones del sistema BiciMAD. El modelo predictivo – cuyo desarrollo se detalla en las siguientes secciones – pretende el conocimiento profundo del comportamiento de los usuarios en el sistema – a través de búsqueda de patrones, agrupaciones o asociaciones difícilmente apreciables sin procesos de minería de datos – para anticiparse a ellos. Por ejemplo, conociendo el pronóstico meteorológico de la ciudad de Madrid para los próximos 7 días, un modelo podría predecir, con mayor o menor acierto, la demanda de servicios por zonas y poder adaptar a ella la planificación de las operaciones (traslados y reposición de bicicletas entre estaciones, reparaciones y mantenimiento, bonificaciones adicionales que se minimicen el riesgo de colapso de estaciones, etc). El conocimiento derivado de la explotación de estos datos resultar de gran interés para multitud de entidades, entre ellas:

- El propio gestor del servicio, para la optimización de sus recursos, mejora de la calidad y disponibilidad del mismo y reducción de costes.

- Entidades públicas, o privadas, interesadas en conocer las preferencias de movilidad de los ciudadanos, así como las variables que afectan a las mismas.

- Entidades privadas que estén interesadas en realizar estudios de mercado que puedan resultar sustitutivos o relacionados con el objeto de estudio.

- Potenciales gestores del servicio interesados en conocer las variables que condicionan las operaciones, y los gastos incurridos e ingresos obtenidos, para implantación de un sistema similar o para la concurrencia en un proceso de licitación.

En cualquiera de los casos anteriores, son procesos sustancialmente diferentes aquellos que pretenden una lectura y respuesta inmediata, a tiempo real, de aquellos que pretenden obtener conocimiento partiendo de una experiencia previa en forma de datos almacenados.

En el análisis de datos a tiempo real, la secuencia de procesos se desarrolla como se muestra en la Ilustración 4.6.



Ilustración 4.6. Diagrama de secuencia de procesos para el análisis de datos en movimiento (mientras se generan). Fuente: elaboración propia a partir de notas de Cisco Networking Academy [31].

Por el contrario, aquellos que parten de experiencias previas siguen la secuencia mostrada en la Ilustración 4.7.



Ilustración 4.7. Diagrama de secuencia de procesos para el análisis de datos almacenados o estáticos. Fuente: elaboración propia a partir de notas de Cisco Networking Academy [31]

El principal motivo por el cual se altera la secuencia de subprocesos reside en la obsolescencia de los datos considerados (relacionado con la ya mencionada volatilidad). Si el interés proceso reside en reaccionar a tiempo ante un comportamiento anómalo, será necesario llevar a cabo las acciones correspondientes con un tiempo de respuesta aceptable.

Para la implantación de procesos de análisis y minería de datos, el estándar abierto «*CRISP-DM*» [32] sirve como hoja de ruta en la definición de modelos de explotación en el ámbito industrial. Esta especificación – cuyo uso es el más extendido para este tipo de procesos – define seis grandes integradas en un proceso reiterativo, tal y como se representa en Ilustración 4.8:

* Comprensión del negocio: necesidad de conocer el entorno de operación, así como una correcta definición de los objetivos y las necesidades de la organización. A su vez deben traducirse a términos compatibles con el análisis y minería de datos, a través de los cuales se podrá definir el modelo matemático al cual el modelo debe atender.
* Comprensión de los datos a emplear: investigación y exploración de los datos disponibles, así como la identificación de información contenida y posibles hipótesis a probar en relación con ella.
* Preparación de datos: todos los procesos que implica la transformación de los datos en bruto hasta constituir un conjunto de datos apto para su uso.
* Modelado: preselección de técnicas a ensayar y evaluar, selección de una o varias y optimización de su rendimiento con el ajuste de hiper-parámetros. Puede implicar volver a la fase anterior para continuar con el ensayo y definición del modelo.
* Evaluación del modelo: cuantificación del rendimiento y desempeño mostrado por el modelo desarrollado – correspondiente al proceso de validación en la presente memoria.
* Implementación: puesta en funcionamiento del modelo, monitorización de los procesos, acciones de mantenimiento correspondientes y gestión de la información producida.

Las diferentes fases pueden sucederse en bucle, pudiendo iterarse la alternación entre varias de ellas hasta obtener el resultado deseado, o integrarse en procesos de mejora continua.

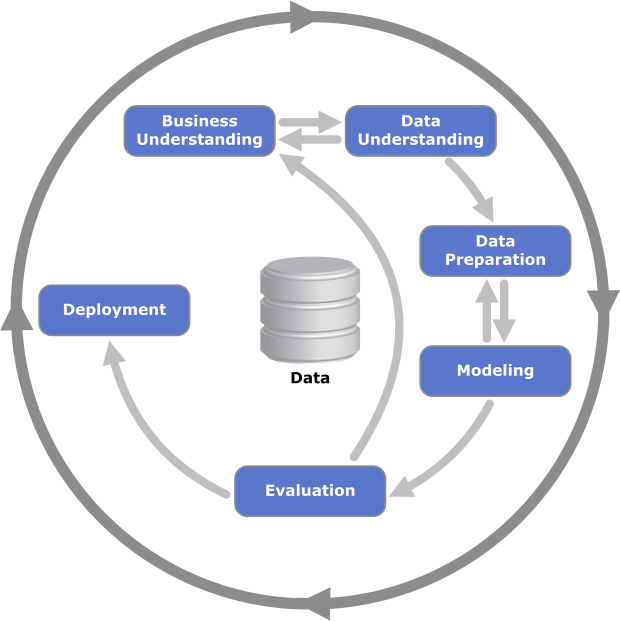
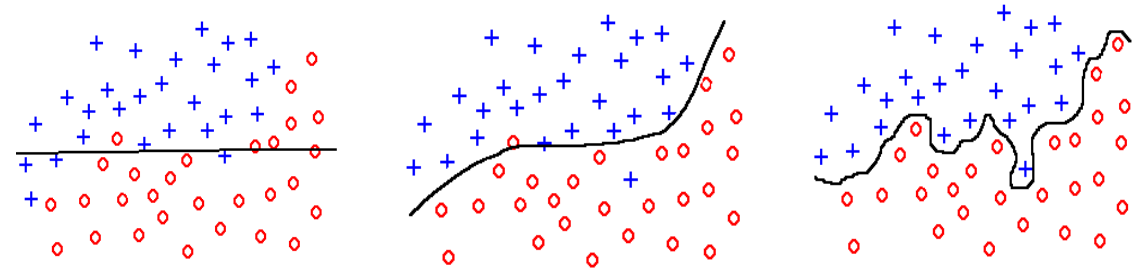


Ilustración 4.8. Diagrama del ciclo de vida del proceso de minería de datos según el estándar CRISP-DM. Recurso elaborado por K. Jensen a partir de la guía IBM SPSS Modeler CRISP-DM.

Por otro lado, una de las mayores amenazas que puede sufrir cualquier modelo predictivo es el sobreajuste (Ilustración 4.9). Aunque el punto de partida sea un conjunto de datos con un gran número de registros, es importante la aplicación de algunas técnicas y metodologías durante el desarrollo del modelo. De otra manera se corre el riesgo de obtener un modelo que describa de forma precisa el conjunto de registros considerados, pero pierda validez a la hora de predecir respuestas del sistema objeto de estudio.



(B)

(C)

(A)

Ilustración 4.9. Representaciones gráficas del sub-ajuste (A), ajuste adecuado (B) y sobre-ajuste (C) de una frontera entre datos agrupados. Fuente: Recurso de Prof. Kichun Lee [33]

Una estrategia adecuada para la definición de parámetros que proporcionen un ajuste conveniente, es el de partición de datos y, en concreto, la técnica de validación cruzada [34]. Ello consiste en la segregación de los datos en dos subconjuntos, uno llamado de entrenamiento y otro de evaluación (Ilustración 4.10). Los datos comprendidos en el subconjunto de entrenamiento son utilizados para el desarrollo de modelos de prueba, cuyo rendimiento es comprobado en el de validación. De esta forma se mide el desempeño del modelo – el cual puede predecir el valor de una de las variables en base al valor de otras – empleando registros sobre los que no se ha entrenado.

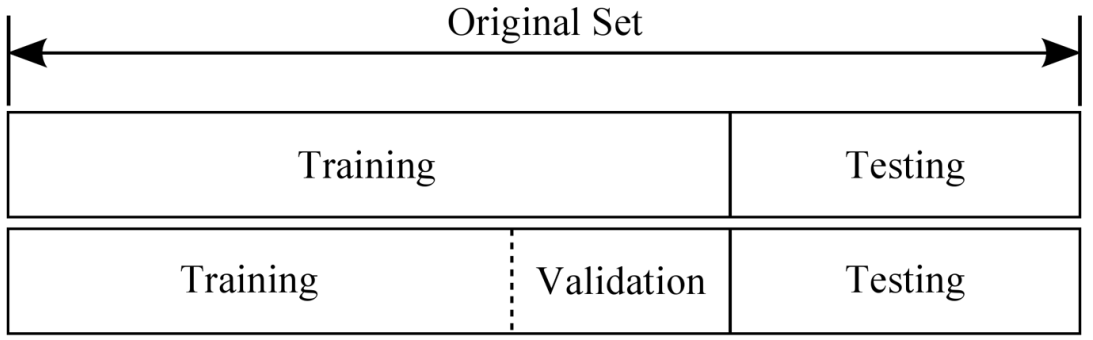


Ilustración 4.10. Esquema de partición de un conjunto de datos en dos subconjuntos para la validación cruzada. Recurso de Prof. Kichun Lee [32]

Lo habitual es ensayar modelos predictivos de distinta naturaleza – diferentes algoritmos, distintas selecciones de componentes a considerar – y posteriormente evaluar cuál de ellos muestra un mejor comportamiento predictivo, de acuerdo con el resultado que se espera de él. Sin embargo, si tanto el entrenamiento de los diferentes modelos, como la validación del mismo se realizan sobre los mismos subconjuntos, puede resultar igualmente en un sobreajuste del modelo seleccionado. Lo mismo sucede si el desarrollo del modelo y la definición de sus parámetros se realiza empleando el mismo subconjunto. Por ello, se emplea una tercera fracción de validación en la Ilustración 4.10.

Una técnica extendida para evitar el sobreajuste en el desarrollo de varios modelos es el conocido como «validación cruzada de K iteraciones» [35] (o por su término en inglés «*K-fold cross validation*»). Esta técnica consiste en la preparación ‘K’ subconjuntos y en la realización de K-1 iteraciones de validación cruzada –alternando en cada una de ellas el subconjunto empleado para la validación y entrenando los distintos modelos en los subconjuntos restantes. La validez de cada modelo puede ser cuantificada a través de distintos indicadores de desempeño – coeficiente de determinación, distribución de errores absolutos, error absoluto medio, raíz del error cuadrático medio (RECM), porcentaje de acierto o porcentaje de falsos positivos, y de falsos negativos (matriz de confusión) entre otros y según cada caso – resultante del promedio, para cada modelo e indicador, de los valores obtenidos en las diferentes iteraciones.

En la Ilustración 4.11 se representa un ejemplo en el cual se ensayan tres modelos diferentes - A, B y C – cuya aptitud es cuantificada a través de los indicadores 1, 2 y 3. En la aplicación de validación cruzada con K-interacciones, el conjunto de datos – representado en azul – es dividido en subconjuntos que se alternan en las diferentes iteraciones para el entrenamiento (verde) y la evaluación (rojo) de los diferentes modelos.



Ilustración 4.11. Esquema explicativo de validación cruzada en k-iteraciones de los modelos A, B y C a través de los indicadores 1, 2 y 3. Fuente: elaboración propia.

Para decidir qué modelo es más conveniente se podrán hacer uso de los indicadores 1, 2 y 3 – generalmente algunos de los estadísticos muestrales anteriormente citados.

En este caso, los valores empleados podrían obtenerse de las expresiones contenidas en la Tabla 4.1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Indicador 1 | Indicador 2 | Indicador 3 |
| Modelo A |  |  |  |
| Modelo B |  |  |  |
| Modelo C |  |  |  |

Tabla 4.1. Indicadores considerados en la Imagen 4.7 para los modelos A,B y C. Elaboración propia.

Como se ha mencionado anteriormente, el modelo predictivo pretende conocer – o aproximar- el valor de una variable en base al valor de otras. Resulta evidente plantear que no necesariamente todas las variables realizan la misma contribución a la determinación del valor de la variable objetivo, sino que unas resultan más significativas que otras, incluso resultar redundantes y dependientes entre ellas.

Por ejemplo, suponiendo un modelo predictivo que establezca el valor de mercado de un vehículo de segunda mano, variables como el año de matriculación, la marca o la cilindrada del motor pueden resultar más significativas que otras como el número de puertas o el último dígito numérico de la matrícula.

Sin embargo, no siempre las variables que más contribuyen a determinar el valor de la variable a predecir son aquellas que muestren mayor correlación en el conjunto de datos de partida, sino que se hace necesario el Análisis de Componentes Principales (ACP) – «*Principal Component Analysis*» (PCA) en inglés. Este análisis, aplicable a complejos conjuntos de datos, permite considerar una reducción dimensional que evidencie dinámicas simplificadas generalmente ocultas. Este método, basado en el álgebra lineal, permite realizar un cambio de base sobre las variables del conjunto, el cual establece un nuevo orden de componentes que minimizan la redundancia recíproca entra ellas [36].

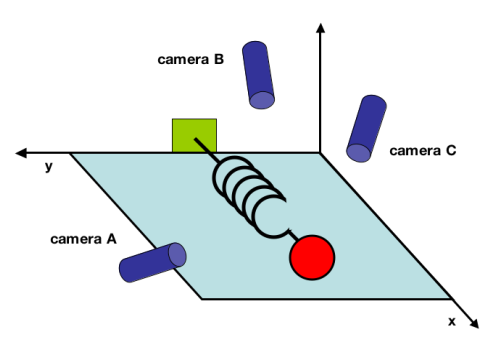
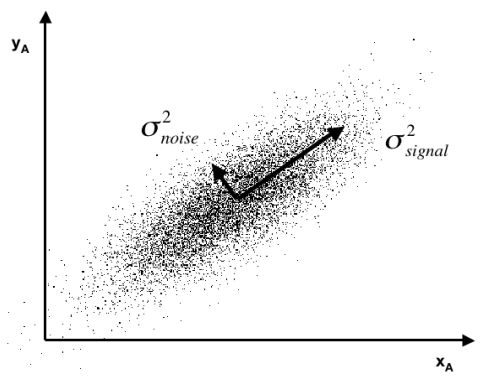


Ilustración 4.12. Diagrama del ejemplo del resorte (izquierda) y representación gráfica del movimiento simulado de la partícula roja en el plano grabado por la Cámara A (derecha). Recurso original de J. Shlens [36]

J. Shlens muestra un ejemplo que facilita visualizar el ACP en su publicación «*A tutorial on principal component analysis*» [36]. El autor habla del movimiento de un sistema ideal compuesto de una partícula adherida al final de un resorte. El movimiento de la partícula es registrado durante un periodo concreto por las cámaras A, B y C según se muestra en el diagrama de la Ilustración 4.12. Considerando su disposición oblicua frente al movimiento del resorte, la representación gráfica del movimiento en el plano captado por la cámara -compuesto por los ejes e - evidencia una composición bidimensional del movimiento. Sin embargo, según se aprecia en el diagrama, aplicando un cambio de base, considerando por ejemplo la proyección en el plano XY (sombreado en azul), es posible proyectar el movimiento de la partícula considerando el eje X el componente principal de su movimiento, sobre el que se produce la oscilación. Por tanto, la dinámica de un movimiento registrado en un espacio tridimensional con posición , puede simplificarse en una expresión del tipo .

El análisis de componentes principales se integra en los denominado modelos de aprendizaje no supervisado. Estos modelos aprenden de la estructura y los datos de un conjunto, contribuyendo al procesamiento e identificación de patrones de comportamiento en etapas posteriores. Sin embargo, de ellos no se obtiene una respuesta explicita que establezca una inferencia, identificación o agrupación. Por el contrario, los modelos de aprendizaje supervisados ofrecen una resolución a este tipo de problemas. Dichos modelos aprenden de registros de los cuales obtiene experiencia – fase denominada «entrenamiento» del modelo. Una vez concluye el desarrollo del modelo, este explicita un valor, categoría o agrupación para cada insumo recibido.

Un ejemplo de posible configuración de modelos de aprendizaje supervisado y no supervisado se resume en el diagrama presentado en la Ilustración 4.13. En una primera fase de entrenamiento – parte superior – se desarrolla un modelo de aprendizaje supervisado a partir de un conjunto de registros destinados a su entrenamiento. Este conjunto contiene valores de la variable objetivo () y atributos asociados (), los cuales son a su vez procesados por un modelo de aprendizaje no supervisado, como pudiera ser el ACP. Como resultado, el modelo no supervisado facilita el procesamiento y aprendizaje por parte del de tipo supervisado. En la parte inferior de la Ilustración 4.13 se representa el modelo, una vez entrenado, en pleno funcionamiento. De nuevo la señal de entrada – de la cual se pretende obtener una estimación de la variable objetivo – es procesada por el modelo de aprendizaje no supervisado. El resultado del mismo es empleado por el modelo supervisado, el cual y con base en ella, ofrece una estimación del valor predicho .



Ilustración 4.13. Diagrama explicativo de diferencias entre modelos de aprendizaje supervisados y de aprendizaje no supervisado, así como una posible integración de los mismos para obtener una variable objetivo Y (regresión o clasificación) a partir de atributos contenidos en X. Fuente: elaboración propia.

Generalmente los modelos de aprendizaje actúan como regresores – estiman un valor numérico a una variable objetivo de este tipo -, clasificadores – estiman la pertenencia del registro a una de las categorías previamente declaradas – o de agrupamientos (o *clustering* en inglés). La principal diferencia de este último con los clasificadores es que parte de su aprendizaje consiste en la identificación de los grupos o *clusters,* que no han sido previamente declarados.

Estos modelos se basan en principios matemáticos, a través de los cuales se realizan las predicciones. Existe una infinidad de herramientas de cálculo multivariable disponibles para su definición – estimadores lineales y no lineales, paramétricos y no paramétricos – entre los cuales se mencionan:

* Regresión lineal múltiple [37].
* Clasificación por métodos de regresión logística [38].
* Regresión bayesiana [39].
* Clasificaciones y regresiones polinómica [40].
* Máquinas de vectores de soporte [41].
* Clasificación y regresión por gradiente estocástico [42].
* Regresión, clasificación y agrupación por el método de los ‘K’ vecinos más próximos [43].
* Arboles de clasificación y regresión [44].

Para el desarrollo de modelos, existe una multitud de herramientas, entornos y programas informáticos que facilitan la ejecución de las técnicas, cálculos, algoritmos y procesos propios de la minería de datos. Algunos de ellas – es el caso de Weka [45] – funcionan a muy alto nivel, como una caja negra de la que se obtienen resultados a partir del conjunto de datos. Otros – como KNIME [46]– permiten diseñar el proceso mediante bloques, los cuales representan subprocesos concretos que también que funcionan como cajas negras. En consecuencia al alto nivel de programación en el que operan, son opciones que presentan posibilidades limitadas frente a otras de más bajo nivel como librerías disponibles para lenguajes de programación como *Java* [47], R [48] y *Python* [49]. El uso de estos lenguajes está ampliamente extendido en la minería de datos y permiten la configuración del proceso a través de su programación línea a línea. Librerías publicadas como *Caret* (en R), *Pandas* [50] o *Scikit Learn* [51](ambas en *Python*) combinan funciones en alto nivel – los cuales incluyen multitud de subprocesos y algoritmos definidos que, de otra forma resultaría muy tedioso ejecutar – con multitud de parámetros configurables para adaptar el proceso a cada caso particular.

Para la realización del presente Trabajo Fin de Grado se ha optado por el uso de Python 3.7, en especial las funciones contenidas en las librería *Pandas*, *Scikit Learn* y *Matplotlib* [52]*,* haciendo uso de Microsoft Visual Studio 2017 [53] como entorno de desarrollo. Puntualmente se han empleado los programas Notepad++[54] y Microsoft Excel 2010 [55] para la visualización y manipulación de determinados conjuntos de datos. Se ha empleado un ordenador personal con un procesador Intel Core i7-3630Q (cuatro núcleos y frecuencia máxima de 3,40 GHz) y 8 GB de memoria RAM.

# Datos empleados

## 5.1 Adquisición

En el presente Trabajo Fin de Grado se han empleado conjuntos de datos obtenidos de dos fuentes principales: conjuntos de datos abiertos y consultas lanzadas a través interfaces de programación de aplicación - más conocidas como API por sus siglas en inglés.

En cuanto a conjunto de datos obtenidos a través de portal de datos abiertos, se corresponden con los datos propios del servicio BiciMAD y a sus operaciones. Estos se encuentran disponibles y publicados en el apartado de datos abiertos de la sede electrónica de la Empresa Municipal de Transportes (EMT) de Madrid [18].En la misma se encuentran hipervínculos para la descarga de diferentes archivos en formato comprimido ZIP, segregados por meses. Se muestran disponibles:

* Datos de uso por meses, desde abril de 2017 hasta junio de 2019 de forma ininterrumpida.
* Datos del estado de las estaciones por día y hora desde julio de 2018 hasta junio de 2019.

En ambos casos, en el interior de los archivos comprimidos se encuentran los datos en formato JSON. Como se ha mencionado anteriormente, este formato contiene información registrando valores precedidos de la declaración del campo al que se refiere – el cual puede contener otros campos y objetos anidados – lo que también dificulta su visualización. Este formato, especialmente apropiado para el intercambio de datos entre sistemas (comunicaciones M2M), supone la presencia de información redundante al declararse de forma reiterada el título de cada campo, tal y como se aprecia en la Ilustración 5.1.

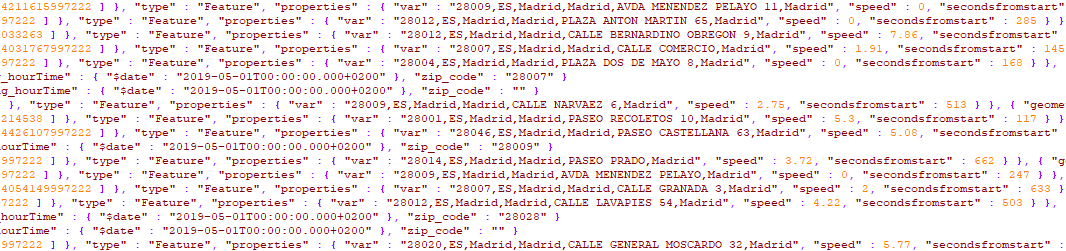


Ilustración 5.1. Captura de texto plano contenido en archivo ‘Datos de uso de Mayo 2019’ en formato JSON.

Respecto a los archivos de datos de uso de BiciMAD, se registran datos en los siguientes campos – según explica la guía informativa [56] que los acompaña en el portal de datos abiertos – para cada uno de los registros, correspondiente con un desplazamiento realizado:

* ‘\_id’: código identificador del trayecto, único para cada registro.
* ‘user\_day-code’: código identificador único para los trayectos realizados por un mismo usuario en un mismo día. Los gestores publican este código con fines estadísticos – intensidad de uso por usuario y día - en lugar de un identificador por usuario para proteger así la privacidad de los usuarios.
* ‘idunplug\_station’: código de identificación de la estación donde se retira la bicicleta.
* ‘idunplug\_base’: código de identificación, en dicha estación, del anclaje del cual se retira la bicicleta.
* ‘idplug\_station’: código de identificación de la estación donde se deposita la bicicleta
* ‘idplug\_base’: código de identificación, en dicha estación, del anclaje en el cual se deposita la bicicleta.
* ‘unplug\_hourTime’: marca temporal – en formato “año-mes-día” seguido de “dia:minuto:segundo+zona horaria” – del momento de desenganche. El registro de marca temporal se muestra agregado por horas, también por motivos de protección de la privacidad de los usuarios.
* ‘travel\_time’: tiempo, en segundos, entre desenganche y enganche de la bicicleta.
* ‘track’: registros relativos al trayecto realizado en formato GeoJSON.
* ‘user\_type’: tipo de usuario. Se registran los siguientes números para cada categoría:

0 - No se ha identificado el tipo de usuario.

1 - Usuario anual (abonado).

2 - Usuario ocasional.

3 - Trabajador de la empresa.

* ‘ageRange’: rango de edad del usuario. Se registran los siguientes números para cada grupo de edad:

0 – No se ha identificado el grupo de edad del usuario.

1 – El usuario tiene menos de 17 años.

2 – El usuario tiene entre 17 y 18 años.

3 – El usuario tiene entre 19 y 26 años.

4 – El usuario tiene entre 27 y 40 años.

5 – El usuario tiene entre 41 y 65 años

6 – El usuario tiene 66 años o más.

* ‘zip\_code’: Código postal asociado al usuario, si es que existe registro.

En total se han obtenido 27 archivos de este tipo que, tras su descompresión y en formato JSON suponen algo más de 14,7 Gigabytes (GB) de información – un promedio de 518,9 Megabytes (MB) por archivo con un mínimo de 77,7 MB (marzo 2018) y un máximo de 1,1 GB (septiembre 2018). En su totalidad engloban 8.822.058 registros únicos de desplazamientos.

En el caso de los datos relativos a situación de estaciones por hora y día, EMT Madrid no presenta ninguna guía explicativa asociada. Sin embargo, el significado de muchos de los campos encontrados, por su encabezado y formato, resultan deducibles. En aquellos campos que han podido suscitar dudas respecto a su significado, se ha hecho uso de la información de contacto ofrecida en el portal de datos abiertos.

En este caso cada registro presenta dos campos en un primer nivel:

* ‘id\_’: identificador de cada registro que corresponde con la marca temporal en la que se han obtenido los datos. Se recogen estado de
* ‘stations’: contiene tantos objetos como estaciones se refieren en el registro. Cada estación muestra a su vez lo siguientes encabezados:
  + ‘activate’: disponibilidad de la estación siendo ‘1’ activa y ‘0’ no activa.
  + ’id’: código de identificación de la estación dentro del sistema de información de BiciMAD.
  + ‘number’: número de estación visible al público. No necesariamente corresponde con el código de identificación que emplea el sistema sino con la señal visible. Por ejemplo, la estación ‘Ortega y Gasset 87’, mostrada en la Ilustración 5.2, es identificada por el usuario como la 145 aunque el código de identificación en el sistema es la 171.
  + ‘name’: denominación de la estación p.ej.: «Puerta del Sol A».
  + ‘longitude’ y ‘latitude’: coordenadas geográficas donde se encuentra ubicada la estación, diferenciada dos campos relativos a la longitud y a la latitud respectivamente.
  + ‘no\_available’: muestra ‘0’ si la estación está en servicio y ‘1’ si se encuentra fuera de servicio.
  + ‘address’: dirección física donde se encuentra ubicada la estación incluyendo via y número.
  + ‘total\_bases’: número total de bases de anclaje de dicha estación.
  + ‘free\_bases’: número total de bases disponibles en dicha estación en ese momento.
  + ‘dock\_bikes’: bicicletas que se encuentran ancladas en ese momento en la estación.
  + ‘light’: grado de ocupación de la estación donde ‘0’ representa baja ocupación, ‘1’ representa ocupación media, ‘2’ alta ocupación y ‘3’ completa.



Ilustración 5.2. Imágenes referidas a la estación 145 «Ortega y Gasset 87». A la izquierda fotografía del tótem de la estación donde se indica que es la estación número 145. A la derecha captura del estado de dicha estación (8 de noviembre de 2018 a las 12:28 horas) en formato JSON.

De forma adicional, los datos de posicionamiento de estaciones de BiciMAD han sido completados con la realización de consultas a EMT Madrid y corrección, realizadas manualmente, en cuyos coordenadas geográficas se ha encontrado una desviación significativa (aquellas con un error mayor a 50 metros o en un vial diferente) realizando mediciones y toma de coordenadas pertinentes con el programa *Google Earth Pro* [57]*.* Tras ello, puede darse por concluido el proceso de adquisición de datos.

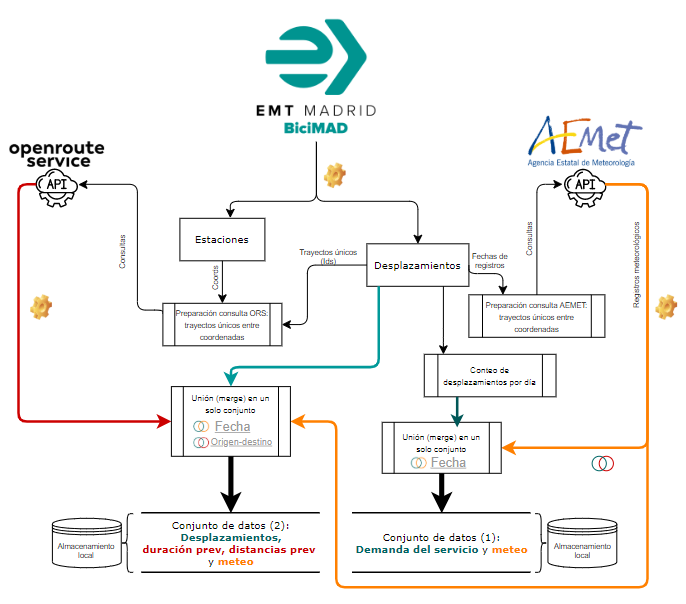


Ilustración 5.3.Esquema del proceso de adquisición de datos llevado a cabo empleando diferentes fuentes. Fuente: elaboración propia.

Tras la descarga, complementación y almacenamiento en local de estos datos se ha concluido el proceso de adquisición de datos propios del servicio de BiciMAD.

En una segunda etapa del proceso de adquisición se han adquirido datos referentes a meteorología así como tiempo y distancia esperados en los desplazamientos. Dichos datos han sido obtenidos de la Agencia Estatal de Meteorología (AEMET) [58] y de Openroute Service [59], respectivamente. En ambos casos a través de consultas a interfaces de programación de aplicaciones (conocidas por sus siglas en inglés API).

En el caso de AEMET, se ha optado por la extracción de datos – valores climatológicos diarios – de forma semiautomática de la aplicación. Para ello se ha solicitado y obtenido una clave de acceso a la aplicación. Con ella se ha podido realizar la consulta correspondiente a los valores registrados, durante el periodo analizado, en la estación meteorológica ubicada en el Parque del Retiro (Ilustración 5.4).

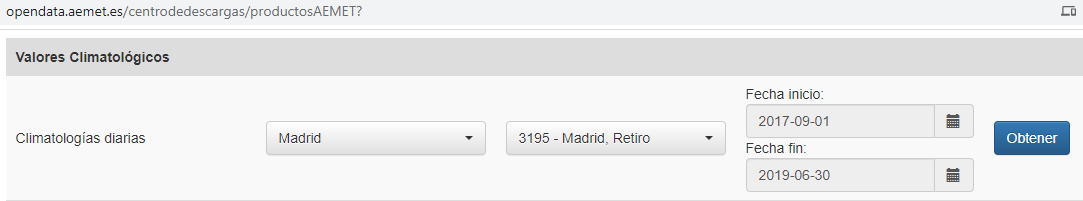


Ilustración 5.4. Captura de pantalla de consulta semiautomática de valores climatológicos en la API de AEMET.

La aplicación proporciona la respuesta mostrada en la Ilustración 5.5 (izquierda), la cual contiene un enlace directo que redirige a los datos requeridos, cuyo extracto se muestra en la Ilustración 5.5 (derecha), en formato JSON.

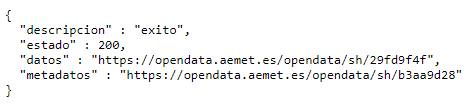
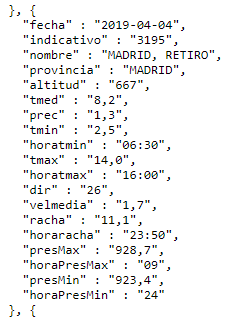


Ilustración 5.5. Captura de pantalla de resultado de la consulta (izquierda) y de extracto de datos provistos por la consulta en formato JSON (derecha).

Los valores consultados solo permiten registros diarios, no así detallado por horas. Se declaran los campos referentes a:

- Fecha

- Estación meteorológica desde la que se tomaron las mediciones, incluyéndose indicativo, nombre, provincia y altitud de la misma.

- Temperaturas media, mínima y máxima registradas en ese día.

- Precipitaciones recogidas en la estación durante ese día.

- Dirección y velocidad media del viento, así como velocidad máxima registrada en rachas.

- Presión atmosférica mínima y máxima registrada en ese día.

- Horas a la que se han registrados los valores máximos y mínimos de temperatura, viento y presión atmosférica.

En cuanto a los datos obtenidos de la aplicación de acceso gratuito *Openroute Service* [59], han sido obtenidos a partir de consultas automáticas realizadas con *Python,* tal y como se muestra en la Ilustración 5.6. En ella se aprecia el código empleado, en el cual se ha construido un bucle iterativo que lanza una consulta – solicita indicaciones de navegación entre cada par de coordenadas geográficas – cada vez que es recorrido. Los datos son incluidos como nuevos registros en el conjunto de datos ‘SalidaAPI’, que posteriormente es convertida y almacenada en un archivo con formato CSV cuya captura se muestra en la Ilustración 5.7.



Ilustración 5.6. Captura de extracto del código en Python empleado en las consultas a OpenrouteService. Fuente: elaboración propia a partir de ejemplos expuestos en Openroute Service [59].



Ilustración 5.7. Captura de archivo CSV confeccionado para la contención de datos resultantes de consulta a la API de OpenrouteService. Fuente: elaboración propia.

Las coordenadas empleadas corresponden a las de origen y destino de cada uno de los trayectos. Han sido obtenidas extrayendo los trayectos de los datos de uso de BiciMAD. De dicho extracto se han eliminado duplicidades y trayectos cuya estación de origen y destino es la misma, resultando un total de 31.341 trayectos únicos (considerando exclusivamente origen y destino). Los códigos ‘*idunplug\_station*’ y ‘*idplug\_station*’ de estos trayectos, cuyo volumen es posible abrir y tratar en Microsoft Excel – herramienta limitada a 1.048.576 filas y 16.384 filas [60]–, han sido sustituidos por las coordenadas geográficas correspondientes, obtenidas a partir de los datos de situación de las estaciones de BiciMAD (una vez acometidas las subsanaciones comentadas anteriormente).

Cabe reseñar las restricciones de uso en *Openroute Service*, el cual está limitado a un máximo de 40 consultas por minuto y un máximo de 2.000 al día, por clave de acceso – única para cada usuario. Dado el volumen de consultas que se precisan durante realización del presente Trabajo Final de Grado, estas limitaciones se han logrado salvar estableciendo tiempo de espera entre consultas - 1,4 segundos dado que – y no realizando más de 2000 consultas por clave y día. Este hecho ha dilatado significativamente la duración del proceso de adquisición de datos. En caso de integrarse este tipo de consultas automáticas de forma habitual, las limitaciones podrían salvarse con otros proveedores de servicios no gratuitos como la plataforma de mapas de *Google Cloud* [61].

Por otro lado, como se aprecia en el código mostrado en la Ilustración 5.7, se hace referencia expresa al modo de transporte como bicicleta eléctrica – «*cycling-electric*» al final de la dirección a la que se dirige la consulta – dado que repercute directamente en el tiempo y distancia del trayecto al habilitarse diferentes vías en base a ello.

El código correspondiente a la adquisición de los datos de *Openroute Service* se muestra disponible en el Anexo I: Código Python de consultas de trayectos a API de Openroute Service.

## 5.2 Preprocesamiento

Los conjuntos de datos mencionados han precisado de un extenso preprocesamiento con el fin de adecuarlos a los análisis que se proponen y desarrollan en el presente Trabajo Fin de Grado.

Una de las mayores dificultades encontradas, durante la manipulación de estos datos, han sido las limitaciones relacionadas con el volumen de los mismos, lo cual en algunos momentos ha comprometido la viabilidad del análisis de dichos registros en un equipo convencional – cabe recordar que solo los datos relativos a uso de BiciMAD suponen 14 GB-. Se han realizado numerosas búsquedas y ensayos iterativos en relación a técnicas y herramientas –no siempre adecuadas – para la gestión masiva de datos de este tipo. Gracias a esto se han logrado estrategias que posibilitan la realización del análisis. Por ejemplo, el uso de partición en porciones de los subconjuntos de datos – distinguible en el código con «*chunk size*» - reduce la memoria física empleada y agiliza el proceso de importación de datos. Este hecho evita errores por memoria insuficiente y reduce sustancialmente el tiempo de ejecución del orden días a menos de pocos minutos.

Continuando con los datos de BiciMAD, se ha procedido a la transformación del formato de datos desde JSON a CSV –facilita su visualización y organización de un valor único por filas y columnas. Igualmente se ha desechado el campo «*tracks*», cuya estructura anidada resulta difícil de contener en dicho CSV – además de constar como cadena de caracteres-. La supresión de este campo – y de la información contenida en él – también ha supuesto la reducción del tamaño del conjunto de los datos de uso de BiciMAD de 14.7 GB a 1.24GB.

Tal y como se muestra en la Ilustración 5.8 – en la cual se aprecia un extracto del código empleado y disponible en el Anexo II: Código Python correspondiente a conversión a CSV, filtrado y unificación de datos estáticos de desplazamientos de BiciMAD

– se establece un bucle iterativo donde, cada vez que es recorrido transforma y filtra uno de los archivos mensuales de datos de uso de BiciMAD. De esta forma, una vez completado el bucle, todos los archivos JSON han sido transformados en otros equivalentes con formato CSV.



Ilustración 5.8. Captura de extracto del código de transformación de JSON a CSV y supresión del campo tracks. Fuente: elaboración propia.

En este caso el intérprete de lectura JSON del paquete *Pandas*, que almacena los datos como una variable de estructura denominada «*Dataframe*» en dicha librería, ha conservado algunos caracteres residuales que se presentan de forma no deseada acompañando a los valores en los campos *‘\_id’* y *‘unplug\_hourTime’*. Al visualizarse se aprecia cómo los caracteres aparecen con la misma estructura en todos los registros, no formando parte de valor de los campos indicados. Por ello, se han incluido funciones de reemplazo – la cuales sustituyen las cadenas de caracteres excedentes por el conjunto vacío – dentro del bucle. De esta forma, tal y como se aprecia en la Imagen 5.7, parte del código incluido en el bucle hace las veces de filtro antes de grabar los registros con formato CSV.

Cabe destacar que, como se aprecia también en la Ilustración 5.8, las funciones de reemplazo explicitan los campos a los que se refieren, pero no así a los registros. Esta posibilidad, que permite realizar operaciones por filas y/o columnas en las estructuras de tipo *Dataframe*, es una de las grandes ventajas de las funciones de *Pandas*. De recurrir a bucles iterativos de tipo *for*, se precisarían tiempos de ejecución del orden de 15 días –según estimaciones propias a partir de ensayos realizados durante el presente Trabajo Fin de Grado y con el equipo empleado– para completar los 8.822.058 registros únicos. Sin embargo, empleando las líneas de código que se muestran en la imagen, el tiempo de ejecución ha sido del orden de minutos.

Una vez completada la transformación a archivos CSV, se ha procedido a su anexión en un único conjunto de datos, supresión de registros duplicados y almacenado en ese mismo formato (CSV). En la Ilustración 5.9 se pueden visualizar los diez primeros registros de dicho conjunto tras cargarlo en una estructura del tipo *Dataframe* en *Pandas*. Se disponen los diez primeros registros (columnas repartidas en dos líneas) precedidos por su índice numérico – del cero al nueve en este caso – y situados los valores bajo el encabezado del campo. En los casos donde no se encuentran registros asociados a un campo de denota con *‘NaN’*.

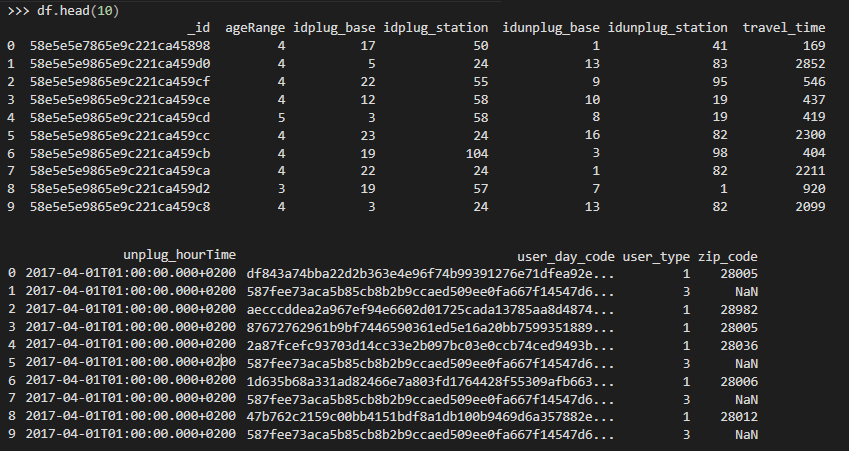


Ilustración 5.9. Captura de pantalla de la visualización, con Pandas, de los diez primeros registros de los datos de uso de BiciMAD. Fuente: elaboración propia.

Paralelamente se han adecuado los conjuntos de datos referentes a registros meteorológicos y de distancias y tiempo para cada trayecto. En ambos casos, al contar con número inferior al millón de registros, se han podido emplear herramientas como Microsoft Excel. En este caso cabe reseñar la especial atención que merece la configuración de formatos de celda – según cada caso pero resulta aconsejable como texto para preservar las cadenas de caracteres – así como los problemas derivados de la interpretación del separador de datos – ‘,’ o ‘;’ son los más frecuentes-. Este último problema es fácilmente subsanable empleando la función de Excel «Texto en columnas».

Una captura de un extracto de su visualización puede apreciarse en la Ilustración 5.10. En pasos posteriores se ha decidido suprimir campos como los referentes a la estación meteorológica – la misma en todos los casos – así como la hora a la que se ha registrado la temperatura máxima, temperatura mínima y hora de rachas de viento. Con ello se pretende evitar un volumen excesivo desechando aquellas variables menos interesantes para su análisis.



Ilustración 5.10. Captura registros de valores meteorológicos. Fuente: elaboración propia a partir de los datos obtenidos de la Agencia Estatal de Meteorología.

En cuanto al conjunto de datos obtenidos a partir de las consultas a la aplicación *Openroute Service*, se han adecuado de forma similar. Partiendo del conjunto de mostrado en la Ilustración 5.7, se ha empleado una combinación de las funciones «reemplazar» y «texto en columnas». Con ello se han obtenido los valores de distancia y duración prevista – según el proveedor de datos – para cada trayecto único tal y como se aprecia en el extracto de la Ilustración 5.11.



Ilustración 5.11. Extracto del conjunto de datos de Openroute Service con campos definidos referentes a distancia y duración prevista. Fuente: Elaboración propia a partir de los datos consultados a Openroute Service.

## 5.3 Unión en un único conjunto de datos

Una vez adecuados los datos, procedentes de diversas fuentes con diferentes formatos, se han aunado en un único conjunto. Para ello, se han aplicado asociaciones relacionales en base a una clave primaria para cada caso (Anexo III: Código Python correspondiente a asociación de datos en un solo conjunto (desplazamientos + distancia y duración + meteo) . Los registros de uso de BiciMAD han servido como conjunto principal, a cuyos registros se han añadido – incorporando nuevos campos – los datos relativos a registros meteorológicos, así como distancia y duración prevista.

Puede resultar lógico que dichas duraciones y distancias previstas se relacionen con los trayectos a partir de cuál es la estación de desenganche y cuál la de enganche – mismo orígenes y destinos. Sin embargo, los registros no cuentan con un identificador de los trayectos que puedan ejercer de clave primaria en este proceso. Para ello se crea un campo – ad hoc – en ambos conjuntos, cuyo valor resulta de la concatenación de ‘*idunplug\_station*’ con ‘-‘ e ‘*idplug\_station*’ en cada registro. Se emplea el carácter ‘-‘ como separar dado que, en otro caso, resultaría el mismo código para diferentes trayectos – por ejemplo ‘1112’ podría interpretarse como de las estaciones 1 a las 112, de la 11 a la 12 o de la 111 a la 2.

Tal y como se aprecia en la Ilustración 5.12, los datos de uso – declarados como ‘df2’ – y de distancia y tiempo previstos – ‘trayectos’ – se han unido empleando la función ‘*merge*’ de *Pandas*. Dicha función asocia ambos registros a través de un campo, presente en sendos conjuntos bajo una misma denominación – ‘id’ en este caso.

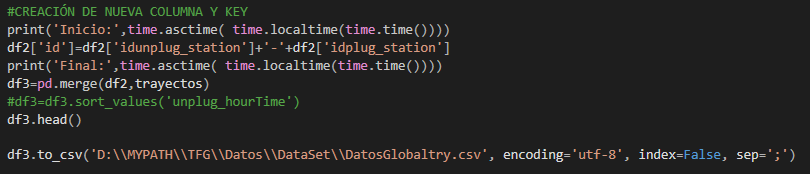


Ilustración 5.12. Captura de extracto del código asocia datos provistos por Openroute Service con los registros de datos de uso de BiciMAD. Fuente: elaboración propia.

El resultado se muestra en la Ilustración 5.12, similar a la Ilustración 5.9, pero donde incluye información relativa a tiempos y distancias.



Ilustración 5.13. Captura de pantalla de la visualización, con Pandas, de los diez primeros registros de los datos de uso tras la incorporación de datos de Openroute Service. Fuente: elaboración propia.

Un proceso similar se ha llevado a cabo para lograr la incorporación de los valores meteorológicos. En esta ocasión la marca temporal sirve de clave primaria, segregando las horas y limitándose a diferenciar los días – AEMET no ofrece registros de este tipo por horas. La gestión de marcas temporales resulta algo más complicada dado que en archivos CSV se almacenan como cadena de caracteres. Por ello, precisan de la conversión a otro formato que permita su interpretación completa – por ejemplo el formato «*datetime*» en *Pandas*. La función ‘*pd.to\_datetime*’ interpreta los caracteres registrados en dicho campo y lo convierte en el formato ‘*datetime’*, variable con atributos tales como año, mes, día, día de la semana, hora, minuto o segundo que registra el dato. Además, permite consultas y aplicación de funciones sobre dichos atributos.



Ilustración 5.14. Captura de pantalla de extracto del código de conversión del campo ‘unplug\_hourTime’ de formato str a datetime.

En este punto, con el fin de agilizar posibles operaciones posteriores – la conversión a *datetime* de este conjunto requiere unos 45 minutos cada vez que es cargado – han sido almacenados diferentes atributos en campos específicos - año, mes, día, día de la semana y hora – tal y como se aprecia en la Ilustración 5.15. De esta forma, algo más rudimentaria, se pueden prevenir problemas potenciales relacionados con la compatibilidad de funciones concretas con el formato *datetime*.

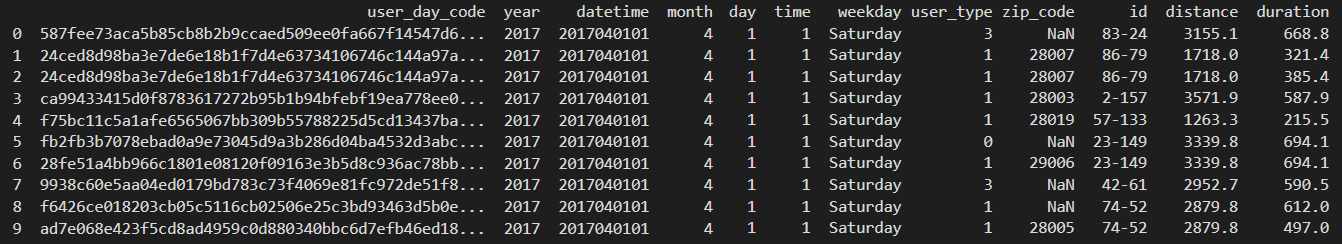


Ilustración 5.15. Captura de extracto de visualización de diez registros del conjunto de datos tras la creación de columnas específicas para año, mes, día, día de la semana y hora. Fuente: elaboración propia.

Para lograr la convergencia de este conjunto con los valores meteorológicos, se emplea una clave primaria generada a partir de los campos que recogen los atributos de la marca temporal. En concreto, se concatenan los registros de ‘*day*’, ‘*month*’ y ‘*year*’ separados por el carácter ‘/’. Este formato expresa cada día de forma unívoca y establece el nexo entre los registros del conjunto de datos de uso y *Openroute service* con los valores meteorológicos recopilados.

En otras palabras: cada registro de dato de uso se acompaña de valores meteorológicos registrados en el día que se produjo, tal y como se aprecia en la Ilustración 5.16.

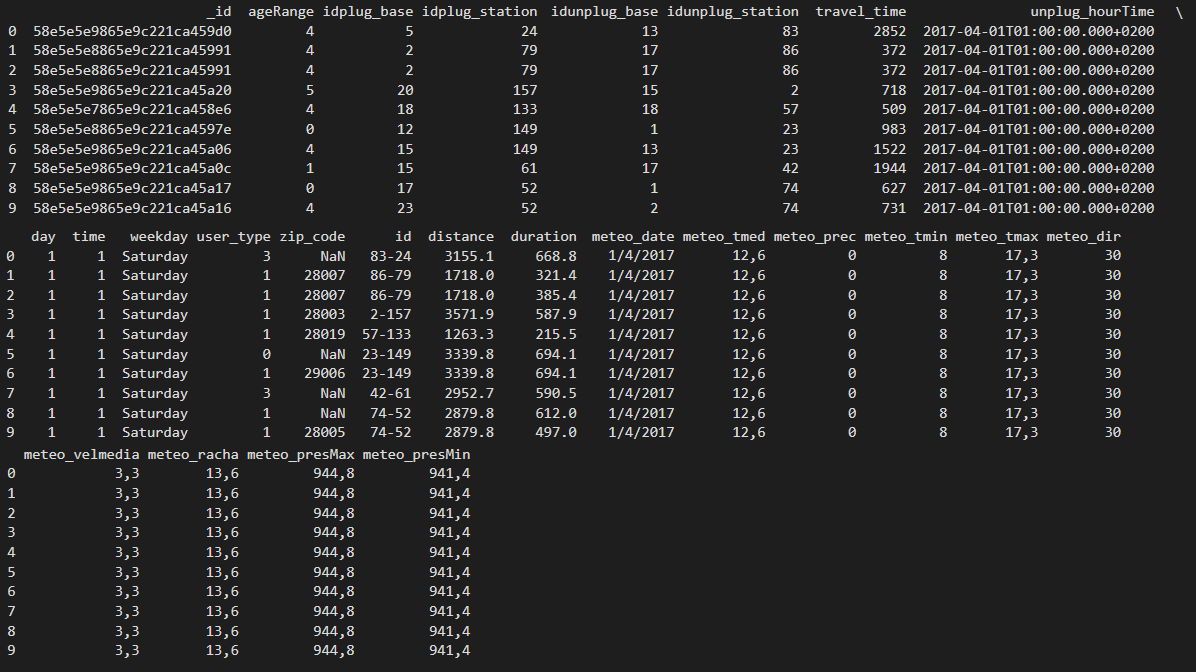


Ilustración 5.16. Captura de pantalla de extracto de visualización del conjunto de datos resultante de la unión de los datos adquiridos.

Por último, antes de analizar el conjunto de datos, se han convertido algunos de los campos a formatos más aptos para su análisis – como es el caso de los valores meteorológicos, anteriormente considerados cadenas de caracteres y ahora datos numéricos tipo *float*. Dicha conversión ha servido para detectar anomalías motivadas con la diversidad del conjunto de datos y sus fuentes.

Ejemplo de ello son los caracteres de separadores de decimales de los valores, para lo que en España se debe emplear la coma ‘,’ [62], aunque el Sistema Internacional admite tanto coma como punto ‘.’[63]. Por tanto, aunque ambos expresen valores de forma correctas, el uso de comas no es compatible con Python, que identifica dichos valores numéricos como cadenas de caracteres.

De la misma forma se han encontrado anomalías en algunos registros - en concreto en los valores meteorológicos del 21 de abril de 2019 - con valores y formatos inadecuados como se aprecia en la Ilustración 5.17.

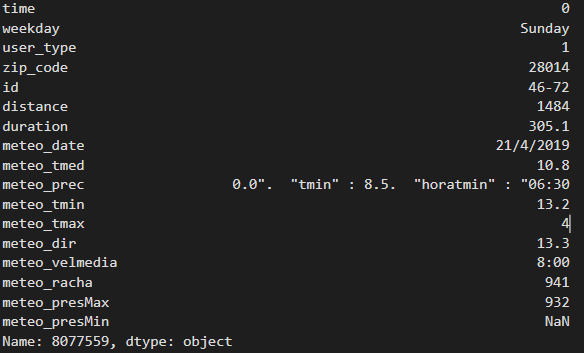
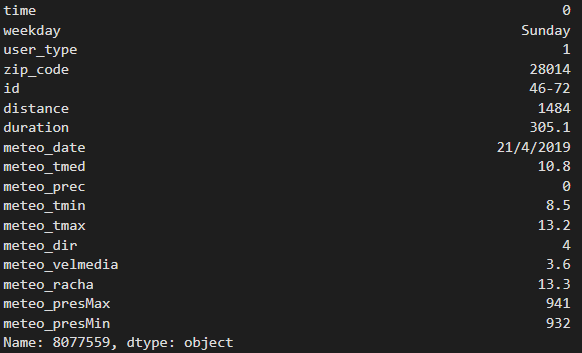


Ilustración 5.17. Captura de subsanación anomalía en el registro 807759, el cual muestra valores en campos que no corresponden (izquierda) y tras su corrección (derecha). Fuente: elaboración propia.

# Análisis de datos disponibles

Una vez aquí es posible explorar – en un sentido estadístico – el conjunto de datos objeto de estudio. Con la función ‘*.describe()*’ de pandas se permite obtener, de forma sencilla y generalizada, un conjunto de estadísticos para cada variable de formato numérico. En la Ilustración 6.1 se muestra el resultado de la misma, del cual se pueden realizar algunas apreciaciones a partir de la cuenta (‘*count’),* la media (‘*mean*’), desviación estándar (‘*std*’), valor mínimo (‘*min’*) y máximo (*‘max’*) registrados, así como los cuartiles primero ( ‘*25%’* que corresponde al percentil veinticinco), segundo (‘*50%’* que corresponde con la mediana o con el percentil cincuenta) y tercero (‘*75%’* que corresponde con el percentil setenta y cinco). Tanto el código correspondiente a la ejecución de esta función, como la del resto de análisis y gráficos empleados en esta sección se encuentran disponibles en el Anexo IV: Código Python correspondiente a generación de gráficos para el análisis descriptivo de los datos empleados.

En el caso de la variable ‘*ageRange*’, el percentil cincuenta nos indica que el usuario mediano se encuentra en un rango de edad entre 27 y 40 años. Este valor no solamente se ve afectado por la distribución de edades de los usuarios, sino por la amplitud del rango de edad. Los valores ‘2’ y ‘3’ representan intervalos de uno y siete años de edad respectivamente, frente a los 13 años del rango de edad representado con el valor ‘4’ o los veinticuatro del nivel ‘5’. En la Ilustración 6.2 se muestra un diagrama de barras que representa la frecuencia de cada grupo de edad en una muestra aleatoria de n=100.000 registros.

En la Ilustración 6.1 también se aprecian los valores de los estadísticos obtenidos a partir de las variables ‘*travel\_time’* y ‘duration’. Dichas variables corresponden a las duraciones reales y previstas – de acuerdo con el punto de origen y destino según *Openroute Service* – respectivamente.

De estos valores se observa por una parte que, pese a que en general los tiempos predichos son sustancialmente inferiores a los reales en los que se realizan los trayectos, ambas variables se registran en el mismo orden de magnitud. Por otra, la presencia de valores atípicos – también conocidos por el término inglés «*outliers*».

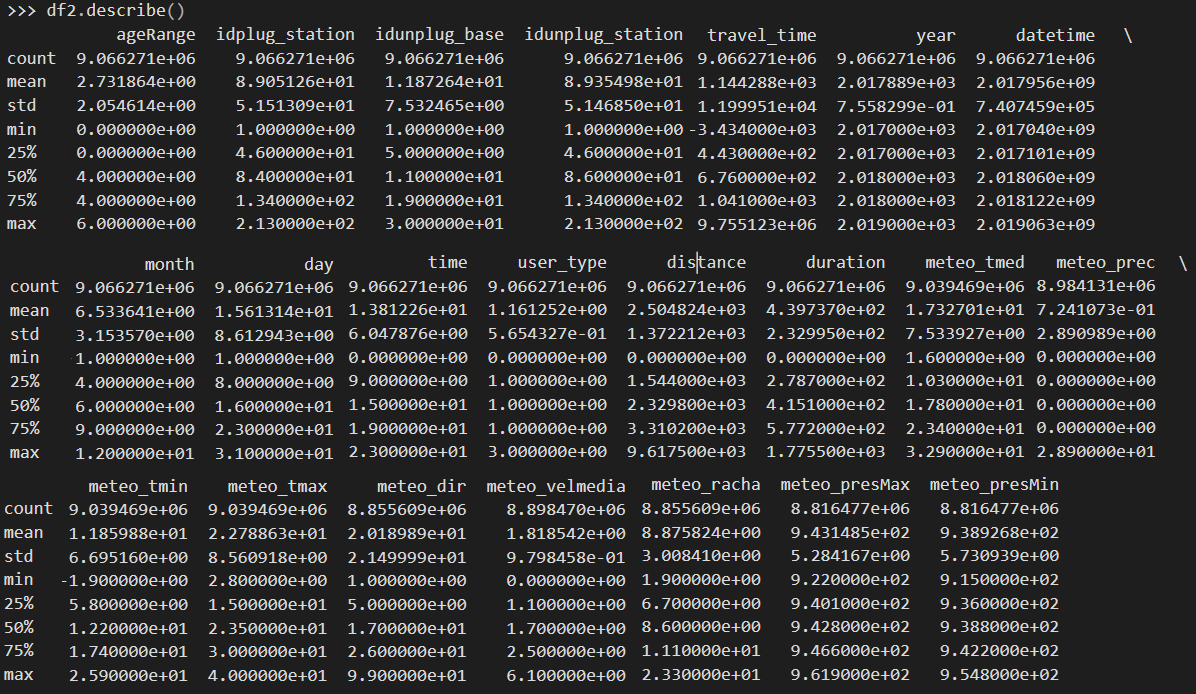


Ilustración 6.1. Captura de resultado de la ejecución de la función .describe() de Pandas donde se muestran estadísticos de las variables numéricas del conjunto de datos objeto de estudio. Fuente: elaboración propia.

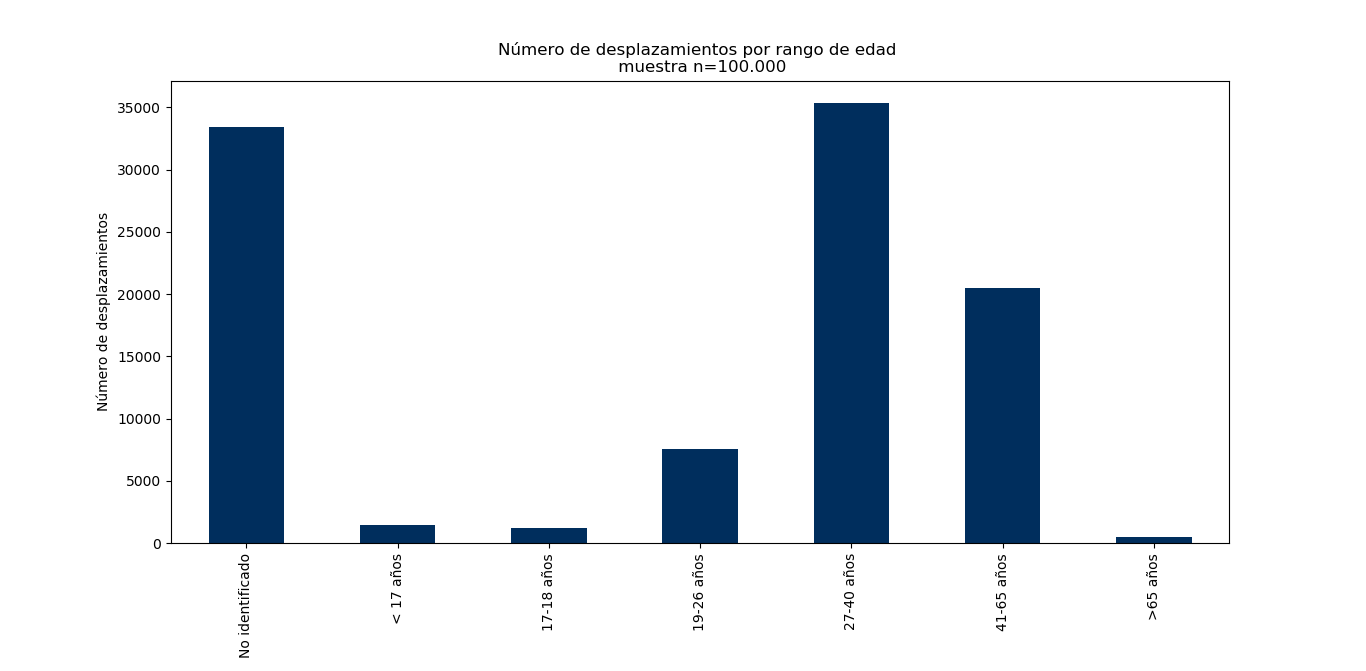


Ilustración 6.2. Diagrama de barras de desplazamientos registrados por rangos de edad (‘ageRange’), basado en una muestra aleatoria del conjunto compuesta por 100.000 registros. Fuente: elaboración propia.

En el conjunto de técnicas de la minería de datos se manifiestan diferentes consideraciones en cuanto al tratamiento de los valores atípicos. Aunque la presencia de los mismos pueda influir en el modelo predictivo de forma perniciosa, no deben desecharse directamente dado que pueden formar parte del propio comportamiento del sistema objeto de estudio, incluso su identificación es el fin de modelos de detección de anomalías – reconocimiento visual [64], detección de fraudes bancarios [65] o de ciberataques [66].

En el caso objeto de estudio del presente Trabajo Final de Grado corresponde diferenciar los valores atípicos propios de los errores de medición – como aquellos valores de ‘*travel\_time’* menores que cero – y aquellos que corresponden a registros propios del uso del sistema BiciMAD.

En la Ilustración 6.3 se encuentra el diagrama de caja que corresponde a las variables ‘*travel\_time*’ y ‘*duration*’ (basado en una muestra de tamaño n=100.000 registros). En la primera de ellas se aprecian valores, acusadamente dispersos, que impiden que se distinga la representación de los cajones. Por tanto, los registros de BiciMAD recogen algunos valores excesivamente dilatados en comparación a lo previsto por *Openroute Service*.

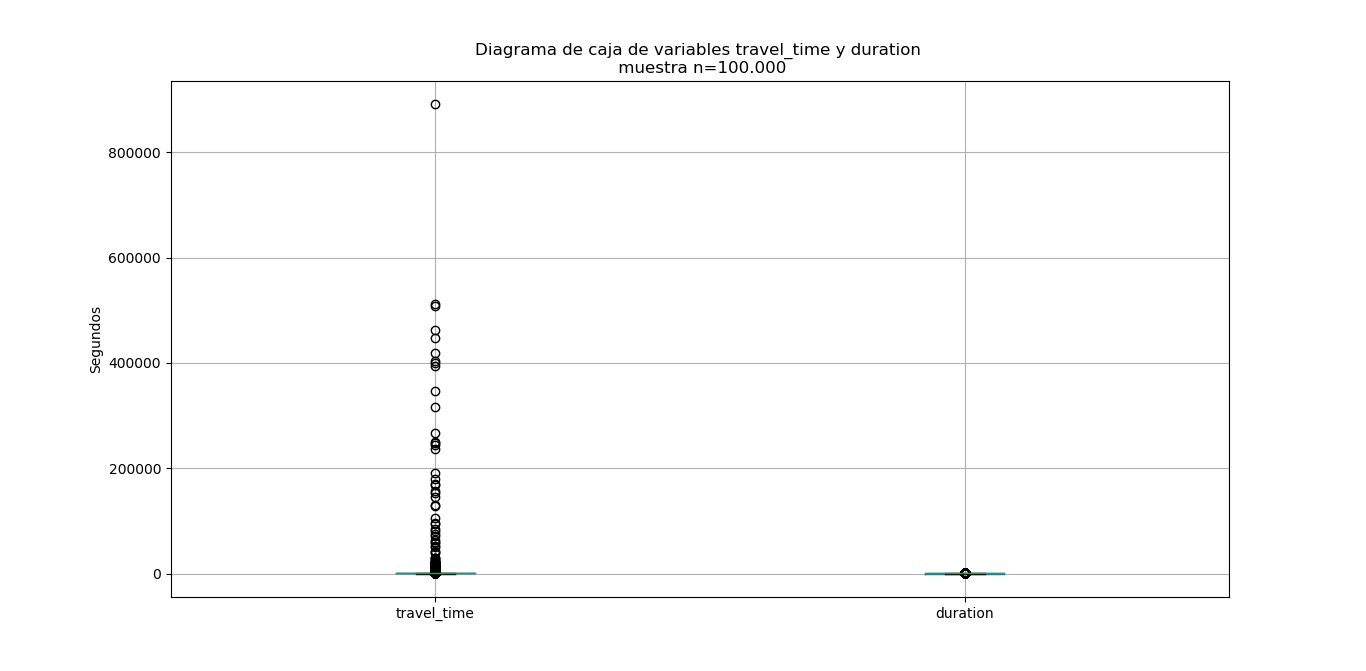


Ilustración 6.3. Diagrama de caja y bigote de las variables travel\_time (izquierda) y duration (derecha). Fuente: elaboración propia.

En la Ilustración 6.4, donde se muestra una vista ampliada centrada en el diagrama de la variable ‘*duration’*, se aprecian diferencias significativas en la distribución de los cuartiles y medianas, algo ya apreciable en los datos obtenidos al ejecutar la función *‘.describe()’* (Ilustración 6.1). Por ello cabe suponer que la duración prevista por la aplicación *Openroute Service* tiende a ser inferior al tiempo real empleado por los usuarios de BiciMAD para efectuar dichos desplazamientos. La Ilustración 6.5 permite apreciar la distribución de tiempos predichos y realmente empleados para una muestra de registros. Aquellos que quedan por encima de la línea representada en rojo, supone un tiempo empleado mayor al predicho, mientras los que quedan bajo la línea se encuentran en la situación inversa. Posibles causas de este desajuste pueden ser la falta de precisión del calculador de rutas – puede suponer una velocidad mayor a la efectiva, obviar aspectos como congestión, tiempo de esperas por semáforos, cortes por vías no transitables o cansancio del usuario – así como por un uso distinto al estricto desplazamiento – por ejemplo usuarios que hacen un uso recreativo. Igualmente, atendiendo a la distribución del mismo parece resulta una variable proporcional a la otra. En cualquier caso, el sesgo mostrada por esta variable en comparación a los tiempos real no será especialmente perjudicial tanto en cuanto el escalado de los valores registrados en las variables es una práctica habitual – incluso necesario en la aplicación de muchas técnicas-, dado que el valor de la información obtenida no reside tanto en el valor registrado sino en la posición relativa del valor registrado en la distribución presentada por el total de registros de dicha variable [67].

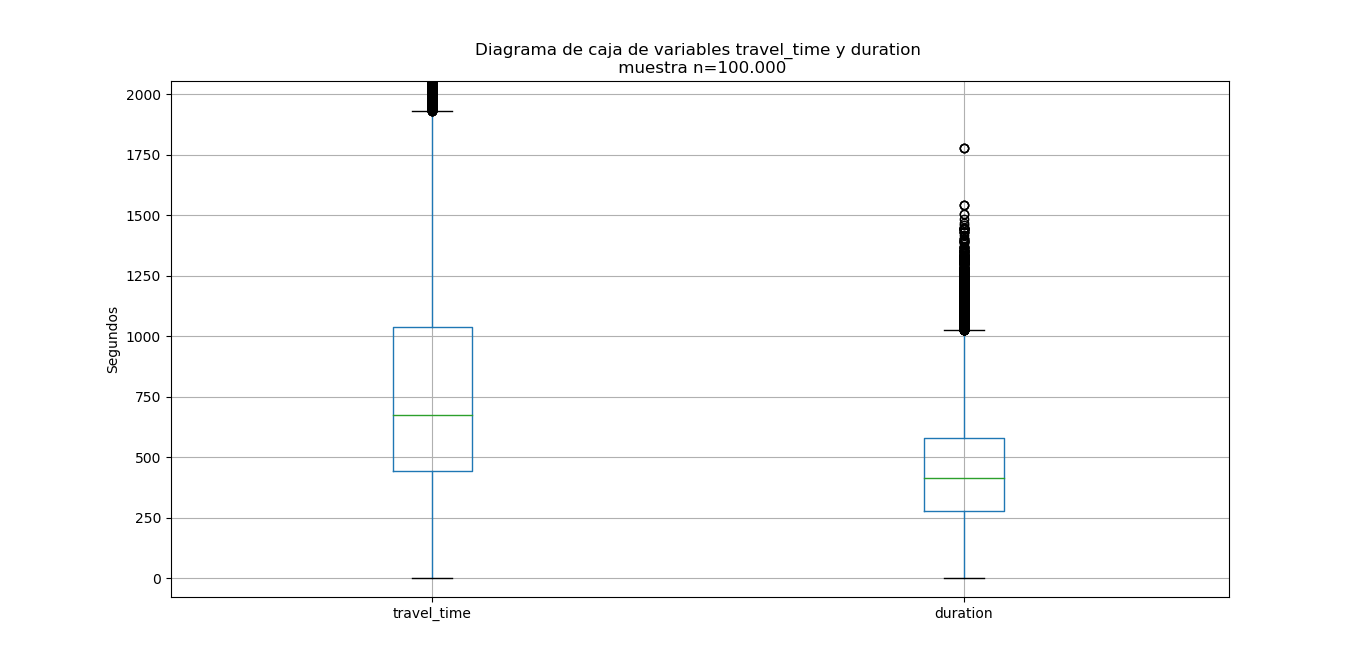


Ilustración 6.4. Vista ampliada de diagramas de caja y bigote de las variables travel\_time (izquierda) y duration (derecha). Fuente: elaboración propia.

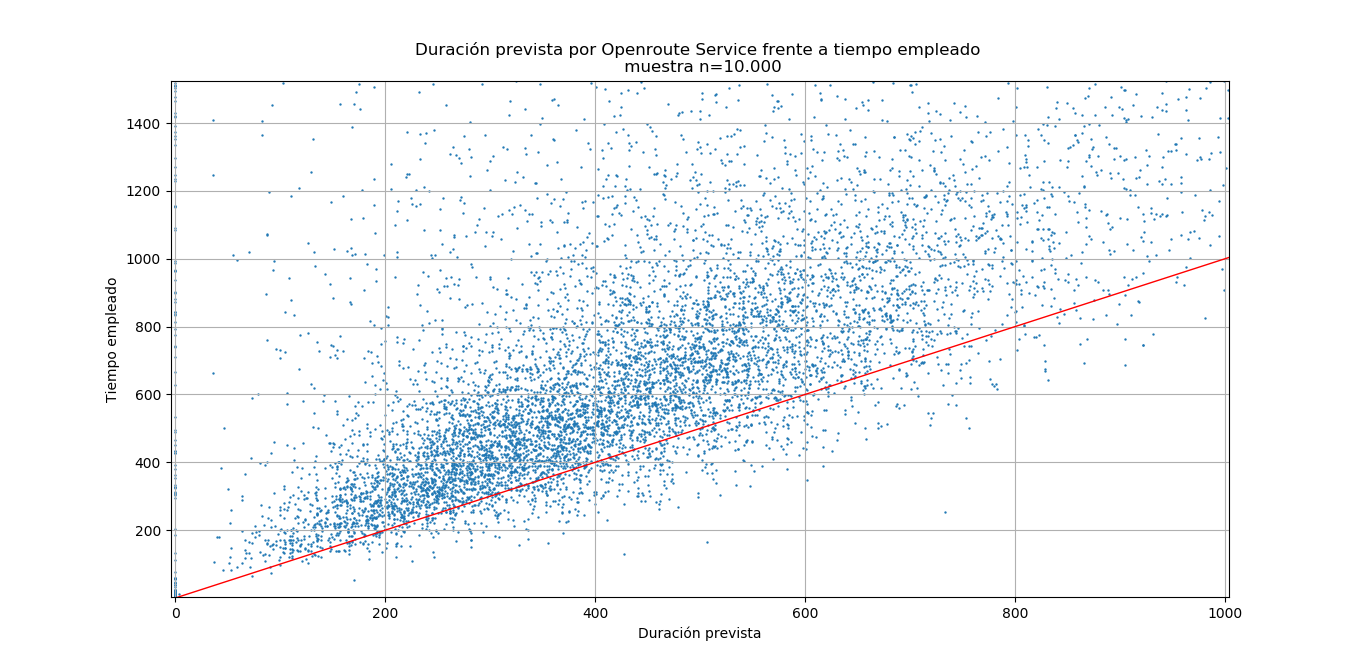


Ilustración 6.5. Nube de puntos relativa a correspondencia entre tiempo predicho y empleado por cada desplazamiento, a la cual se superpone una recta de pendiente 1. Vista ampliada del gráfico basado en una muestra n=10.000 registros. Fuente: elaboración propia. Fuente: elaboración propia.

En relación al tiempo de viaje en los desplazamientos, resulta de interés analizar la distribución general de los mismos para conocer el uso realizado del servicio BiciMAD. En la Ilustración 6.6 se muestra un histograma, basado en una muestra de 100.000 registros, donde se representa la frecuencia de la duración de los desplazamientos. Dicha representación se aproxima a una distribución asimétrica sesgada a la derecha, por lo que la mayoría de los registros se encuentran por encima de del tiempo de viaje medio – también inferior a mediana y moda. Destaca que alrededor del 1,75% desplazamientos registran un tiempo de viaje próximo a cero, lo que posiblemente se deba casos en los que la bicicleta ha presentado algún problema técnico y ha vuelto a ser enganchada en la misma estación. En la Ilustración 6.7 se aprecia la distribución de tiempos de todos los registros con tiempo de viaje menor a 60 segundos, subconjunto que representa en torno al 2% de los registros.

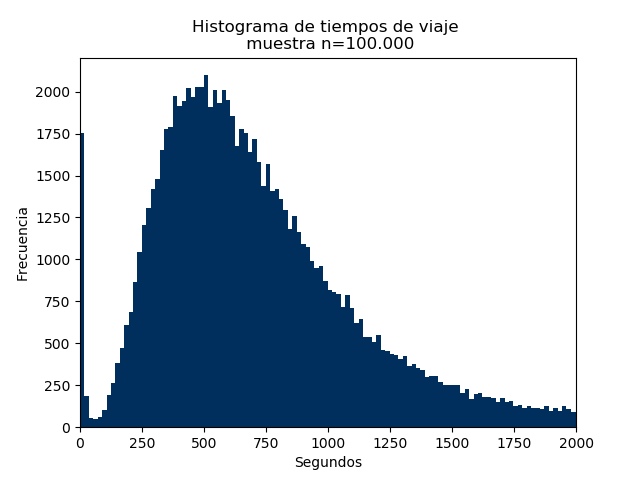


Ilustración 6.6. Histograma de variable ‘Travel\_time’ basado en muestra aleatoria n=100.000 registros. Fuente: elaboración propia.

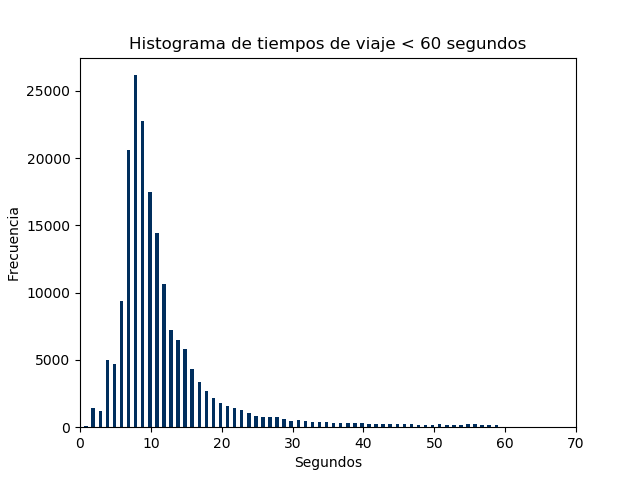


Ilustración 6.7. Histograma de valores de variable ‘Travel\_time’ menores a 60 segundos. Fuente: elaboración propia.

Atendiendo a la frecuencia de uso del servicio por meses y por días de la semana - cuyos diagramas de barras se muestran en las ilustraciones Ilustración 6.8 y Ilustración 6.9 respectivamente (para una muestra de n=100.000 registros). En el primer caso, se aprecia cierta estacionalidad en el uso del servicio, alcanzando máximos en aquellos meses con una temperatura más suave - alrededor de 20 grados en los meses de junio y septiembre tal y como se muestra en la Ilustración 6.10. Por tanto, cabe suponer que existe una influencia directa de la meteorología en posibles patrones de uso del servicio. En el segundo caso, realizando la misma observación en base a los días de la semana en los que se produce el desplazamiento, aparentemente no muestra estacionalidad – al menos atendiendo solamente al día de la semana en el cual se producen los desplazamientos. Los máximos se aprecian en miércoles, viernes y domingo, lo que contrasta con mínimos de frecuencia de uso en jueves y sábados. Por ello, no cabe descartar alguna influencia de esta característica en algún posible patrón de comportamiento del usuario y uso del servicio.

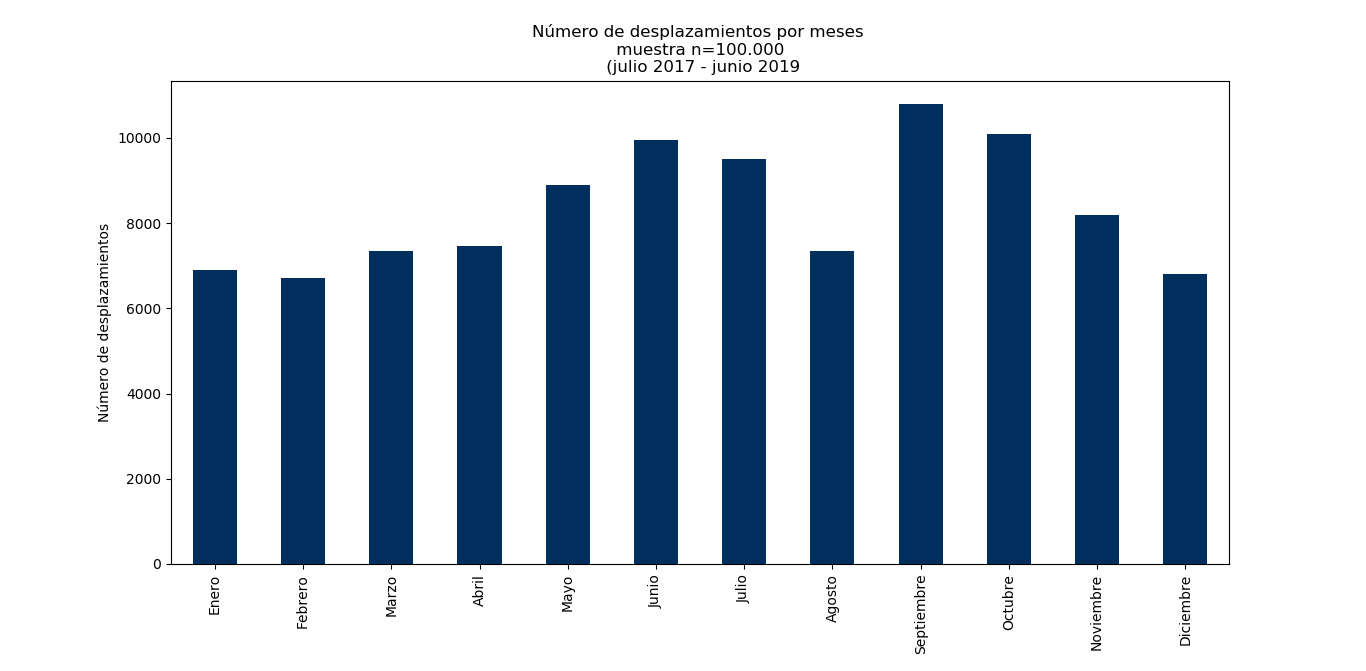


Ilustración 6.8. Diagrama de barras de frecuencia de uso del servicio BiciMAD por meses basado en una muestra de n= 100.000 registros pertenecientes al periodo junio 2017 – junio 2019. Fuente: elaboración propia.

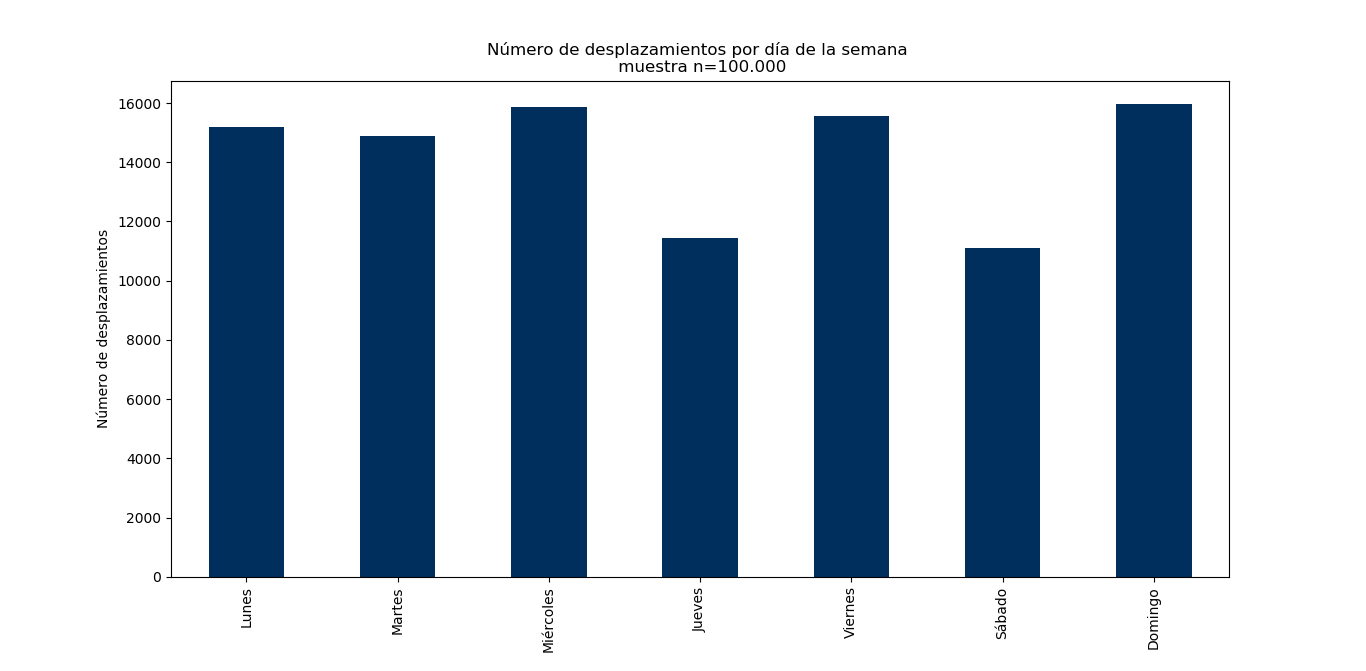


Ilustración 6.9. Diagrama de barras de frecuencia de uso del servicio BiciMAD por días de la semana basado en una muestra de n= 100.000. Fuente: elaboración propia.

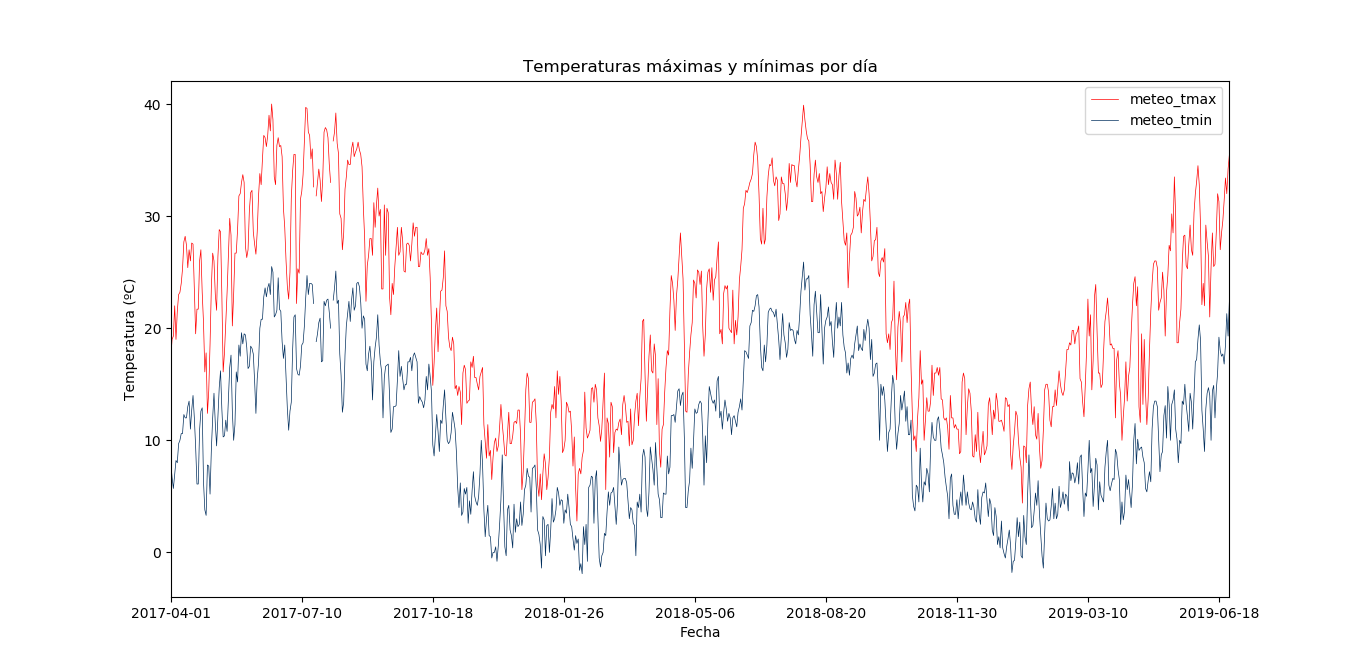


Ilustración 6.10. Temperatura máxima (rojo) y mínimas (azul) registradas diariamente en la estación meteorológica del Parque del Retiro, Madrid. Fuente: elaboración propia a partir de datos de AEMET.

De la misma forma, los gráficos mostrados en la Ilustración 6.10 – observados junto al contenido en la Ilustración 6.11 – refuerzan la suposición expuesta en el párrafo anterior: la intensidad de uso del servicio –en términos de número de registros – puede verse incidida por los registros de temperaturas máximas y mínimas, entre otros registros meteorológicos. Haciendo extensiva esta observación en términos de duración de los desplazamientos registrados, en la Ilustración 6.12 se aprecia como dicha estacionalidad también se percibe en los valores medianos de la variable ‘*travel\_time’* por días. A diferencia del caso anterior, la duración de trayecto mediana tiende a ser menor en los meses de invierno y mayor en los de verano cuando. Aunque el servicio registra menos desplazamientos, la duración de los mismos pueden verse favorecidas por las temperaturas cálidas, entre otras variables meteorológicas que deben de afectar, tales como precipitaciones o rachas de viento en todos los periodos considerados.

En los gráficos mostrados en la Ilustración 6.11 se aprecia un crecimiento considerable, que ocurre de forma súbita en junio de 2019. Aunque no existe ninguna causa confirmada para dicho aumento tan significativo, sí coincide el inicio de una importante ampliación de capacidades y distritos que alcanza el sistema de BiciMAD [68].

Estas mismas disertaciones – globales y sobre el sistema considerado en su conjunto – pueden trasladarse a casos y condiciones concretas como la demanda y disponibilidad de servicio en cada una de las estaciones. Ejemplo de ello es el gráfico contemplado en la Ilustración 6.13, donde se representa el flujo neto de bicicletas (diferencia entre trayectos con origen y trayectos con destino) en la estación 1 (Puerta del Sol) en el periodo abril 2017- junio 2019.

Tras un análisis descriptivo del conjunto de datos del cual se parte, una aplicación adecuada de las herramientas de minería de datos será capaz de identificar patrones en la dinámica del servicio de entre una multitud de variables concurrentes, en un conjunto de datos tan voluminoso como el que se considera objeto de estudio en el presente Trabajo Fin de Grado.

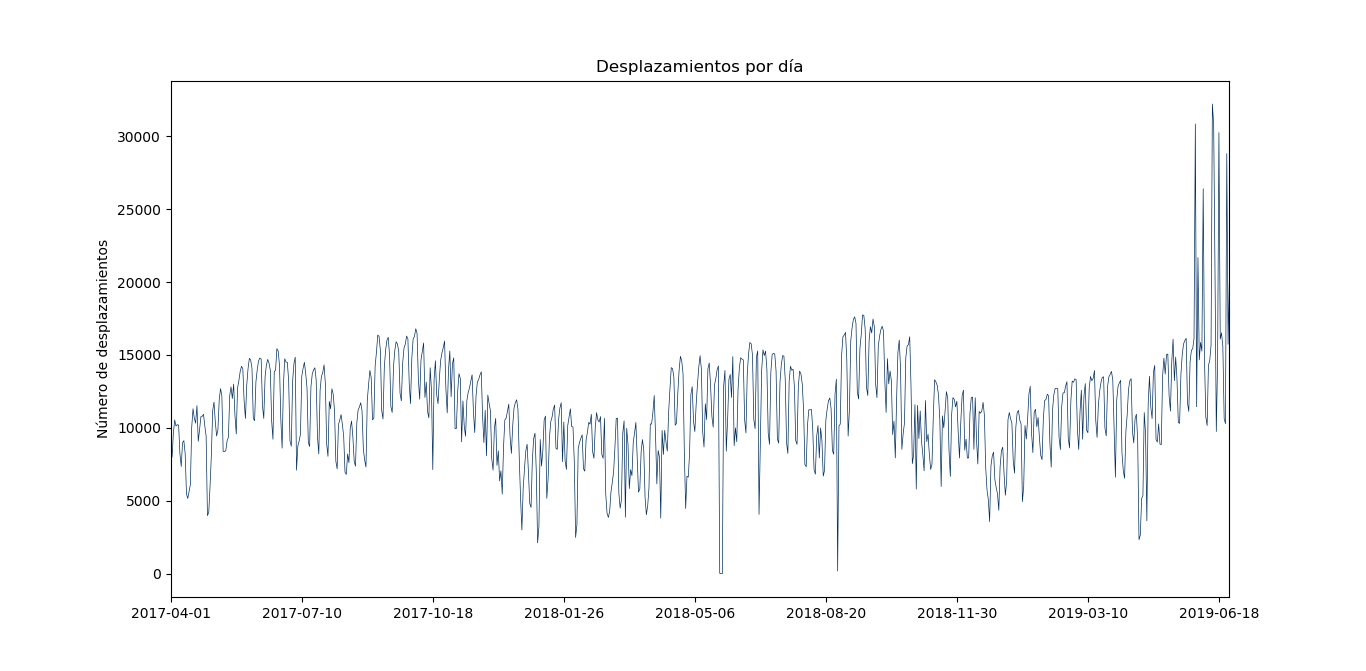
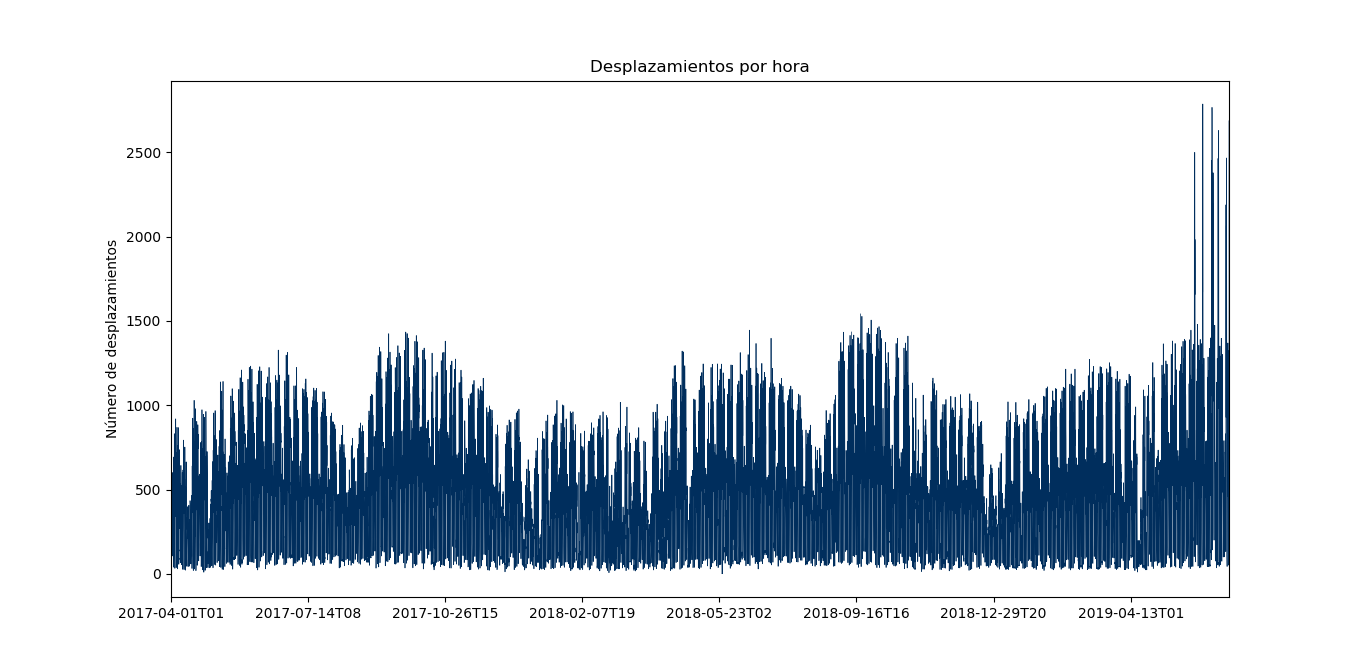


Ilustración 6.11. Gráficos correspondientes al número de desplazamientos registrados por día y hora (arriba) y al número de desplazamientos por día en el servicio BiciMAD entre abril de 2017 y junio de 2019. Fuente: elaboración propia a partir de datos de EMT Madrid.

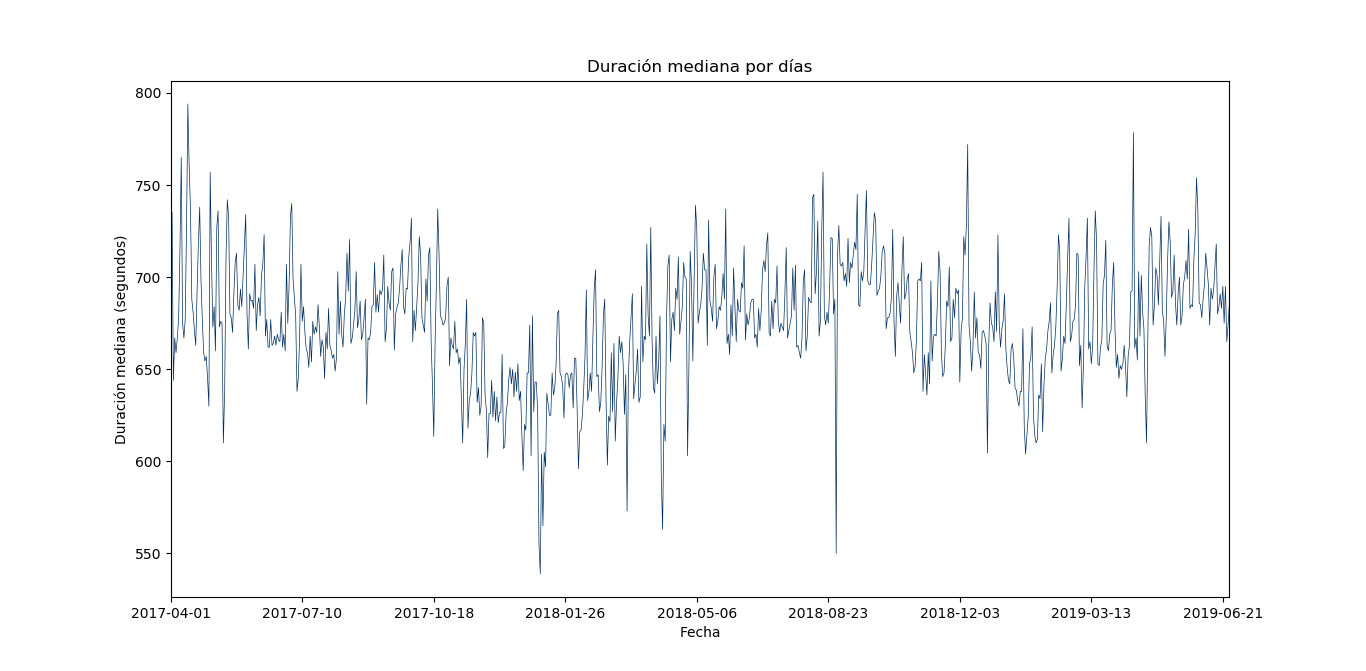


Ilustración 6.12. Gráfico de duración mediana de los desplazamientos en el servicio BiciMAD por días en el periodo abril 2017 a junio de 2019 (desechados valores relativos al 25, 29 y 30 de mayo de 2018 para facilitar la visualización de la tendencia). Fuente: elaboración propia a partir de los datos de EMT Madrid.

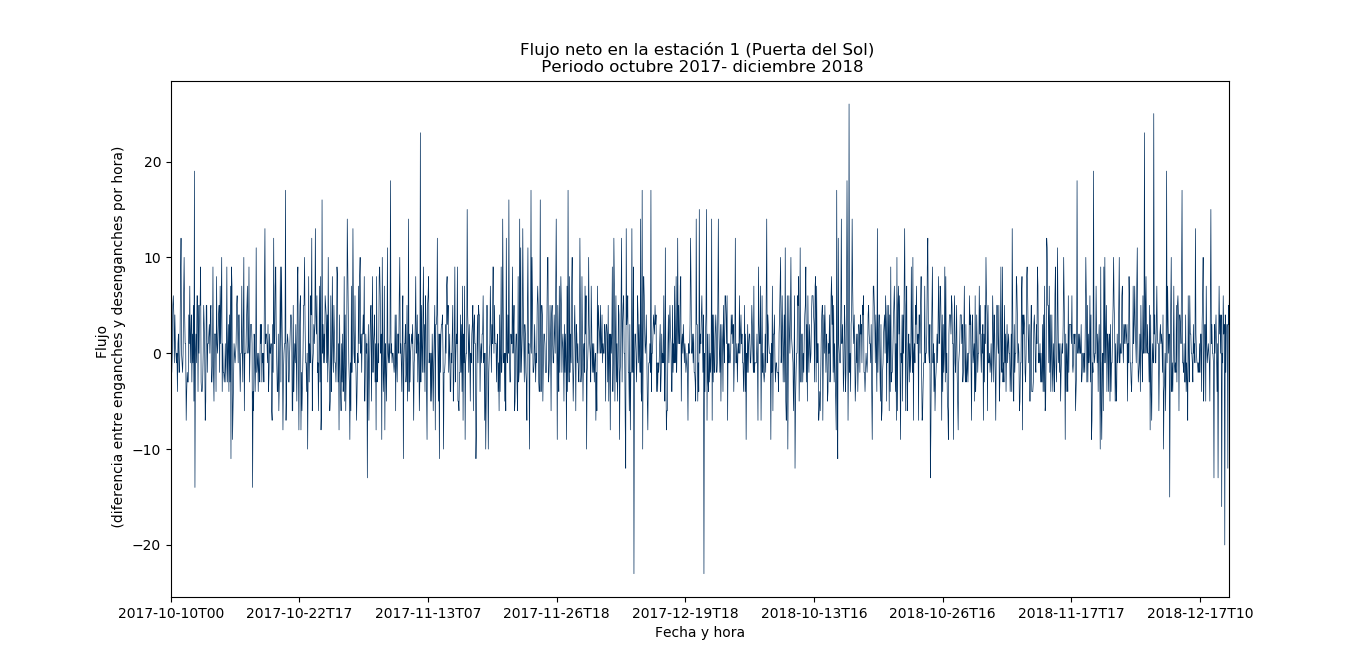


Ilustración 6.13. Flujo neto (diferencia entre desplazamientos iniciados y finalizados) en la estación 1 (Puerta del Sol) de Bicimad por día y hora en el periodo octubre de 2017 - diciembre 2018. Fuente: elaboración propia a partir de los datos de EMT Madrid.

# Líneas de trabajo con minería de datos

Una vez se ha procedido a la adecuación y exploración de los valores contenidos en el conjunto de datos, procede la definición de varias líneas de trabajo que permitirían el desarrollo de diferentes modelos predictivos, partiendo siempre del conocimiento general del servicio y de sus operaciones.

En una situación ideal, el operador del servicio puede contar con este mismo conjunto de datos sin haber excluido campos como la identificación del usuario abonado en cada uno de sus desplazamientos. Dichos registros serían de gran interés dado la utilidad que podría resultar del desarrollo de posibles modelos predictivos. Ejemplo de ello es la clasificación de los usuarios – con técnicas como árbol de clasificación o «K-vecinos más próximos» (o *K-nearest neighbors*) –lo que habilitaría la predicción de características principales de un desplazamiento en el momento del desenganche de la bicicleta y atendiendo a condiciones ambientales como la meteorología o momento en el que se produce. De la misma forma, permitiría mejorar el conocimiento de los patrones de uso y demanda del servicio, aspectos esenciales para la optimización del mismo y la gestión la oferta a través de la disponibilidad de unidades.

Partiendo de los datos disponibles, se definen como realizables dos posibles líneas de trabajo, entre otras muchas posibles:

* A través del conteo de desplazamiento por días, entrenamiento y validación de un modelo que prediga la demanda en base a condiciones ambientales (como el pronóstico meteorológico) y factores de estacionalidad (mes, día de la semana, etc). Dicho modelo puede aplicarse a cada una de las estaciones, por zonas o distritos, o de forma global a todo el sistema.
* Clasificación de tipo de usuarios y/o estaciones que permita predecir la duración y/o distancia de un recorrido, haciendo uso de variables auxiliares tales como velocidad media prevista o factores de distancias y tiempos reales frente a previstos.

Ambos modelos pueden resultar complementarios en un ámbito de aplicación real del mismo, en el cual podrían consolidarse como un proceso dinámico y retroalimentado – constituyendo una máquina de aprendizaje – en el cual deberían integrarse servicios API como los de AEMET, *Openroute service* o equivalentes. Debido a la limitación de recursos computacionales y de la delimitación del alcance del presente Trabajo Fin de Grado, se decide desarrollar el primero de los modelos definidos y posponer la segundo como línea de trabajo propuesto para futuras ocasiones.

A la línea de trabajo seleccionada se asocia, como hipótesis, que la meteorología y determinados factores de estacionalidad influyen en la demanda experimentada por el sistema BiciMAD.

## 7.1 Procesamiento de datos: Imputación de valores omitidos

Uno de inconvenientes más frecuentes en la aplicación de técnicas de minería de datos, es la existencia de valores nulos en los registros empleados para entrenar y evaluar cualquier modelo. Dichos valores nulos pueden ser resultado de errores de lectura en sensores, valores que no aplican (campos opcionales), filtrado de valores no admisibles, errores en la entrada y procesado, entre otras causas. Para hacer frente a ello se recomiendan diferentes estrategias [69]:

* Supresión de registros que contienen valores nulos, renunciando a una parte del volumen de los datos considerado inicialmente.
* Supresión de variables que contengan valores nulos, con la consiguiente pérdida del valor de las variables de datos omitidas.
* Imputación de valores que pudieran resultar razonables, tales como:

-Media o mediana de los valores registrados en dicha variable.

- Regresión – ya sea líneal o no – a partir del resto de valores no nulos.

- Cualquier otra posibilidad, ya sea a través de algoritmos y modelos extendidos (*NIPALS, JointM, IPCA*, etc) u otros confeccionados ad hoc para el caso de aplicación.

En el caso de los datos que son objeto de estudio en el presente trabajo, se ha aplicado la combinación de varias de estas estrategias para la resolución de valores nulos (Anexo V: Código Python correspondiente a imputación de valores omitidos en conjunto de registros meteorológicos. En el caso de los registros meteorológicos de AEMET, muchos incluyen valores no válidos – procedentes de posibles errores de lectura de algún sensor o no disponibilidad de la estación meteorológica seleccionada – tal y como se aprecia en la Ilustración 7.1, que corresponde a la ejecución del atributo ‘*.info()*’ del conjunto de datos antes de la subsanación de valores nulos.

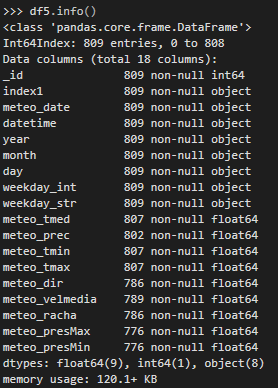


Ilustración 7.1. Captura de pantalla del resultado de ‘df5.meteo’ donde se aprecian valores nulos en registros de temperatura, precipitaciones, dirección y velocidad del viento y de presión atmosférica. Fuente: elaboración propia.

Tras la exploración de los valores con registros nulos, y la interpretación de las variables implicadas y la incidencia de dicho fenómenos en ellas, se ha procedido a la ejecución de una estrategia de resolución del problema.

Cabe reseñar que se ha desestimado la supresión de registros con valores nulos dado que, al relacionarse posteriormente con los registros de desplazamiento, la no disponibilidad de datos meteorológicos para un día concreto supondría renunciar a miles de registros de desplazamiento. Ello conllevaría una merma considerable del volumen total de datos.

En su lugar, partiendo de las variables del registro diario de temperatura máxima y temperatura mínima (‘*meteo\_tmax’* y ‘*meteo\_tmin’* respectivamente), las cuales tan solo registran dos valores nulos no consecutivos, se ha procedido a la imputación de los mismos. En los casos de ‘*meteo\_tmax’* , ‘meteo\_tmed’ y ‘*meteo\_tmin’*, se ha inferido cada imputación como de cada valor nulo como el promedio entre los registros equivalentes en el día anterior y posterior. El código y el resultado de dicha aplicación se aprecia en la Ilustración 7.2.

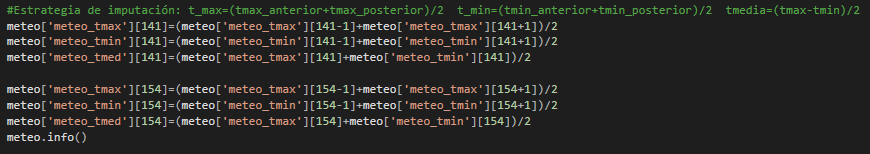


Ilustración 7.2. Capturas de pantalla de código correspondiente a las imputaciones de valores nulos en los registros 141 y 154 de variables de temperatura (arriba), así como la salida de la instrucción ‘meteo.info()’ (abajo) tras la ejecución del código anterior . Fuente: elaboración propia.

A continuación, y a partir de las variables numéricas sin valores nulos – es decir, ‘*meteo\_tmed’*, ‘*meteo\_tmin’*,*’meteo\_tmax’*,*’day*’,*’month*’ y ‘*year*’ – se han obtenido valores razonables a imputar en los valores nulos de la variable ‘*meteo\_presMax*’ tal y como consta en la Ilustración 7.3.

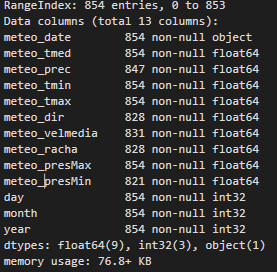
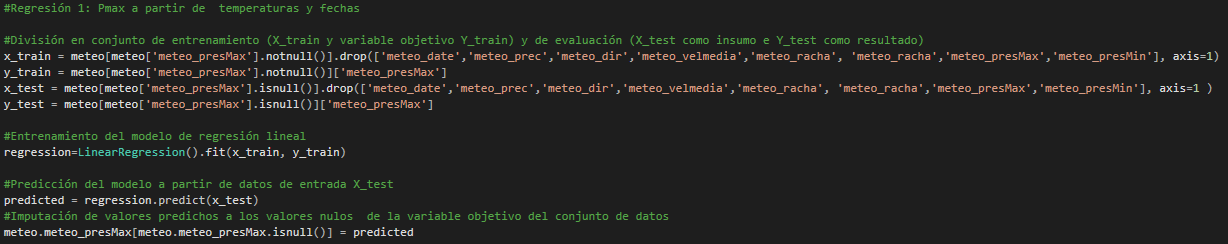


Ilustración 7.3. Capturas de pantalla del código correspondiente imputación de valores nulos de variable ‘meteo\_presMax’ mediante regresión lineal (arriba) y de la salida de la instrucción ‘meteo.info()’ (abajo) tras la ejecución del código anterior. Fuente: elaboración propia.

Una vez los valores nulos de la variable ‘*meteo\_presMax’* han sido reemplazados, esta variable queda habilitada a participar en el entrenamiento de nuevas estrategias y modelos de regresión lineal. De esta forma los valores nulos de la variable ‘*meteo\_presMin’* se reemplazan mediante la imputación de valores obtenidos de una nueva regresión lineal, tal y como se muestra en la Ilustración 7.4.

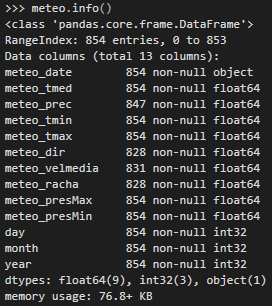
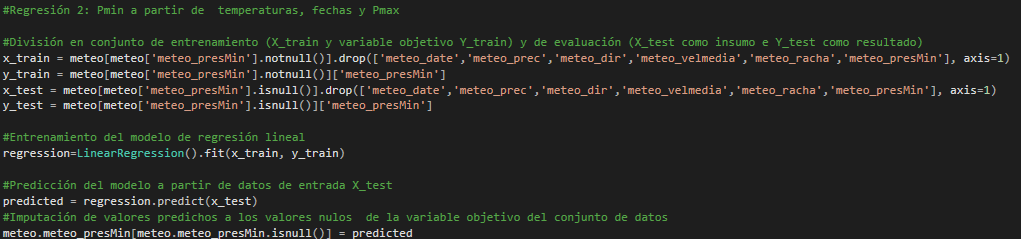


Ilustración 7.4. Capturas de pantalla del código correspondiente imputación de valores nulos de variable ‘meteo\_presMin’ mediante regresión lineal (arriba) y de la salida de la instrucción ‘meteo.info()’ (abajo) tras la ejecución del código anterior. Fuente: elaboración propia.

Tras ello se ha procedido a verificar que ningún valor mínimo de presión es superior al valor de presión máxima de dicho registro, en cuyo caso se ha modificado ‘*meteo\_presMin’* para que la variación en dicho registro resulte equivalente al promedio de oscilaciones de presión atmosférica de todos los registros (Ilustración 7.5).

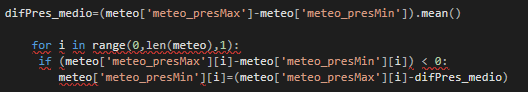


Ilustración 7.5. Captura de pantalla del código de imputación de valores de presión mínimas para aquellos que resulten, tras la regresión lineal, mayores a la presión máxima. Fuente: elaboración propia.

De forma similar se ha replicado esta imputación e incluido de forma sucesiva las variables restantes en el siguiente orden: *’meteo\_prec’,*  ‘*meteo\_velmedia’,* ‘*meteo\_dir’* y ‘*meteo\_racha’.* De esta forma, se ha pretendido que la inferencia de valores nulos por regresión lineal se produzca entre variables que mantienen algún tipo de correlación, como sucede con estas variables meteorológicas. Es decir, la temperatura atmosférica mantendrá relación con la presión atmosférica, que a su vez lo hará con el viento y las precipitaciones (Wooten, 2011). Igualmente, la secuencia de variables inferidas se ha producido de tal forma que se ha hecho primero en aquellas con un número inferior de valores nulos, las cuales han participado en la inferencia de otras variables posteriores con mayor cantidad de registros por imputar.

Tras finalizar este proceso iterativo, el conjunto de datos queda habilitado para su almacenamiento.

Una vez queda consolidado dicho registro de valores meteorológicos, donde se ha producido la subsanación de valores nulos, es posible repetir la asociación relacional, en base a la fecha de registro, con los registros de desplazamientos. Dichos valores nulos no han supuesto un inconveniente para la exploración de los datos en el apartado de análisis de datos – de hecho, la imputación de valores razonables desvirtuaría la visualización de los datos adquiridos. El resultado de dicho proceso de asociación se puede apreciar en la comparación de imágenes dispuestas en la Ilustración 7.6.

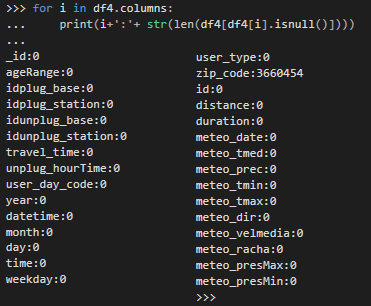


Ilustración 7.6. Capturas de pantalla de relación de variables y cantidad de valores nulos que contienen antes de asociar el conjunto de registros meteorológicos sin valores nulos (izquierda) y después de ello (derecha). Fuente: elaboración propia.

Tal y como se aprecia, la variable ‘*zip\_code’* – correspondiente al código postal del domicilio declarado por algunos usuarios abonados – es la única que conserva valores nulos. Debido al escaso valor que puede ofrecer en la predicción de la demanda que se le presupone, además de resultar un código numérico y no un valor cuantitativo, no tendría sentido la imputación de un valor razonable – tal como el promedio, mediana o resultado de regresión lineal – y por ello se opta por la supresión de esta variable.

## 7.2 Aprendizaje no supervisado: reducción dimensional

En cada una de las líneas de trabajo considerados se cuenta con un conjunto de datos con un amplio número de variables. Cada una de ellas dispone datos para cada registro, que pueden resultar de mayor o menor relevancia en la obtención de información y conocimiento a partir de ella. Es decir, cada una de las variables puede contribuir a describir, en mayor o menor medida, patrones de funcionamiento del sistema objeto de estudio. De acuerdo con esta característica, se ha recurrido a la aplicación de la técnica de reducción dimensional a través del análisis de componente principal (Anexo VII: Código Python correspondiente a número de variables a considerar en análisis PCA) – conocido en inglés como *principal component analysis* o PCA – la cual ha sido presentada en capítulo «Minería de datos».

En primer lugar, se ha realizado la segregación de los valores objetivo a predecir – ‘Y’ correspondiente en este caso con la variable ‘*\_id’* que contiene el conteo de desplazamientos producidos en la fecha referida – del resto de variables numéricas – contenidas en ‘X’ – de las que se espera que contribuyan a la determinación del valor objetivo (Ilustración 7.7).



Ilustración 7.7. Esquema explicativo sobre la segregación del conjunto de datos de uso del servicio por días en variable objetivo (Y) y variables de entrenamiento (X). Fuente: elaboración propia.

Para ello, dado los diferentes órdenes de magnitud de los valores contenidos en cada una en las diferentes variables, se aplica una escala uniforme a las variables de entrenamiento en X a través de la expresión definida en la Ecuación 7.1.

Ecuación 7.1. Expresión empleada por ‘StandardScaler()’ dentro de la librería Scikit Learn. Fuente: elaboración propia a partir de recurso propio de Scikit Learn.

donde Z es el valor calculado a partir del promedio de la variable ( ), el valor registrado inicialmente () y la desviación típica de los registros de dicha variable (). Las complicaciones derivadas de la heterogeneidad en las órdenes de magnitud de distintas variables, se puede paliar sustituyendo dichos valores por indicadores de su posición relativa en la distribución de valores. Es decir, su valor normalizado, apreciable en la Ilustración 7.8. Este proceso habilita y simplifica procesos como la reducción dimensional o el entrenamiento de muchos modelos [70].

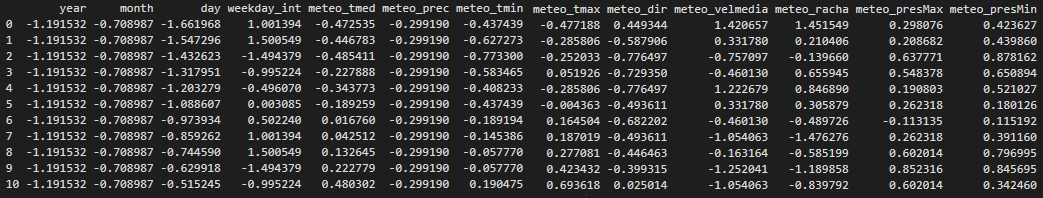
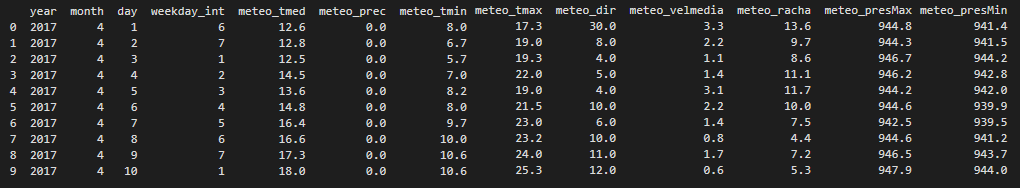


Ilustración 7.8. Capturas de pantalla que muestran los diez primeros registros del subconjunto de entrenamiento X antes (arriba) y después (abajo) del proceso de normalización.



Ilustración 7.9. Captura de pantalla de la matriz de covarianza, acompañada de leyenda en el lateral izquierdo en relación a la variable a la que corresponde el dígito numérico (de 0 a 12) dispuestos tanto en filas como en columnas. Fuente: elaboración propia.

Una vez escalado el conjunto de datos, es posible obtener la matriz de covarianzas entre las diferentes variables (Ilustración 7.9), la cual proporciona una primera aproximación a la dependencia lineal recíproca entre las diferentes variables presentes en el conjunto.

Ya sea a través del cálculo de autovalores y autovectores de la matriz de covarianza, o a través de la aplicación de funciones propias de librería ‘*Scikit Learn’,* es posible obtener los vectores que representan los componentes principales a emplear, siendo un parámetro a definir entre 1 y el número de variables que participan en la variable. El ya citado método de reducción dimensiones, pretende la expresión de la misma información contenida en un nuevo sistema de variables (componentes) – a través de un cambio de base – que reduce la dependencia lineal entre ellas. El método pretende evitar que las nuevas variables – denominadas componentes principales y las cuales no tienen por qué corresponderse con ningún significado concreto– expresen información redundante [71]. Para ello, define una primera componente principal de tal forma que, proyectando sobre ella los datos entrenados, maximiza la varianza de los mismos. De forma sucesiva se obtiene el resto de componentes principales, de forma similar a la explicada, pero condicionada a la no correlación con las variables anteriores – es decir, ortogonales a las mismas [72]. En otras palabras, las componentes principales pueden obtenerse a partir de los primeros autovectores ordenados en base a sus autovalores asociados (de mayor a menor). De esta forma se constituye una suerte de espacio vectorial en el cual, al expresar los datos – a través de la correspondiente matriz de cambio de base – permite la «concentración» de información acerca de las dinámicas de funcionamiento del sistema.

En la Ilustración 7.10 se representa el gráfico de peso o importancia de las diferentes variables sobre las dos componentes principales PC1 y PC2. Este gráfico, de uso frecuente en este tipo de análisis, representa la dependencia lineal entre las variables originales y las componentes principales. Las diferentes componentes lineales pueden expresarse como combinación lineal de las anteriores en los términos que se representa en dicho gráfico a partir de los coeficientes disponibles en la tabla apreciable en la Ilustración 7.11. En la misma se muestran dichos coeficientes para las componentes principales disponibles en este caso, a diferencia de la imagen anterior, donde solo se representan las dos primeras para facilitar así su visualización en un plano de «2D». Aunque su representación gráfica se dificulte o imposibilite, matemáticamente es posible realizar este ejercicio hasta alcanzar la totalidad de los doce componentes principales.

De esta forma, la componente principal podría evaluarse en este caso como

Ecuación 7.2. Función que evalúa el componente principal PC1 obtenido. Fuente: elaboración propia.

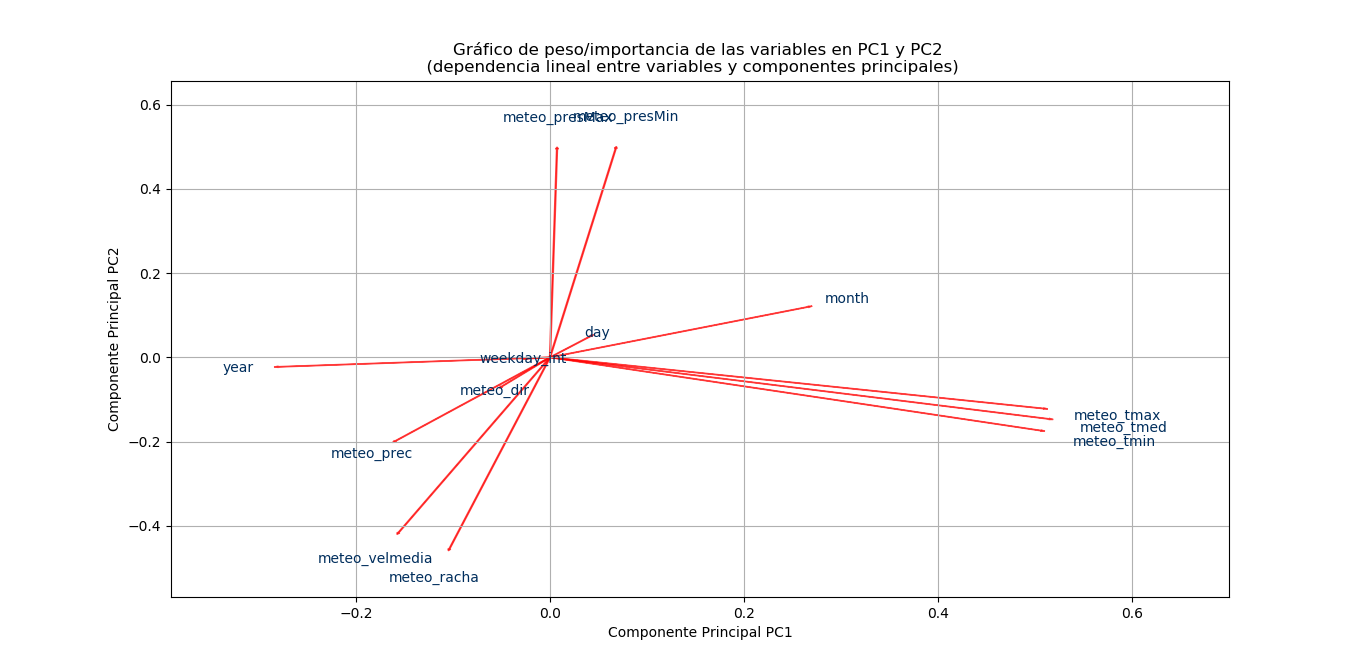


Ilustración 7.10. Gráfico de peso o importancias de las diferentes variables del conjunto de datos sobre las dos principales componentes PC1 y PC2. Fuente: elaboración propia inspirado en código del usuario «seralouk» en el repositorio Stack Overflow.[73].

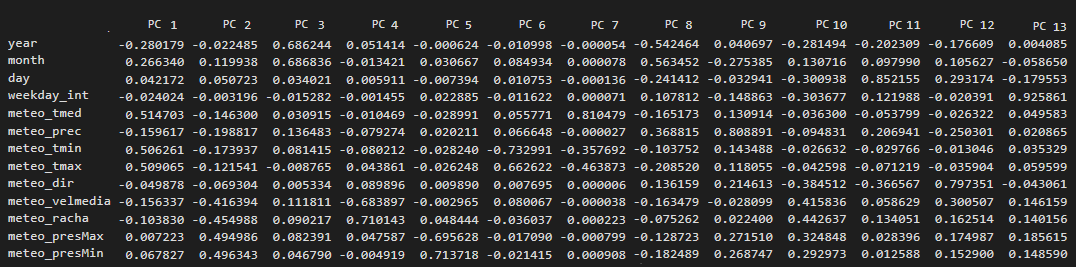


Ilustración 7.11. Tabla de coeficientes que definen los 12 componentes principales (columnas) como combinación lineal de las variables del conjunto de datos (filas). Fuente: elaboración propia.

A su vez, la aplicación de este método de aprendizaje no supervisado habilita tomas de decisiones que permiten la reducción de dimensional del conjunto de datos. Para ello se emplean los parámetros de «proporción de varianza explicada» y «proporción de varianza explicada acumulada», los cuales miden en qué proporción un modelo, o parte del mismo, da cuenta de la dispersión que produce en la variable objetivo.

En la Ilustración 7.12 se representa la proporción de varianza explicada acumulada (ordenadas) al emplear un determinado número de variables principales. De esta forma, si se emplease solamente la primera componente principal se obtendría un 26% de la varianza explicada. Empleando las dos principales un 48% y así sucesivamente hasta completar el 100% con las trece componentes principales, tal y como se aprecia en la Tabla 7.1.

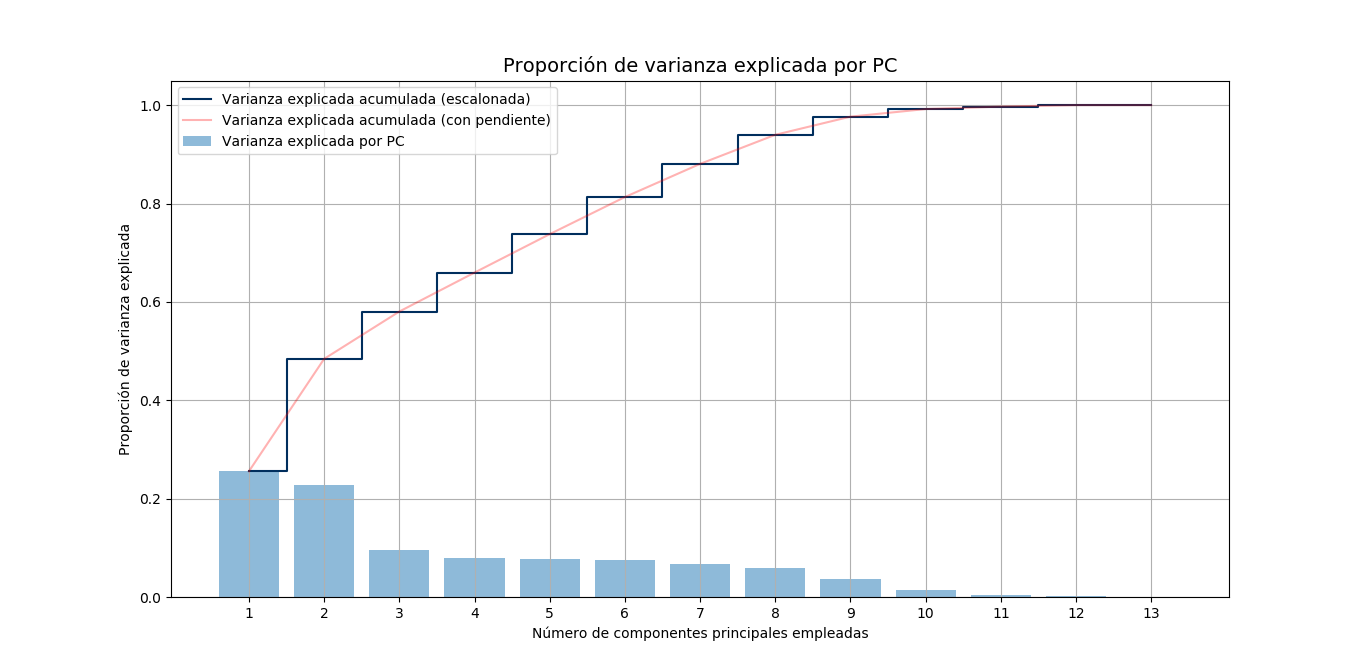


Ilustración 7.12. Gráfico que representa la proporción de varianza explicada por cada componente principal así como el valor acumulado por la misma y anteriores. Fuente: elaboración propia a partir de líneas de código (modificadas) de L. Lui [74].



Tabla 7.1. Tabla de proporción de varianza explicada por el número de componentes principales usadas. Fuente: elaboración propia.

La reducción dimensional del modelo supone la determinación del número de componentes principales a considerar, lo cual enfrenta la varianza capaz de expresar el modelo con limitaciones y reducciones del rendimiento del entrenamiento del modelo con técnicas de minería de datos que pueden emplearse posteriormente. Por ello esta decisión se refuerza con diferentes métodos entre los que se encuentran [75]:

* Selección arbitraria, o en base a las limitaciones de algún modelo, de un número determinado de componentes principales a emplear.
* Selección de una proporción varianza explicada acumulada objetivo a alcanzar, empleando el número de variables que primero alcance dicho valor (por ejemplo, un 70%, 80%, 95%, etc)
* Empleo del gráfico «*Scree plot*» (equivalente al representado en la Ilustración 7.12 y apoyado por la Tabla 7.1) para observar en qué punto se produce una reducción significativa en la tasa de crecimiento de la varianza acumulada. A partir de entonces el aporte de las componentes principales será menor y se sugiere su omisión, seleccionando únicamente aquellas componentes principales con una contribución más significativa.

De acuerdo con este último método y a la vista de lo mostrado en la Ilustración 7.12 así como de la Tabla 7.1, se aprecia una reducción significativa en la pendiente de la proporción de varianza explicada a partir de las nueve componentes principales. Por tanto, de optar por esta cantidad habilitaría una reducción dimensional, de alrededor del 30% respecto al subconjunto original, a la vez que se espera una proporción de varianza explicada de en torno al 98%. Esta decisión, que pudiera resultar un tanto conservadora en cuanto a esta última proporción, se sustenta al mostrarse un amplio reparto de la varianza explicada entre las diversas componentes principales. De igual forma, dato el volumen reducido de datos relativamente reducidos (alrededor de 800 registros) así como que se estiman recursos computacionales suficientes para las técnicas concretas que se prevén ensayar. La definición de dicho parámetro puede verse afectada por otros factores como tiempos de ejecución requeridos, nuevos modelos a ensayar o porque así lo considere criterio del analista de datos.

Por tanto, una vez decidido considerar las nueve componentes principales, se procede a la constitución de la matriz de cambio de base . De esta forma, los datos que se vayan a entrenar o a evaluar pueden trasladarse al subespacio vectorial definido, con la consiguiente reducción dimensional, a través del producto vectorial expresado en la Ecuación 7.3, donde es el número de registros considerados, representa al conjunto de datos a transformar y el mismo una vez realizado al cambio de base al subespacio obtenido a través del análisis de componente principal (PCA).

Ecuación 7.3. Producto matricial correspondiente al cambio de base constituido por las componentes principales empleadas. Fuente: elaboración propia.



Ilustración 7.13. Captura de pantalla de matriz de cambio de base W. Fuente: elaboración propia.

Los procesos hasta ahora descritos sirven de ayuda para analizar cómo responden los datos considerados al proceso de análisis de componentes principales (PCA), y en base a ellos, facilitar la toma de decisión del número de componentes a considerar. Una vez determinado, funciones propias de la librería «*Scikit Learn*» facilitan la aplicación de este método de forma más simplificada y eficiente a la hora de ser ejecutado.

En definitiva, esta técnica, a través de un cambio de base, permite la concentración de información en un número reducido de componentes principales y, a su vez, facilita la decisión de cuántas han de ser consideradas. Sin embargo, al tratarse de una transformación lineal, ofrece buenos resultados solo cuando la dinámica del sistema es expresable mediante correlaciones lineales.

De forma similar, el análisis de componentes independientes (ACI, o ICA por sus siglas en inglés) facilita el procesamiento computacional de modelos multivariables cuando las variables son estadísticamente independientes [76]. Esta técnica es frecuentemente empleada para la denominada «separación ciega de señales», consistente en la identificación de señales independientes superpuestas [77].

## 7.3 Selección de Modelo

Haciendo uso de los algoritmos de aprendizaje disponibles en la librería «*Scikit Learn*», se ha procedido a la realización de diferentes experimentos en los cuales se ha pretendido considerar la idoneidad y desempeño de estos, así como de las técnicas de aprendizaje no supervisado previamente aplicadas.

Habiendo partido del ensayo individual un modelo de regresión lineal – el cual se ha reproducido sin aplicar análisis de componentes, posteriormente aplicando PCA así como ICA y, finalmente, la combinación de ambos – se han obtenido resultados por inferiores a lo esperado. Dichos resultados – apreciables en la Ilustración 7.14 y que obtienen un coeficiente de correlación – no sostienen que este algoritmo sea estimado – al menos sin procesamiento adicional – como un modelo predictivo robusto y confiable. De hecho, el índice de correlación no aumenta sino que disminuye en el caso de aplicación anterior de PCA e ICA frente a la aplicación directa del conjunto de datos, una vez normalizado.

Imagen que contiene mapa, tabla, agua, hombre

Descripción generada automáticamente

Ilustración 7.14. Representación gráfica de valores predichos por el modelo de regresión lineal ‘LinearRegression()’ de Scikit Learn frente a valores reales de la variable ‘\_id’. Fuente: elaboración propia.

Por un lado, cabe suponer que difícilmente el modelo pueda aproximarse al valor 1 debido a que los datos considerados, en ningún caso explican la totalidad de las casuísticas que influyen en el comportamiento de los usuarios (festivos, cortes de tráfico, eventos sociales y deportivos, restricciones de tráfico por Madrid Central, etc). Por otro, el hecho de que se obtenga peor índice de correlación al aplicar PCA, indica que es posible que el modelo de regresión lineal no responda adecuadamente ante posibles correlaciones lineales en la dinámica del funcionamiento del sistema.

Debido a esto, se han considerado y ensayado diversos modelos de regresión de distinta naturaleza – tanto lineales como no lineales- con fin de medir la respuesta de cada uno frente al conjunto de datos en cuestión (Anexo VIII: Código Python empleado en la selección de modelo (sin PCA). En primer lugar, no se han detallado parámetros que definen la configuración de cada uno de ellos – más allá de los considerados por defecto en la librería *Scikit Learn* – salvo alguna excepción con fin de hacer posible la realización de la prueba. En aras del rigor y para evitar el sobreajuste de modelos frente al modelo entrenado, se ha medido el coeficiente de determinación como el promedio de los cinco obtenidos aplicando validación cruzada, en cinco subconjuntos, a través de la función ‘cross\_val\_score()’.

Se han considerado y probado los siguientes modelos y algoritmos de aprendizaje supervisado:

* Regresión lineal [37], a través del algoritmo ‘LinearRegression()’ de la librería *Scikit Learn*, cuya documentación se encuentra disponible en el sitio web [51].
* Regresión logística [38], a través del algoritmo ‘LogisticRegression()’ de la librería *Scikit Learn*.
* Regresión de arista [78], a través del algoritmo ‘RidgeRegression()’ así como su extensión probabilística ‘BayesianRidge()’, ambos en la librería *Scikit Learn.*
* Regresión con el método Lasso [79], a través del algoritmo ‘Lasso()’ de la librería *Scikit Learn.*
* Combinación de las dos técnicas anteriores a través del método de «red elástica» [80], a través del algoritmo ‘*ElasticNetCV()*’ de la librería *Scikit Learn.*
* Regresión con el «algoritmo pasivo agresivo» [81], a través de la función ‘PassiveAggressiveRegressor()’ de la libería *Scikit Learn.*
* Algoritmos de regresión basados en máquinas de soporte vectorial SVR [41], a través del algoritmo ‘*SVR()*’ de la librería *Scikit Learn.*
* A través de la técnica de «*los K vecinos más próximos*» [43], a través del algoritmo ‘*KNeighborsRegressor()*’ disponible en la librería *Scikit Learn*.
* Algoritmos de aprendizaje con árboles de decisión [44] como son la regresión con bosques aleatorios [82], árboles extremadamente aleatorios [83] y potenciación del gradiente en árboles de decisión [84], a través de ‘RandomForestRegressor()’, ‘ExtraTreesRegressor()’ y ‘GradientBoostingRegressor()’ respectivamente, disponibles en la librería *Scikit Learn.*
* Modelo de regresión logarítmico como el de proceso de Gauss [85], través de ‘GaussianProcessRegressor()’ de la librería *Scikit Learn.*

Tras ensayar y probar, de forma iterativa, cada una de técnicas de minería de datos, los resultados preliminares son mostrados en la Ilustración 7.15 – para el caso de aplicación directa, sin reducción dimensional previa. De esta forma, destacan como candidatos a modelos más idóneos – de forma concreta para este conjunto de datos – los árboles extremadamente aleatorios (), y el bosque aleatorio ().

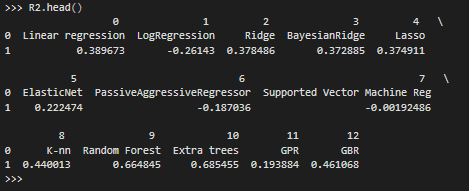


Ilustración 7.15. Captura de pantalla del resultado del proceso iterativo de evaluación inicial de diferentes modelos ante el conjunto de datos sin reducción dimensional previa. Fuente: elaboración propia a partir de líneas de código inspiradas en la documentación disponible en scikit-learn.org [86].

Del coeficiente de determinación se espera un valor comprendido entre 0 y 1, siendo este último el ajuste absoluto del modelo. Sin embargo, en la Ilustración 7.15 se aprecian no solo valores positivos menores o iguales a 1, sino que también aparecen coeficientes con signo negativo. Esta particularidad queda reflejada en la documentación del paquete ‘*metrics*’ de la librería *Scikit Learn’*, donde se explicita que los valores negativos indican que el modelo ensayado responde aún peor que una entrada de datos aleatoria.

A continuación, se han analizado aquellos modelos con mejor coeficiente de determinación, visualizando los respectivos histogramas de error absoluto (Ecuación 7.4) que se encuentran disponibles en la Ilustración 7.16. Estos reflejan la respuesta de cada modelo a los 162 de 809 registros aleatorios que forman el subconjunto de evaluación, el cual no ha participado en el desarrollo del modelo predictivo.

Ecuación 7.4. Definición de error absoluto como la diferencia entre el valor predicho por el modelo y el valor realmente registrado. Fuente: The Data Science Design Manual [87].

Imagen que contiene barco, tabla

Descripción generada automáticamente

Ilustración 7.16. Histogramas de valores de error absoluto registrados en los diferentes modelos. Fuente: elaboración propia.

De forma ideal, un modelo debe presentar una distribución de errores absolutos aguda, centrada en cero y sin registros dispersos en los extremos. De esta forma, la concentración de un elevado número de registros próximos a cero es señal de un modelo que muestra exactitud y precisión. En este sentido, aunque todos los modelos representados en la Ilustración 7.16 tienden a registrar valores de error absoluto próximos a cero, parece que el bosque aleatorio, los árboles extremadamente aleatorios e, incluso la regresión logarítmica, responden mejor que otros como la regresión lineal, los k-vecinos más próximos y el modelo de potenciación del gradiente en árboles de decisión. En términos de precisión destacan los modelos de árboles extremadamente aleatorias y, en especial, el bosque aleatorio. En este último se aprecia que los valores de frecuencia máxima de error absoluto registrado, se encuentra ligeramente desplazado a la izquierda del cero, por lo que el modelo mantiene cierta tendencia a predecir valores ligeramente menores a lo debido. Este desplazamiento, así como otros comportamientos no deseables, serán abordados en la sección posterior de ajuste de parámetros propios del modelo predictivo seleccionado.

Al repetir estos experimentos, aplicando esta vez reducción dimensional mediante análisis de componentes principales (PCA), se aprecia una degradación en los índices de determinación de los modelos ensayados (Ilustración 7.17), así como de su respuesta en términos de error absoluto (Ilustración 7.18).

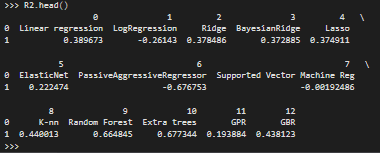


Ilustración 7.17. Captura de pantalla del resultado del proceso iterativo de evaluación inicial de diferentes modelos ante el conjunto de datos tras aplicar reducción dimensional empleando las nueve componentes principales. Fuente: elaboración propia a partir de líneas de código inspiradas en la documentación disponible en Scikit-learn.org [86]

Imagen que contiene tabla

Descripción generada automáticamente

Ilustración 7.18. Histogramas de valores de error absoluto registrados en los diferentes modelos tras la aplicación de PCA a través del uso de las nueve componentes principales. Fuente: elaboración propia.

Por último, en las siguientes imágenes se aprecia el resultado de la instrucción ‘*.describe()*’, la cual permite explorar algunos estadísticos de referencia – cuenta, media, mínimo, máximo, primer cuartil, segundo cuartil o mediana, tercer cuartil y cuarto cuartil o valor máximo registrado – del conjunto de datos predicho por cada modelo (Ilustración 7.19) así como de los errores absolutos registrados por la regresión líneal, regresión por método Lasso, K-vecinos más próximas, árboles extremadamente aleatorios, bosque aleatorio y potenciación del gradiente en árboles de decisión (Ilustración 7.20).

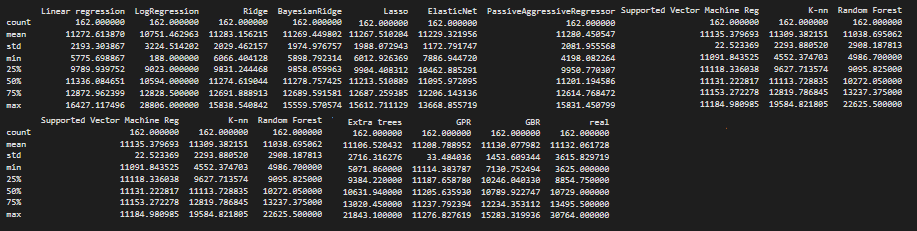


Ilustración 7.19. Captura de pantalla del resultado de la función ‘.describe()’ que muestra la cuenta, media, desviación estándar, mínimo, máximo, así como primer, segundo y tercer cuartil de los valores predichos por cada modelo así como los valores reales de la variable objetivo en el subconjunto de los registros evaluados. Fuente: elaboración propia.

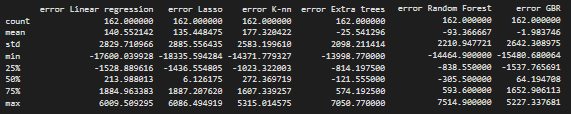


Ilustración 7.20. Captura de pantalla del resultado de la función ‘.describe()’ que muestra la cuenta, media, desviación estándar, mínimo, máximo así como primer, segundo y tercer cuartil de los valores predichos por cada modelo así como los valores reales de la variable objetivo en el subconjunto de los registros evaluados. Fuente: elaboración propia.

A vista de los resultados obtenidos, se decide continuar con la definición de parámetros de los modelos de árboles extremadamente aleatorios y de bosque aleatorio.

## 7.4 Definición de parámetros

Una vez concluido un análisis comparativo entre distintos modelos, de los cuales dos han sido seleccionados para su desarrollo, procede continuar con la definición de los parámetros – o hiper-parámetros – que establecerán características relevantes en el modelo predictivo que será entrenado y validado posteriormente.

Para ello, se ha hecho uso de la técnica de búsqueda en cuadrícula («*Grid Search*» en inglés). Este, integrada con la técnica de validación cruzada (*K-folds)* en la función ‘*GridSearchCV()’* de *Scikit Learn*. Esta instrucción permite evaluar el modelo indicado probando cada una de las combinaciones posibles entre los valores de los parámetros que sugiera el programador. Para los modelos de regresión, esta búsqueda secuencial trata de optimizar - dentro de los valores propuestos – la calidad del modelo en base a criterios como el error cuadrático medio (*MSE* por sus siglas en inglés) o error absoluto medio (*MAE* en inglés).

Antes de proceder con esta búsqueda o de tomar alguna decisión en relación a estos hiper-parámetros, resulta primordial atender en detalle al funcionamiento y lógica de los modelos seleccionados. En este caso, tanto los árboles extremadamente aleatorios como el bosque aleatorio comparten gran parte de principios y lógica al estar basados en árboles de decisión.

El aprendizaje mediante los árboles de decisión consiste en la clasificación de los valores de una variable objetivo con base en los atributos que se contienen en el mismo registro. Esta clasificación, frecuentemente representada de forma gráfica como en la Ilustración 7.21, se forma a partir de nodos (cuadriculas coloreadas), unidos mediante flechas, y que, alternándose, forman las denominadas ramas. De esta forma, se suceden procesos dicotómicos hasta caracterizar un grupo de registros, en función los nodos recorridos, que de forma deseable serán homogéneos en cuanto al valor de la variable objetivo.

En el árbol de clasificación mostrado en la Ilustración 7.21, se muestra la clasificación del conjunto de datos *flor iris* [88], en el cual se pretende la clasificación e identificación de la especie (‘*setosa’, ‘virginica’,* o ‘*versicolor’)* de diferentes ejemplares, acompañados de características morfológicas, que ha sido tomadas de forma experimental y registradas en diferentes variables. Cada nodo – representado con una caja – contiene información en relación con la variable y límite establecido, la heterogeneidad entre los registros considerados – empleando el índice Gini [89]–, tamaño de la muestra que se considera (‘*sample’*), número de ejemplares de cada especie (‘*value’*), y aquella especie predominante en la muestra considerada (‘*class’*). Se denomina nodo padre aquel del que parte una clasificación en base a una condición lógica, resultando en dos nodos hijos: a la izquierda aquel que cumple la condición (‘*True’*) *y* la derecha el que no (‘*False’*). De igual forma, el nodo del cual parte el árbol se denomina raíz, y aquellos en los cuales finaliza – hijo de un nodo del que procede, pero que no vuelve a dividirse en otros nodos – se denominan hojas.

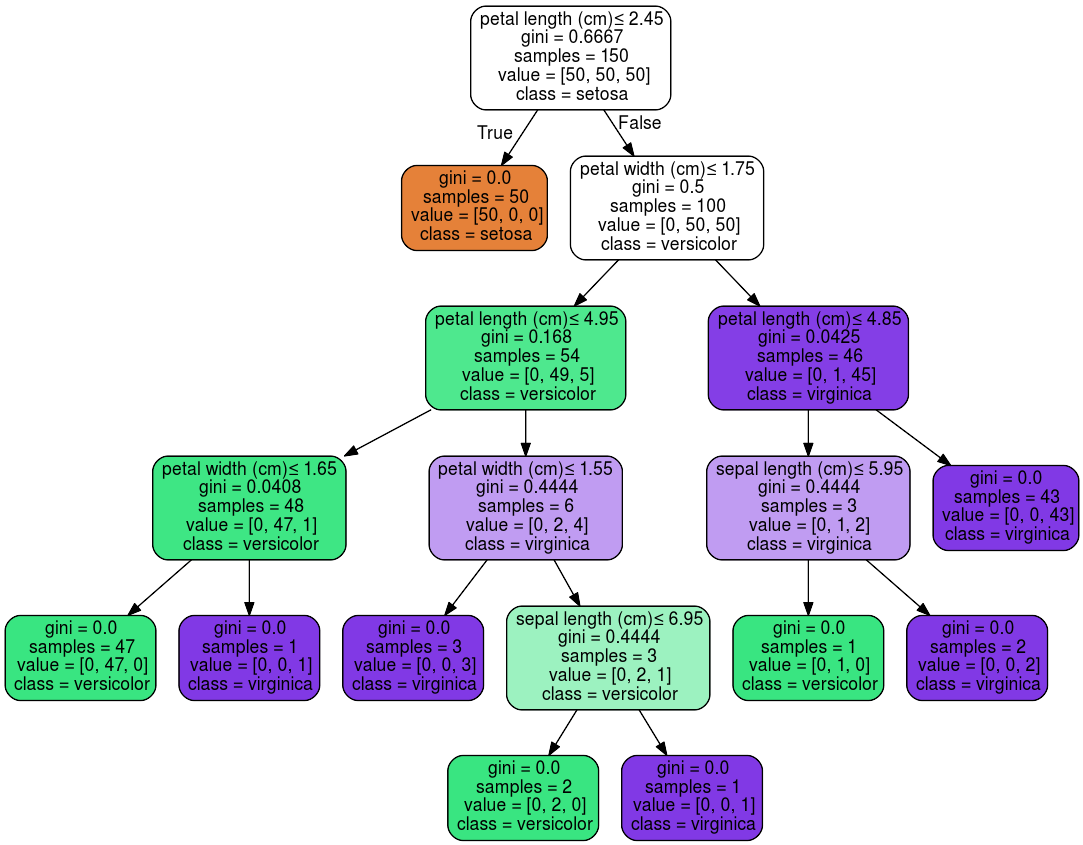
* 

Ilustración 7.21. Ejemplo de árbol de clasificación aplicado en el conjunto de datos flor iris, empleando funciones de la librería Skearn. Fuente: scikit-learn.org [90].

De esta forma, empezando por la parte superior de la Ilustración 7.21, se toma una muestra inicial de 150 ejemplares, de los cuales 50 registran una longitud de pétalo (‘*petal length’*) menor o igual a 2,45 centímetros – especímenes ‘*setosa’* en su totalidad y, en consecuencia, con índice Gini 0,0. Por tanto, el modelo concluye que todo ejemplar con longitud de pétalo mayor a 2,45cm será de especie ‘*setosa’*. Por otro lado, los 100 ejemplares que no cumplen dicha cualidad quedan agrupados en un nuevo nodo, en el cual se establece como nueva condición: que el ancho del pétalo sea inferior o igual a 1,75 cm. Este proceso se sucede hasta obtener muestras homogéneas en cada uno de los nodos finales – como sucede en el ejemplo mostrado – o en base a alguno de los siguientes parámetros, expresados en la nomenclatura empleada por *Scikit Learn*:

* ‘max\_depth’ o profundidad máxima: establece el número máximo de niveles o generaciones de nodos que se permite alcanzar en el desarrollo del árbol.
* ‘min\_samples\_split’ o muestra mínima para la división: define un mínimo de registros contenidos en un nodo para que se establezca una nueva segregación.
* ‘*min\_samples\_leaf*‘ o muestra mínima para la definición de un nodo: no se permite el establecimiento de un nodo a partir de una muestra de registros menor que el valor de este hiper-parámetro. La diferencia entre este y el parámetro anterior es que la muestra mínima para la división no atiende al tamaño de los nodos resultantes, aspecto que si recoge la muestra mínima para la definición del nodo.
* ‘min\_impurity\_decrease’ o reducción mínima de la impureza o heterogeneidad: reducción de impureza mínima esperada, en términos relativos o absolutos del hiper-parámetro ‘criterion’ – la contribución de cada nodo suele cuantificarse a través de indicadores como índice Gini en los árboles de clasificación y error cuadrático medio o error absoluto medio en los árboles de regresión. El algoritmo se desarrolla y divide, tendiendo a la reducción del indicador considerado.
* ‘max\_features’ o número de características: hace referencia al número de atributos considerados en cada clasificación. En el caso mostrado en la Ilustración 7.21 solo ha participado una variable en cada condición lógica impuesta. Sin embargo, este parámetro permite regular el número máximo de variables consideradas, de forma simultánea, en el criterio empleado para la separación de cada nodo.

De forma análoga, los árboles de regresión realizan predicciones con base en el promedio los valores contenidos en las hojas. De esta forma, en el desarrollo del árbol – durante el entrenamiento del modelo – se forman las ramas siguiendo un principio de reducción del error marginal medio– promedio de las diferencias entre valores reales y predichos en cada nodo-. Un ejemplo de árbol de regresión se muestra en la Ilustración 7.22, en el cual se pretende obtener el precio de una vivienda (‘*value’)* a partir de atributos expresados en 14 variables (). De esta forma, los procesos de clasificación dicotómica parten de un nodo con 506 registros, que van clasificándose en base a condiciones lógicas que maximizan la reducción del error cuadrático medio (‘*mse’*). Esto sucede de forma sucesiva hasta alcanzar la hoja, donde se toma como valor predicho el promedio de valores que la variable objetivo anota en los registros ahí contenidos.

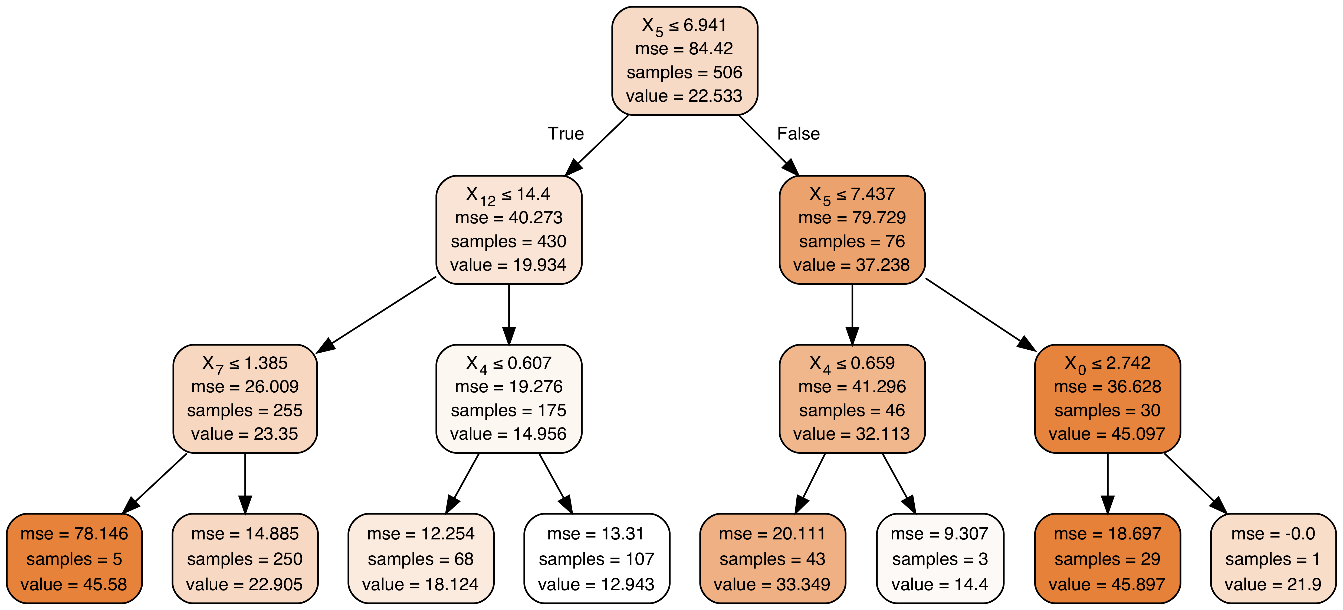


Ilustración 7.22. Ejemplo de árbol de regresión aplicado a un conjunto de datos de precios de la vivienda en Boston [91]. Fuente: recurso obtenido en explained.ai [92].

De esta forma quedaría descrito el funcionamiento general de los árboles de aprendizaje, elemento común a la regresión por árboles extremadamente aleatorios y a través de bosque aleatorio. Aunque ambos se basan en este mismo algoritmo, su funcionamiento comprende el desarrollo una multitud de árboles diferentes.

Ambos modelos consisten en el desarrollo simultáneo de diferentes áboles – en *Scikit Learn*, el número de árboles queda definido en el hiper-parámetro ‘*n\_estimator’* -, para cada uno de los cuales se toma como muestra un subconjunto aleatorio de los registros disponibles – en *Scikit Learn* se habilita a través del hiper-parámetro ‘*bootstrap’-.* Aunque en todos los árboles se suele considerar todos los atributos disponibles (existen variantes del método), en cada nodo se selecciona un número inferior o una proporción de ellos – de nuevo en Scikit Learn, puede ser definido con el parámetro ‘*max\_features’*-.

Ante una multitud de árboles que divergen en su desarrollo, suelen aparecer resultados discrepantes entre ellos. Por esto, en los clasificadores basados en árboles extremadamente aleatorios y en bosques aleatorios, toman como valor predicho – en caso de una variable objetivo de tipo categórica – la respuesta más frecuente entre las respuestas de todos los árboles. En el caso de los modelos de regresión, el valor predicho resulta como promedio de las estimaciones obtenidas localmente por cada uno de los árboles. Una de las grandes fortalezas de estos modelos radica en la disminución de la varianza a través del carácter probabilístico que se obtiene con este este tipo de estimaciones paralelas [93].

La diferencia principal entre los árboles extremadamente aleatorios y los bosques aleatorios reside en cómo se definen las variables y los valores – o límites – sobre las cuales, en cada nodo, se produce la división en dos nodos hijos. Mientras que el bosque aleatorio establece este límite estableciendo límites en atributos y valores que maximicen la caída del indicador de heterogeneidad - como índice Gini o error cuadrático medio-, los árboles extremadamente aleatorios establecen dichos límites de forma aleatoria [83]. En consecuencia, en los árboles que forman un bosque aleatorio siempre podrá observarse una caída progresiva del índice considerado, mientras que en los árboles extremadamente aleatorios no siempre tiene por qué ser así, salvo que se impongan restricciones como la reducción mínima de impureza.

La similitud entre estos dos modelos implica que sus principales hiper-parámetros sean comunes, tal y como consta en la correspondiente documentación en *SciKit Learn*, cuyo listado pueden apreciarse en la Tabla 7.2.

|  |  |
| --- | --- |
| **RandomForestReressor()** | **ExtraTreeRegressor()** |
| *n\_estimators* | *n\_estimators* |
| *criterion* | *criterion* |
| *max\_depth* | *max\_depth* |
| *min\_samples\_split* | *min\_samples\_split* |
| *min\_samples\_leaf* | *min\_samples\_leaf* |
| *min\_weight\_fraction\_leaf* | *min\_weight\_fraction\_leaf* |
| *max\_features* | *max\_features* |
| *max\_leaf\_nodes* | *max\_leaf\_nodes* |
| *min\_impurity\_decrease* | *min\_impurity\_decrease* |
| *min\_impurity\_split* | *min\_impurity\_split* |
| *bootstrap* | *bootstrap* |
| *oob\_score* | *oob\_score* |
| *n\_jobs* | *n\_jobs* |
| *random\_state* | *random\_state* |
| *verbose* | *verbose* |
| *warm\_start* | *warm\_start* |
| *ccp\_alpha* | *ccp\_alpha* |
| *max\_samples* | *max\_samples* |

Tabla 7.2. Hiper-parámetros definidos en la documentación de SciKit Learn para los modelos de bosque aleatorio (izquierda) y árboles extremadamente aleatorios (derecha). Fuente: elaboración propia a partir de la información disponible en la documentación de SciKit Learn[94] [95].

De los parámetros referidos, se procede a la explicación de aquellos no han sido explicados anteriormente:

* ‘*min\_weigh\_fraction\_leaf’*: representa el peso mínimo, en relación al total de registros, necesario para la formación de un nodo final u hoja.
* ‘*max\_leaf\_nodes’*: máximo de nodos finales u hojas permitidas en cada árbol.
* ‘*min\_impurity\_split*’: impureza minima (en términos de índice Gini, error cuadrático medio, o cualquier otro que se haya referido en el hiper-parámetro ‘*criterion’*) para la definición de un nodo final u hoja. De encontrarse por encima de este límite, volverá a clasificarse en dos nuevos nodos – salvo que contravenga el criterio establecido en ‘*min\_impurity\_decrease’*.
* ‘*oob\_score’*: habilita el cálculo del índice de determinación mediante el método ‘*out-of-bag*’ [82].
* ‘*n\_jobs’*: permite el procesamiento paralelo de las intrucciones de entrenamiento del modelo y predicción de valores, entre otras. La posibilidad de establecer procesos concurrentes se ve condicionada a la arquitectura del máquina con la que se interprete el código. De esta forma, de determinarse el valor ‘1’, solo se llevará a cabo un proceso, ‘2’ dos de forma paralela y así de forma sucesiva. El registro ‘-1’ hace referencia al uso de toda la capacidad disponible en el equipo empleado.
* ‘random\_state’: es empleado como insumo del generador de números aleatorios del cual se vale el algoritmo. Dicho generador proporciona aleatoriedad en las funciones de redimensionamiento de muestras tomadas en cada nodo y/o árbol, atributos considerados en cada nodo y, en en el caso de los árboles extremadamente aleatorios, la determinación de las variables y los valores donde se establecen los límites de cada nodo.
* ‘verbose’: número entero que determina el nivel de detalle mostrado en consola durante la ejecución de las instrucciones de entrenamiento y de predicción. A mayor valor, mayor cantidad de información expresada.
* ‘warm\_start’: de habilitarse, evita duplicar cálculos reutilizando información previamente calculada en árboles anteriores. De lo contrario, la reiteración de cálculos y desarrollos idénticos se realiza sin considerar si ha sido previamente ejcutado. Cada árbol se calcula independientemente al resto.
* *‘ccp\_alpha’*: establece un coste de utilidad para palear el exceso de complejidad – y sobreajuste – en los árboles. Permite priorizar los nodos en caso de que deban ser desechados en un proceso de simplificación conocido como poda (o por el término ‘pruning’ en inglés).
* ‘*max\_samples’*: si la función ‘*bootstrap’* queda habilitada, establece un maximo en el número de registros considerados en cada árbol – en el caso de bosque aleatorio – o en cada nodo – árboles extremadamente aleatorios.

En los casos donde pudieran producirse contradicciones entre límites y requisitos configurados a través de los hiperparámetros, las limitaciones serán dispuestas de acuerdo a los explicitados en la documentación de *SciKit Learn.*

Debido al elevado número de hiper-parámetros a considerar, las innumerables combinaciones y posibilidades que se presentan, así como la limitación en los recursos computacionales con los que se cuenta en la elaboración del presente Trabajo Fin de Grado, se hace irrealizable una búsqueda exhaustiva – a través de *‘GridSearchCV()’*- con los que obtener una configuración óptima del modelo. En su lugar, se propone una búsqueda secuencial entre un número limitado de posibilidades, así como una justificación de los valores que han sido directamente asignados. Una definición adecuada y equilibrada de los hiper-parámetros resulta esencial en la optimización del rendimiento del modelo, así como para evitar el sobre ajuste del mismo.

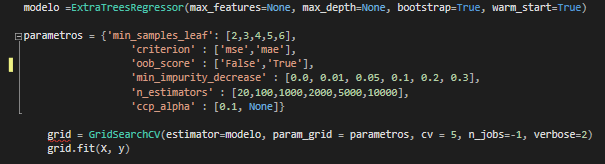


Ilustración 7.23. Captura de pantalla del código donde se establecen los hiper-parámetros sobre los cuales se establece la búsqueda secuenciar con la función GridSearchCV() para el modelo de árboles extremadamente aleatorios. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96]

El código empleado en la búsqueda secuencial en el modelo de árboles extremadamente aleatorios se aprecia en la Ilustración 7.23. El código mostrado consta de cuatro líneas principales:

* Definición de la variable ‘modelo’, donde se define la función ‘ExtraTreeRegressor()’ – correspondiente a la regresión mediante árboles extremadamente aleatorios – así como varios hiper-parámetros cuyos valores han sido fijados a partir de la siguientes razones:
  + ‘*max\_features = None’:* No se establece ningún límite de atributos a considerar en cada nodo – más allá del número de atributos disponibles en el conjunto de datos-. El carácter aleatorio en la selección de atributos potencia la diversidad entre diferentes árboles al igual que previene al modelo de sobreajuste. Fijando el valor ‘*None’* se evita limitaciones en dicho proceso de selección.
  + ‘*max\_depth=None’*: el establecimiento de árboles excesivamente desarrollados puede ser causa de sobreajuste, aunque una limitación taxativa del número de niveles desde la raíz hasta las hojas puede ser impedimento para la correcta definición de algunos patrones de comportamiento. Por esto, valores ensayados en parámetros como ‘*min\_samples\_split’* y ‘*min\_impurity\_decrease’* resultan útiles en la contención del sobreajuste del modelo. El uso de validación cruzada por iteraciones también previene al modelo de un posible sobreajuste, debido a que el rendimiento del modelo es medido como promedio de los cinco obtenidos en cada iteración.
  + ‘*bootstrap = True’*: establece el reajuste aleatorio del tamaño y de los registros seleccionados en cada nodo.
  + ‘*warm\_strart = True*’: evitar la duplicidad de cálculos idénticos permite optimizar los recursos computacionales disponibles, favoreciendo un menor tiempo de ejecución del código.
* Declaración del objeto ‘*parametros’*, donde se definen los hiper-parámetros a determinar acompañados de la lista de valores que se ensayan en cada uno de ellos. La limitación en el número de opciones a probar motiva que, en algunos casos, se haya optado por valores representativos de diferentes órdenes de magnitud más que de valores concretos.
  + ‘*min\_samples\_leaf’:* se establecen como mínimo de 2 a 6 registros para que un nodo pueda ser constituido como hoja.
  + ‘*criterion*’: se prueban error cuadrático medio y error absoluto medio como indicadores de impureza, aspecto de especial relevancia si existen requisitos en la reducción de impureza en cada nodo.
  + ‘*oob\_score’*: se ensaya la configuración de parámetros habilitando y deshabilitando el cálculo del error mediante el método ‘*out-of-bag*’.
  + ‘*min\_impurity\_decrease’*: se restringe el mínimo detrimento de la impureza – en términos del indicador establecido en ‘*criterion’ –* para la división de un nodo en los dos siguientes. Se ensayan valores mínimos que van en caídas relativas desde el 0% hasta el 30%.
  + *‘n\_estimators’*: se estima la calidad del modelo con diferentes órdenes de magnitud en el número de árboles a desarrollar. Previsiblemente, un mayor número de árboles supone un beneficio potencial en cuanto a la precisión y a la reducción del sesgo estadístico del modelo. Sin embargo, supone a su vez un mayor consumo de recursos computacionales, por lo que debe procurarse un equilibrio entre el número de árboles y el beneficio producido en cuanto a la calidad del modelo.
  + *‘ccp\_alpha’*: se evalúa la imputación de algún coste a la complejidad del modelo frente a ninguno.

Tras concluir la búsqueda secuencial con la función ‘*GridSearchCV()’*, el código mostrado en la Ilustración 7.24 proporciona la configuración que resulta la mejor entre todas las ensayadas. Se ha obtenido un índice de correlación de – aplicando validación cruzada por ‘*K-folds’* en el subconjunto de entrenamiento y evaluación – asignando los siguientes valores a los hiper-parámetros que se detallan a continuación:

* ‘*ccp\_alpha’*: 0,1
* ‘criterion’: ‘*mse’*
* *‘min\_impurity\_decrease’*: 0,3
* *‘min\_samples\_leaf’*: *2*
* *‘n\_estimators’*: *100*
* *‘oob\_score’* : *‘False’*

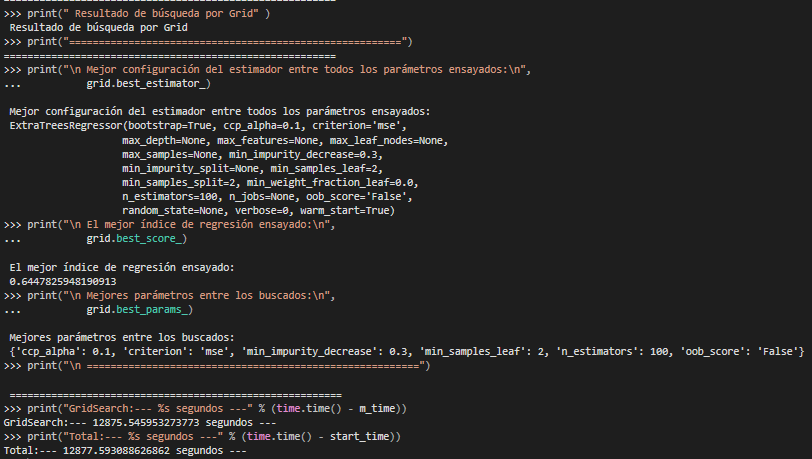


Ilustración 7.24. Captura de pantalla que muestra los resultados de la búsqueda secuencial mostrada en la Ilustración 7.23. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96].

En cuanto al modelo de bosque aleatorio – de forma similar al anterior – se ha procedido realizando una nueva búsqueda secuencial de los hiper-parámetros a través de la función ‘*GridSearchCV()’*. Tal y como se aprecia en la Ilustración 7.25, parte de los hiper-parámetros ensayados difieren de los anteriores probados en la búsqueda secuencial para el modelo de árboles extremadamente aleatorios:

* En el parámetro ‘*bootstrap’*, se ha probado habilitando *(‘True’)* y deshabilitando (‘*False’)* el método homónimo, el cual considera en cada árbol un subconjunto de registros entre todos los disponibles*.*
* En cuanto al número máximo de atributos considerados en cada clasificación en nodos (‘*max\_features’*), debido a que este modelo no establece atributos y límites aleatorios en la clasificación, sino que los selecciona optimizando la caída del índice de heterogeneidad; se han ensayado los siguientes valores: ‘*auto’* (sin restricciones), 7, 10 y ‘*sqrt’* (raíz cuadrada del número de atributos disponibles).

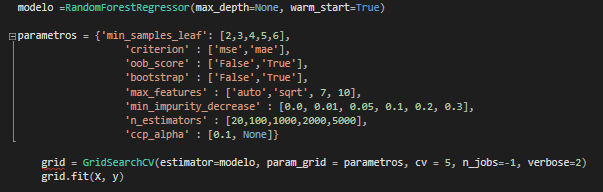


Ilustración 7.25. Captura de pantalla del código donde se establecen los hiper-parámetros sobre los cuales se realiza la búsqueda secuencial con la función GridSearchCV() para el modelo de bosque aleatorio. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96]

De esta búsqueda secuencial resultan, habiendo obtenido un índice de determinación , se ha concluido una configuración de parámetros expuesta en la Imagen 6.26:

* ‘*bootstrap*’: ‘*False*’
* ‘*ccp\_alpha’*: ‘*0.1’*
* ‘*criterion*’ : ‘*mse*’
* ‘*max\_features*’: ‘*10*’
* ‘*min\_impurity\_decrease*’: ‘*0,3*’
* ‘*min\_samples\_leaf’*: ‘*3’*
* *‘n\_estimators’*: *‘100’*
* ‘*oob\_score’*: *‘False’*



Ilustración 7.26. Captura de pantalla que muestra los resultados de la búsqueda secuencial mostrada en la Imagen 6.25. Fuente: elaboración propia a partir de líneas inspiradas en código de N.Halder [96]

## 7.5 Validación de árboles extremadamente aleatorios y bosque aleatorio

Concluida la definición de los hiper-parámetros en ambos modelos, procede la validación de los mismos. Corresponde a esta fase la obtención del rendimiento efectivo de los modelos obtenidos. Resultados anteriores, como el coeficiente de determinación obtenido, no deben ser considerados como definitivos debido a es previsible que exista cierto sobreajuste respecto a los datos empleados en su desarrollo – aunque para ello se haya empleado validación cruzada. Con este fin, se mide el desempeño del modelo empleando el subconjunto de registros reservado para su validación, cuyo volumen supone un 20% del total (162 registros).

En este punto cabe destacar que, en comparación con la fase correspondiente a la selección entre otros modelos, se ha contado con un número sustancialmente menor de registros. Tal y como se esquematiza en la Ilustración 7.27, el entrenamiento y evaluación del modelo se ha llevado a cabo empleando una porción del total de registros disponibles – correspondiente al 80%-. Sobre el mismo se ha desarrollado de nuevo la validación cruzada (*K-fold* con K=5) necesaria durante la búsqueda secuencial de parámetros. Esta última se ve representada en la Ilustración 7.27 (esquina inferior derecha), donde para cada configuración de parámetros se han evaluado cinco experimentos – cada uno corresponde con una combinación de cuatro subconjuntos destinados al entrenamiento (‘A’) y uno para evaluación (‘B’)-.

Por este motivo, cabe esperar una degradación de los indicadores de ajuste previamente medidos – por ejemplo, el coeficiente de determinación – debido a que el proceso de aprendizaje ha podido verse afectado al ser expuesto a un número menor de registros que en las ocasiones anteriores. Ello atiende a razones metodológicas, lo que actúa en pos de la mitigación del sobreajuste el modelo final. Cabe destacar que *SciKit learn* establece estos subconjuntos de forma aleatoria, sin atender a ningún orden cronológico ni de ningún otro tipo.



Ilustración 7.27. Diagrama del proceso de validación cruzada y subconjuntos de registros empleados durante las fases de definición de parámetros y validación de modelos. Fuente: elaboración propia.

Una vez definidos los parámetros del modelo de árboles extremadamente aleatorios, en su validación se ha obtenido un índice de correlación (Ilustración 7.28) y un error absoluto cuyo histograma se expone en la Ilustración 7.29. Esta distribución muestra características oportunas como la concentración de registros de error absoluto próximos al cero. Sin embargo, presenta otros aspectos no deseables como la amplitud de la distribución – motivada por la presencia de valores altos de error absoluto – así como que, de nuevo, el centro de la distribución se encuentra desplazada respecto al cero. Esto evidencia una tendencia del modelo a predecir valores inferiores al valor real de número de desplazamiento registrados.

Imagen que contiene mapa, agua, hombre, blanco

Descripción generada automáticamente

Ilustración 7.28. Representación gráfica de valores predichos por el modelo de árboles extremadamente aleatorios, tras la definición de hiper-parámetros, frente a los reales del conjunto ensayado. Fuente: elaboración propia.

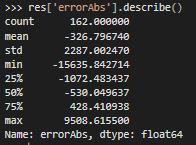
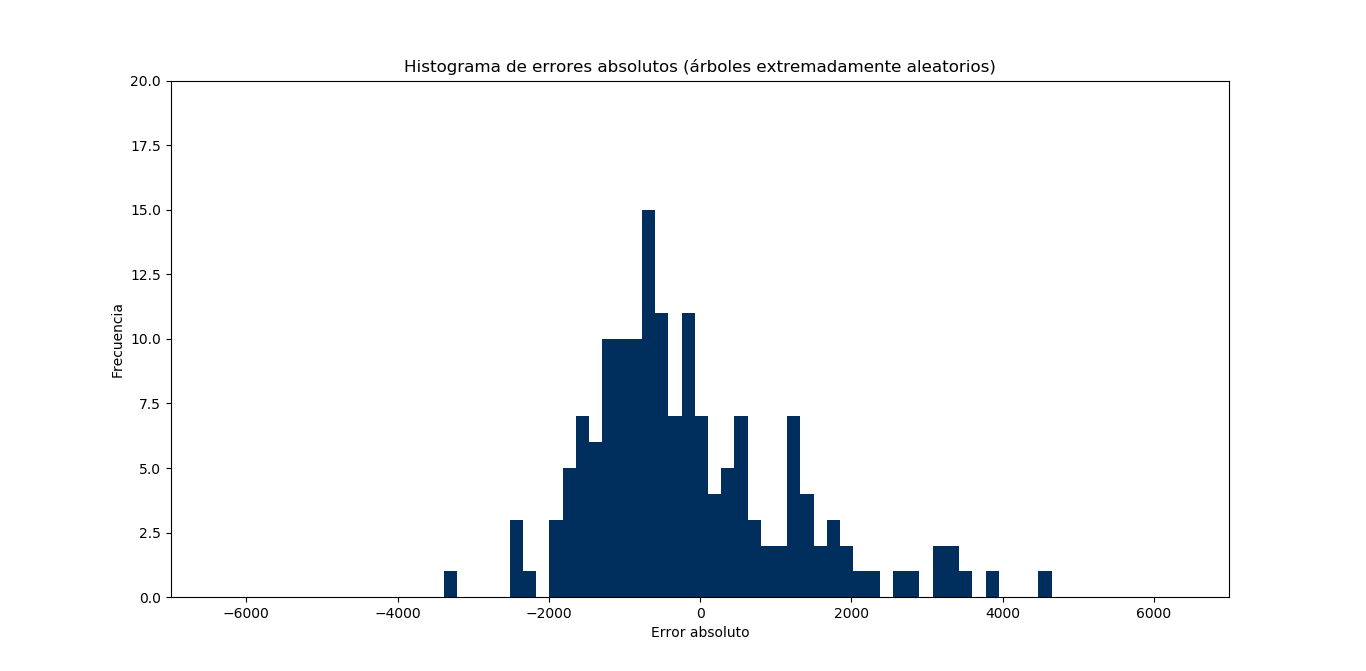


Ilustración 7.29. Distribución de registros de error absoluto obtenido por el modelo de árboles extremadamente aleatorios, tras la definición de hiper-parámetros, en su representación gráfica en forma de histograma (arriba) y cuantificada (abajo). Fuente: elaboración propia.

En cuanto al modelo de bosque aleatorio, la validación de su rendimiento ha resultado en coeficientes de determinación de alrededor de (Ilustración 7.30). Un detrimento considerable de dicho indicador experimentado durante el proceso de validación del modelo. Ello indica un posible sobreajuste del modelo que resulta del proceso de definición de parámetros. A su vez se ha observado que la calidad del ajuste ha experimentado variaciones en diferentes ejecuciones de su entrenamiento, para los mismos registros y parámetros. Dicha variabilidad puede ser resultado del valor establecido en el hiper-parámetro ‘*max\_features’*, inferior a los atributos disponibles en el conjunto de datos. El hecho de sucederse una selección aleatoria de las variables que participan en la clasificación de cada nodo, puede condicionar el desarrollo del resto del árbol. Esta inestabilidad en el coeficiente de determinación – no experimentada en el modelo de árboles extremadamente aleatorios – puede contenerse incrementando el número de árboles en el bosque entrenado, aunque ello no conlleva una mejora sustancial del ajuste logrado por el modelo. El histograma con los valores de error absoluto registrados se encuentra en la Ilustración 7.31.

Imagen que contiene mapa, tabla, hombre, agua

Descripción generada automáticamente

Ilustración 7.30. Representación gráfica de valores predichos por el modelo de bosque aleatorio, tras la definición de hiper-parámetros, frente a los reales del conjunto ensayado. Fuente: elaboración propia

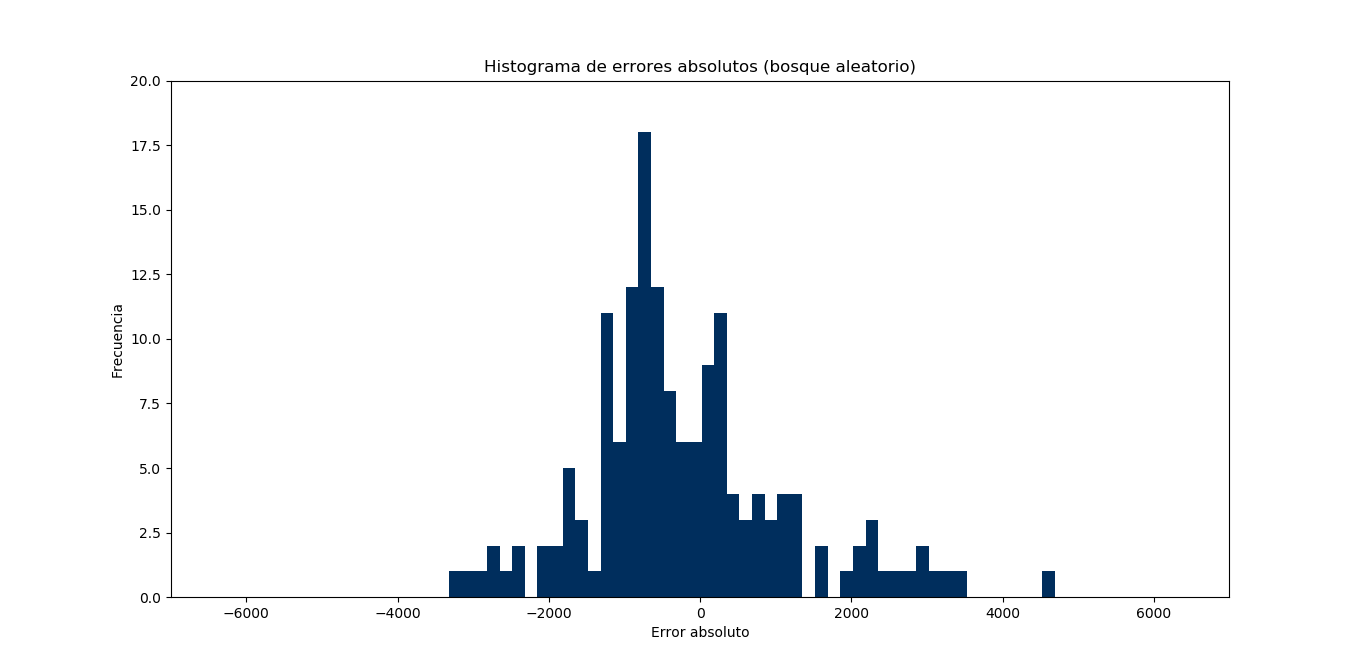


Ilustración 7.31. Distribución de registros de error absoluto obtenido por el modelo de árboles extremadamente aleatorios, tras la definición de hiper-parámetros, en su representación gráfica en forma de histograma (arriba) y cuantificada (abajo). Fuente: elaboración propia.

Estableciendo una comparativa entre ambos modelos, en la Ilustración 7.32 pueden apreciarse ciertas similitudes morfológicas entre las respectivas distribuciones de error absoluto. Este análisis se apoya en los parámetros de referencia error absoluto medio, primer, segundo (mediana) y tercer cuartil, así como la desviación estándar – mostrados en la Tabla 7.3 y se acompaña de otros indicadores de calidad del modelo como el coeficiente de determinación () y la raíz del error cuadrático medio (RECM).

A la vista de estos resultados, no resulta fácil decantarse por alguno de estos dos modelos dado que este en este caso, los árboles extremadamente aleatorios han obtenido un coeficiente de determinación muy superior al de bosque aleatorio. Sin embargo, la distribución de este último presenta mejores indicadores en términos de precisión (variabilidad) y exactitud (sesgo). Debido al coeficiente de determinación .(Ecuación 7.5) notoriamente superior, así como a un comportamiento más estable, se opta en este caso por el modelo de árboles extremadamente aleatorios.

Ecuación 7.5. Expresión que define el coeficiente de determinación .

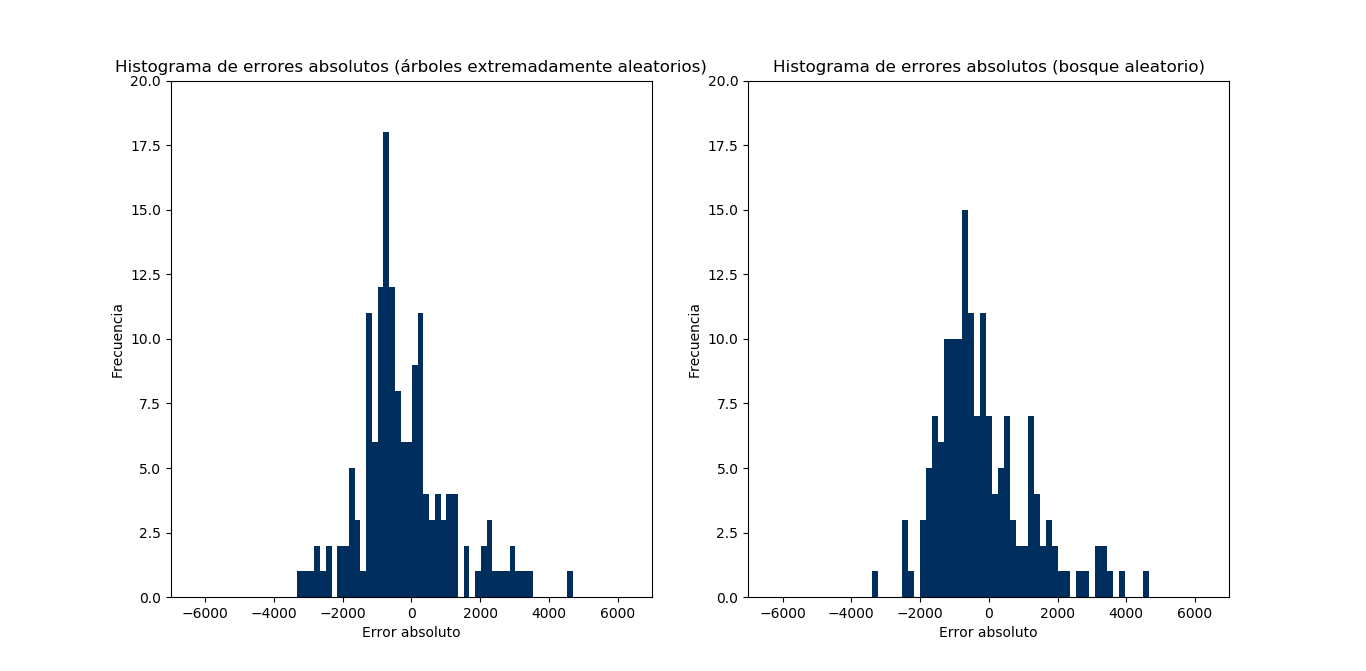
****

Ilustración 7.32. Comparativa de distribuciones de errores absolutos obtenidos por los modelos de árboles extremadamente aleatorio (izquierda) y bosque aleatorio (derecha). Fuente: elaboración propia.

****

Tabla 7.3. Comparación de indicadores significativos de la calidad de los modelos validados. De izquierda a derecha se representa el coeficiente de determinación (R^2), raíz de error cuadrático medio (RECM) y distribución del error absoluto obtenido (medio, primer segundo y tercer cuartil y desviación estándar). Fuente: Elaboración propia.

Destacan como principales puntos de mejora del modelo lo que concierne a la variabilidad y sesgo de la respuesta del mismo. La Ilustración 7.33 facilita la distinción de estos términos en la respuesta de cualquier modelo predictivo. En las cuatro dianas representadas con círculos concéntricos, es deseable que la salida del modelo – representados con puntos azules – ocupen la parte más central de las mismas. En el primer caso, comenzando por la izquierda, se trata de un modelo impreciso e inexacto, puesto que los valores predichos se encuentran muy dispersos, así como alejados del deseado. Por el contrario, en el extremo derecho se encuentra el mejor de los escenarios, representando una situación ideal en la cual el modelo responde de forma precisa y exacta por la concentración y proximidad de la salida al valor deseado. Desafortunadamente, el modelo obtenido en el presente Trabajo Fin de Grado, muestra una combinación de los representados en la parte central. El segundo por la izquierda – preciso pero inexacto – logra concentrar su respuesta en torno a un punto, aunque alejado del centro (sesgo o *bias* en inglés). El tercero por la izquierda – exacto pero impreciso – registra valores alrededor del centro pero muy dispersos unos de otros (variabilidad). En las distribuciones de error absoluto observadas (Ilustración 7.32), el sesgo corresponde con una distribución desplazada a la izquierda – cuyo error absoluto mediano se sitúa en torno a 530 registros. En este sentido, podría estudiarse la respuesta del modelo si a su salida se sumase un parámetro fijo – como el error absoluto medio o mediano – con fin de subsanar el sesgo.

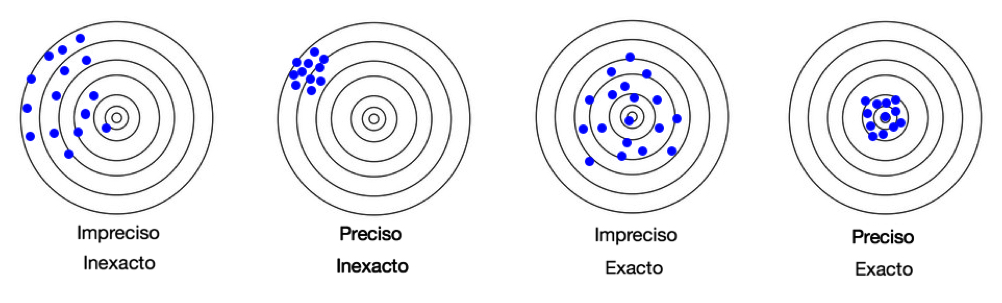


Ilustración 7.33. Diagrama explicativo de precisión y exactitud. Fuente: recurso propio de Óscar Moreno (INTEF).

En este punto, y sin menoscabo de posibles desarrollos y ciclos de mejora continua, el modelo se encontraría listo para su almacenamiento, integración y uso en los procesos de negocios que la organización interesada considere necesario (Ilustración 7.34).

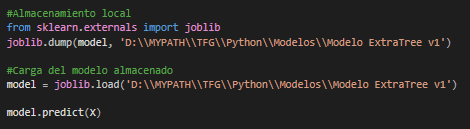


Ilustración 7.34. Captura de pantalla de líneas de códigos correspondientes al almacenamiento de un modelo una vez ha sido entrenado (parte superior), carga del mismo para su uso posterior (centro) y empleo del mismo para predecir unos valores dada una entrada ‘X’ (parte inferior). Fuente: elaboración propia a partir de la documentación disponible en Scikit-learn.org [86].

## 7.6 Líneas de trabajo para la mejora del modelo

En esta sección se presentan algunas líneas de trabajo a tener en cuenta durante el desarrollo e integración del modelo en la organización. De hecho, la organización interesada – de considerar el modelo predictivo, su continuidad e integración – recomendablemente deberá desarrollar procesos de mejora continua, haciéndose valer de metodologías como *PDCA* o *DMAIC* [97] que apoyen el proceso iterativo definido por el estándar *CRISP-DM*. Por tanto, es deseable que el modelo se encuentre en constante control, transformación y mejora. Con ello no solamente se logra refinar la respuesta del modelo a los datos de entrada, sino que también permitirá atender a cambios que pueden producirse en la dinámica de funcionamiento del servicio y de su demanda. Por ejemplo, las medidas de distanciamiento social y repercusiones en el comportamiento social tras la pandemia de *COVID-19,* afectarán de forma previsible al uso de BiciMAD y al comportamiento del usuario. Por tanto, según se obtengan más registros de uso del servicio, deben incorporarse al entrenamiento de futuras versiones del modelo. En el caso concreto que se menciona, sería conveniente la incorporación de una nueva variable – por ejemplo, de tipo booleana o categórica – que dé cuenta de un cambio significativo en la dinámica del sistema.

En cuanto a la disponibilidad de recursos computacionales, en un caso de aplicación e implementación real – o en el que pueda disponerse de mejores recursos materiales – podría realizarse una búsqueda de hiper-parámetros más exhaustiva. En ese caso, se pueden explorar alguna de las siguientes alternativas en una búsqueda de mayor capacidad computacional:

* Empleo de equipos convencionales con mayores recursos computacionales y con disponibilidad exclusiva para esta actividad.
* Distribución paralela de la carga de trabajo en diferentes equipos a través de entornos de trabajo como *Hadoop MapReduce* [98].
* Creación y uso de máquinas virtuales en grupos de servidores [99], pudiendo ser equipos propios o ajenos.
* Acceso de servicios de cálculo en la nube – tales como *Microsoft Azure* [100], *Amazon Web Services* [101], *Google Cloud* [102]o *IBM Cloud* [103], los cuales ofrecen una inmensa capacidad computacional a través de tarifas de pago por uso.

Otra de las mejoras posibles consistiría en la configuración de modelos de aprendizaje en conjunto – *‘ensemble learning’* en inglés [104] o de modelos de aprendizaje con redes neuronales [105].

Tal y como se ha comentado con anterioridad, otra de las dificultades que encuentra el modelo para lograr un mayor ajuste, es que el conjunto de datos empleados contempla información relativa a la estacionalidad y a la meteorología registrada. Sin embargo, no recoge otras muchas circunstancias y casuísticas que influyen la demanda y disponibilidad del servicio. Eventos culturales, deportivos, litúrgicos, restricciones de tráfico entre muchos posibles. En la Tabla 7.4 se muestran aquellos registros para los que, durante el proceso de validación, el modelo ha realizado una predicción que difiere en más de 3.000 desplazamientos del trayecto realizado. Atendiendo a la fecha a las que corresponden estos once registros, ocho de ellas corresponden con posibles particularidades que han podido influir en la demanda del servicio. Por ejemplo, se aprecian festivos de relevancia en Madrid en los cuales se han registrado menos desplazamientos de los predichos (día de la Hispanidad, Semana Santa, Verbena de la Paloma, Nochebuena y Nochevieja). Por otro lado, hay registros que prácticamente duplican la predicción y cuyas fechas coinciden con una de las mayores incorporaciones de estaciones y bicicletas al sistema de BiciMad [68]. Esta ampliación puede asociarse una mayor disponibilidad del servicio y aumento de la demanda derivada del incremento de la capacidad y alcance – a nuevos distritos – de la red. Debido a que junio de 2019 es el último mes del cual se tienen registros, resulta difícil evaluar si estos efectos se prolongan en el tiempo.



Tabla 7.4. Análisis de fechas en las que el modelo ha registrado una raíz de error cuadrático superior a los 3.000 trayectos (en el subconjunto de validación), acompañado de posibles casusas y efectos que han podido afectar a la demanda del servicio. Fuente: elaboración propia.

Por último, en esta misma tabla llama la atención el valor registrado el 30 de mayo de 2018 – 12 registros, lo que equivale a poco más que una milésima parte del valor predicho. No habiéndose encontrado evidencia de interrupción del servicio en tal fecha, cabe suponer que se deriva de algún problema en la generación, transferencia o transformación de los datos. Se observa que, analizando todo el conjunto de datos, solo se registran cuatro fechas con menos de 2.000 desplazamientos Tabla 7.5. A excepción de uno de ellos – 188 desplazamientos el 29 de agosto de 2018- se observa que se concentran en torno a las mismas fechas: 25 de mayo (25 registros), 29 de mayo (1) y 30 de mayo (12) de 2018. Por tanto, no cabe descartar que en ese periodo se haya podido producir alguna incidencia en la captación, almacenamiento o procesamiento de datos.



Tabla 7.5. Fecha y número de desplazamientos de los registros contenidos en el conjunto de datos con menos de 2.000 registros diarios. Fuente: elaboración propia.

Aunque las apreciaciones que se derivan de estas observaciones no resulten concluyentes, señalan la conveniencia de – en futuras versiones del modelo – incluir variables que reflejen si el periodo analizado se trata un festivo, si se esperan restricciones de tráfico, eventos multitudinarios en la ciudad, número de estaciones operativas, etc. Adicionalmente se pueden incluir atributos adicionales que procedan de fuentes de datos propias – por ejemplo, el número de trayectos registrados el mismo día del año anterior – o de terceros, mediante el uso de interfaces de programación de aplicaciones (API) o automatización robótica de procesos (RPA), que permiten abastecer al sistema de datos como la demanda eléctrica prevista en la Comunidad de Madrid [106][107]. Este dato se asocia de forma frecuente a actividad económica de un territorio [108].

# Conclusiones

A continuación, se recopilan y comparten algunas de las reflexiones emanadas de la realización del presente Trabajo Final de Grado. Las mismas atienden tanto al caso particular del modelo desarrollado – resultados obtenidos, herramientas y metodologías empleadas – así como al contexto y el papel que juega el asunto de referencia en la transformación digital de las organizaciones, la economía y la sociedad.

Con la realización del presente documento – así como de los archivos que lo acompañan – se ha pretendido documentar el proceso de desarrollo de un caso de aplicación de herramientas de minería de datos y aprendizaje automático. El alcance del Trabajo Fin de Grado, así como los recursos disponibles, han condicionado el desarrollo del modelo. Sin embargo, este puede considerarse suficiente para establecer un punto de partida para un proyecto de desarrollo de modelo predictivo y para su implantación real. A su vez, los resultados obtenidos apuntan a la validez de la hipótesis planteada, en relación a que la demanda del servicio se ve condicionada por factores ambientales como la meteorología en la ciudad de Madrid (definida en la sección «7.1 Líneas de trabajo con minería de datos»).

En cuanto al ámbito de aplicación seleccionado, se ha procurado que la propuesta pueda contribuir a un servicio de alto valor social. BiciMAD pretende dar respuesta a uno de los mayores retos de la sociedad actual: la movilidad sostenible en el transporte individual de personas. Dicho servicio participa en la electrificación del transporte individual y en el desarrollo de infraestructuras de transporte de uso compartido en la capital del país– apoyándose en las tecnologías de información – siendo un elemento a desarrollar en la evolución de las «Ciudades Inteligentes».

La transformación digital en las organizaciones, en el seno de la «Cuarta Revolución Industrial» o «Industria 4.0», introduce una consecución de cambios significativos a la cual la ingeniería de Organización Industrial no es ajena. El cambio de paradigma en muchos procesos, entre los cuales se encuentran los correspondientes a la planificación y programación de la producción de bienes y de la operación de servicios, requiere en los organizadores de nuevas competencias y habilidades. Esto ha servido de motivación en la aplicación de una serie de herramientas, cuyos detalles no se contemplan en los planes de estudios del grado al que se pretende promocionar, y que actualmente se hacen necesarios para el desarrollo profesional. Por ello, se ha querido aprovechar – a título personal – la realización de este trabajo para la investigación, aprendizaje y práctica de la minería de datos, procurando así el desarrollo de habilidades y la adquisición de competencias asociadas, ligadas al desarrollo profesional en el ámbito de la organización industrial. De hecho, el modelo propuesto está íntimamente relacionado con el campo del pronóstico de la demanda, tan necesario en las organizaciones y en la gestión de sus cadenas de suministros.

Algunas de las lecciones aprendidas durante el desarrollo del trabajo se enumeran a continuación:

* En la minería de datos y en el aprendizaje automático, el empleo de elementos que actúan a un alto nivel como cajas negras, supone un riesgo en la obtención de conocimiento. Es deseable que el empleo de estos conlleve el entendimiento de los principios de los cuales se valen para dar respuesta a los datos de entrada. De lo contrario, no solamente supondría una pérdida del control de proceso llevado a cabo, sino que puede acarrear respuestas no deseables, procesos ineficientes y generación de ruido que impida la obtención del conocimiento.
* En un caso de implantación real, tanto la identificación de necesidades como la interpretación de resultados y desarrollo del modelo, precisa de conocimiento especializado en el proceso que se analiza. Por tanto, las organizaciones que pretendan externalizar el desarrollo de soluciones de este tipo deben prestar atención al mismo en las diferentes fases del proceso. Tan importante es la validez, precisión y exactitud del modelo, como que este se encuentre en línea con las necesidades e intereses estratégicos de la organización.
* En el desarrollo del modelo predictivo de demanda del servicio BiciMAD por días, se ha atendido al comportamiento global del sistema y no por estaciones. En los casos probados, con la configuración de modelo ensayada, los resultados han sido considerablemente peores cuando dicho análisis atendía a las estaciones de forma individual.
* En relación con este último aspecto, resulta de interés la búsqueda de modelos predictivos que atiendan a la demanda por estación o por grupos de ellas – por ejemplo, por proximidad o por distritos. De esta forma, además de contar con una demanda predicha de forma individual, podría construirse la demanda global como la suma de todas las individuales. Además, permitiría atender a detalles como si la estación es deficitaria o excedentaria, así como los registros de reposición de unidades, estaciones fuera de servicio, y otros aspectos que pueden resultar cuellos de botella en el sistema.
* Los datos empleados – de acceso público en través del portal de datos abiertos de EMT Madrid [18] – omiten campos (como el identificador de usuario que realiza el desplazamiento) o agrega el contenido del mismo (marcas temporales agrupadas por día y hora). Sin embargo, el operador del servicio, con acceso a la totalidad de campos en los registros, tiene posibilidad de construir modelos predictivos que contemple dichos detalles para describir el comportamiento del sistema.
* Los registros empleados incluyen algunos factores ambientales que pueden afectar a la demanda, como la meteorología. Sin embargo, existen otras muchas casuísticas que también influyen y no se contemplan en el registro. Por tanto, cabe esperar que el ajuste obtenido sea susceptible de mejoras en este sentido. Igualmente, y a la vista de los resultados, resulta de interés la inclusión de variables adicionales que previsiblemente pueden condicionar la demanda del servicio. Por ejemplo, indicando si el día al que corresponde el registro es festivo, si se han programado eventos multitudinarios, o si se prevén restricciones de tráfico. Adicionalmente pueden incluirse otros valores como número desplazamientos registrados el mismo día del año anterior. De la misma forma, tal y como se propone en «7.6 Líneas de trabajo para la mejora del modelo», pueden incluirse otros indicadores como la demanda eléctrica prevista, por Red Eléctrica Española, para la comunidad de Madrid en ese mismo día.
* A la luz de los resultados obtenidos del análisis de componentes principales, el cual no ha mejorado el desempeño del modelo – y en consecuencia ha sido desestimado – cabe pensar que las dinámicas del sistema no muestran un comportamiento lineal en los datos. De hecho, los procesos documentados en «7.3 Selección de Modelo» muestran como los dos algoritmos de aprendizaje que mejor han respondido a la dinámica del sistema no son paramétricos, por tanto, no lineales.
* Siguientes versiones del modelo que se ha desarrollado, así como otros análisis que pueden realizarse a partir de los datos registrados, cobran una especial relevancia tras la crisis sanitaria sufrida en España a causa del COVID-19. No solamente contribuyen a la trazabilidad y seguimiento de posibles contagios relacionados con el uso del servicio, sino que se hacen necesarios para describir nuevas dinámicas de funcionamiento de la demanda de servicios de movilidad.

Durante la realización del presente Trabajo Fin de Grado se han obtenido algunas conclusiones, las cuales tienen con un carácter más genérico y que inspiran las siguientes reflexiones:

* La denominada jerarquía del conocimiento se hace cada vez más necesaria en las organizaciones. De su integración en los procesos, dependen la obtención y afianzamiento de importantes ventajas competitivas, además de realizar importantes contribuciones a la gestión del riesgo e incertidumbre, así como al análisis de costes y salud financiera de las organizaciones.
* La información, el conocimiento y la sabiduría obtenida resultan de utilidad en diferentes niveles de las organizaciones. Desde la programación de la producción y las operaciones, hasta el nivel ejecutivo y estratégico, pasando por la planificación de los recursos, se pueden hacen valer de informaciones y pronósticos del tipo que se ha desarrollado y propuesto en este trabajo.
* Esta realidad a la que se hace referencia propicia la aparición de nuevos productos, servicios y modelos de consumo, lo cual las organizaciones deberán tener en cuenta y prestar mayor atención a los procesos de vigilancia tecnológica.
* La complejidad de las técnicas empleadas, así como la importancia de la metodología a seguir, pone de relieve la importancia de la capacitación y cualificación del personal que debe desarrollarla. Por tanto, las organizaciones que quieran aplicar este tipo de procesos deben hacer importantes esfuerzos invirtiendo en formación y a atracción del talento.
* La realización de este Trabajo Fin de Grado ha sido posible a la disponibilidad de datos abiertos publicados por la empresa pública EMT Madrid. Sin embargo, las organizaciones privadas son muy recelosas con compartir o dar acceso a registros de este tipo por la ventaja competitiva que conlleva su explotación, así como que describen en buena medida la intensidad, aprovechamiento y planificación de procesos clave. Esto conlleva un agravio para aquellas organizaciones que, ya sea por el volumen de sus operaciones o por el tipo de proceso que llevan a cabo, no cuentan con el volumen de datos necesarios para lograr el aprovechamiento de los mismos.
* La interconexión de sensores en objetos que propicia «internet de las cosas», y sus correspondientes aplicaciones en procesos industriales, supone una oportunidad para las organizaciones más pequeñas en la adquisición de datos de sus propios procesos. Esta fuente de datos puede ser complementada con más datos obtenidos a través de procesos robóticos automatizados (RPA) y consultas a interfaces de aplicación (API), gratuitas o de pago por uso.
* La consideración de los proyectos de transformación digital y explotación de la información como procesos de coste indirecto, conlleva un nuevo agravio entre las organizaciones en función de su tamaño y su estructura de coste. Para la implantación de este tipo de procesos se hace necesaria una visión estratégica y una importante asignación de recursos (personal especializado, equipos informáticos, origen de datos, formación, etc). Aquellas organizaciones concentradas en lo inmediato, en la gestión del «día a día», ven impedida la implantación de estos proyectos que, presumiblemente, suponen un consumo de recursos con una amortización que no se espera a corto sino a medio y largo plazo.
* La automatización de los procesos, como de la percepción de entornos y toma de decisiones – que caracterizan a la inteligencia artificial – conlleva una reducción de carga de trabajo para las personas, que en consecuencia puede conllevar la pérdida de puestos de trabajo – aunque en cierta medida se compense con la creación de otros de alta cualificación – e intensificar la brecha digital en la sociedad. Por ello, se plantean diferentes debates y retos que la sociedad debe afrontar como transformaciones sociales del siglo XXI. Un posible régimen fiscal que atienda a los procesos robotizados, supresión de carga de trabajo para las personas a favor de reducción de las jornadas laborales, medidas dirigidas a paliar la precarización del mercado laboral – especialmente en puestos no cualificados – son algunos temas que pueden ser sometidos a debate.
* A su vez, los procesos de aprendizaje automático plantean dilemas éticos en cuanto al tratamiento de la información y privacidad de las personas. Por ello, los procesos asociados deben desarrollarse en un marco regulatorio que salvaguarde las libertades y derechos de los ciudadanos.
* Por último, en un mundo globalizado e interconectado, donde acontecimientos puntuales pueden producir cambios de orden mundial, cobra aún más relevancia el uso de mecanismos de aprendizaje automático. La situación vivida tras la crisis sanitaria del COVID-19 es ejemplo de ello, para el cual algunas organizaciones ya están aplicando algunas soluciones comentadas.

# BIBLIOGRAFÍA

[1] D. López-Rodríguez and A. García-Linares, “Predictive and Populational Model for Alzheimer’s Disease Using Structural Neuroimaging BT - XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013,” 2014, pp. 285–288.

[2] S. H. Liao, Y. J. Chen, and Y. T. Lin, “Mining customer knowledge to implement online shopping and home delivery for hypermarkets,” *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3982–3991, 2011.

[3] D. Nikolov and D. Kim, “Learning to Predict Movie Ratings from the Netflix Dataset,” pp. 5–6.

[4] JM. Rízquez-Fernández, “Aplicación de redes neuronales artificiales al campo de la navegación aérea,” *Hispaviación*, 2019. [Online]. Available: http://www.hispaviacion.es/aplicacion-redes-neuronales-artificiales-al-campo-la-navegacion-aerea/. [Accessed: 20-May-2020].

[5] B. Marr, “That’s Data Science: Airbus Puts 10,000 Sensors in Every Single Wing!,” *Data Science Central*, 2015. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/that-s-data-science-airbus-puts-10-000-sensors-in-every-single. [Accessed: 20-May-2020].

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. 2008.

[7] M. Ruiz, L. E. Mujica, M. Quintero, J. Florez, and S. Quintero, “Magnetic Flux Leakage and Principal Component Analysis for metal loss approximation in a pipeline,” vol. 628, no. 1, 2015.

[8] E. Madrid, “BiciMAD.” [Online]. Available: https://www.bicimad.com/. [Accessed: 20-May-2020].

[9] “Qué es BiciMAD,” *EMT Madrid*. [Online]. Available: https://www.bicimad.com/index.php?s=que. [Accessed: 20-May-2020].

[10] Portal de datos abiertos del Ayuntamiento de Madrid, “BiciMAD. Alta de usuarios y usos por día del servicio público de bicicleta eléctrica.” [Online]. Available: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=6d8bdae2be63c410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD. [Accessed: 21-May-2020].

[11] City Performance Tool, “Madrid 2020-2030,” 2017.

[12] Comunidad de Madrid, “Encuesta Sintética de Movilidad en la Comunidad de Madrid ESM14,” 2014.

[13] “Motosharing Muving.” [Online]. Available: https://www.muving.com/. [Accessed: 20-May-2020].

[14] “car2go España.” [Online]. Available: https://www.car2go.com/ES/es/. [Accessed: 20-May-2020].

[15] A. Emilio *et al.*, “Análisis de la demanda potencial de usuarios de BiciMAD . Recomendaciones para la extensión del sistema .,” 2016.

[16] V. Mrkajic, D. Vukelic, and A. Mihajlov, “Reduction of CO 2 emission and non-environmental co-benefits of bicycle infrastructure provision: the case of the University of Novi Sad, Serbia,” *Renew. Sustain. Energy Rev.*, vol. 49, pp. 232–242, 2015.

[17] “Bonopark.” [Online]. Available: http://www.bonopark.es/. [Accessed: 20-May-2020].

[18] E. Madrid, “Datos estáticos BICIMAD - OPENDATA.” [Online]. Available: https://opendata.emtmadrid.es/Datos-estaticos/Datos-generales-(1). [Accessed: 20-May-2020].

[19] EMT Madrid, “Informe de situación BiciMAD - febrero 2018,” 2018.

[20] “Observatorio de la Bicicleta Pública en España.” [Online]. Available: https://bicicletapublica.es/. [Accessed: 22-May-2020].

[21] S. A. Barcelona de Servicios Municipales, “Bicing.” [Online]. Available: https://www.bicing.barcelona/es. [Accessed: 20-May-2020].

[22] S. L. . JCDecaux España, “Sevici Sevilla.” [Online]. Available: http://www.sevici.es/. [Accessed: 20-May-2020].

[23] Ayuntamiento de Zaragoza, “Bizi Zaragoza.” [Online]. Available: https://www.bizizaragoza.com/. [Accessed: 20-May-2020].

[24] EMT Málaga, “Málagabici.” [Online]. Available: http://malagabici.malaga.eu/webpublica/. [Accessed: 20-May-2020].

[25] EMT Madrid, “Tarifas BiciMAD.” [Online]. Available: https://www.bicimad.com/index.php?s=tarifas. [Accessed: 22-May-2020].

[26] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “The KDD Process for Extracting Useful Knowledge from Volumes of Data,” *Commun. ACM*, vol. 39, no. 11, pp. 27–34, 1996.

[27] J. Rowley, “The wisdom hierarchy: Representations of the DIKW hierarchy,” *J. Inf. Sci.*, vol. 33, no. 2, pp. 163–180, 2007.

[28] Cisco, “VNI Complete Forecast Highlights,” 2016.

[29] Cisco, “Cisco VNI Global IP Traffic Forecast 2017-2022,” 2018.

[30] Ayuntamiento de Málaga, “Lineas y horarios bus - Google Transit - Conjuntos de datos - Datos abiertos.” [Online]. Available: https://datosabiertos.malaga.eu/dataset/lineas-y-horarios-bus-google-transit. [Accessed: 20-May-2020].

[31] Cisco Networking Academy, “IoT Fundamentals: Connecting Things.” .

[32] C. Pete *et al.*, “CRISP-DM,” *Cris. Consort.*, no. 1.0, p. 76, 2000.

[33] K. Lee, “Data Mining for Business Intelligence.” .

[34] M. Stone, “Cross-Validatory Choice and Assessment of Statistical Predictions,” *J. R. Stat. Soc. Ser. B*, vol. 36, no. 2, pp. 111–147, May 1974.

[35] S. Yadav and S. Shukla, “Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification.” IEEE, pp. 78–83, 2016.

[36] J. Shlens, “A Tutorial on Principal Component Analysis,” 2014.

[37] J. Amat-Rodrigo, “Introducción a la Regresión Lineal Múltiple,” *cienciadedatos.net*, 2016. [Online]. Available: https://www.cienciadedatos.net/documentos/25\_regresion\_lineal\_multiple. [Accessed: 22-May-2020].

[38] L. Igual and S. Seguí, *Introduction to Data Science: A Python Approach to Concepts, Techniques and Applications*. 2017.

[39] T. P. Minka, “Bayesian linear regression,” 1998.

[40] G. Celant and M. Broniatowski, “Interpolation and Extrapolation Optimal Designs 1. Polynomial Regression and Approximation Theory,” in *ISTE Ltd. John Wiley & Sons, Inc.*, 2016, pp. 3–4.

[41] H. Drucker, C. J. C. Surges, L. Kaufman, A. Smola, and V. Vapnik, “Support vector regression machines,” 1997.

[42] L. Bottou and O. Bousquet, “The Tradeoffs of Large-scale Learning,” in *Optimization for machine learning*, S. Sra, S. Nowozin, and S. J. Wright, Eds. Cambridge, Mass: MIT Press, 2012, pp. 351–368.

[43] N. S. Altman, “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression,” *Am. Stat.*, vol. 46, no. 3, pp. 175–185, 1992.

[44] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[45] Machine Learning Group. Universidad de Waikato, “WEKA.” [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 27-May-2020].

[46] KNIME.com AG, “KNIME.” [Online]. Available: https://www.knime.com. [Accessed: 21-May-2020].

[47] Oracle Technology Network, “Oracle Java SE.” [Online]. Available: https://www.java.com. [Accessed: 21-May-2020].

[48] The R Fundation, “R.” [Online]. Available: https://www.r-project.org/. [Accessed: 21-May-2020].

[49] Python Software Fundation, “Python.” [Online]. Available: https://www.python.org/. [Accessed: 21-May-2020].

[50] W. McKinney, “Pandas.” .

[51] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[52] “Matplotlib: Python plotting — Matplotlib 3.2.1 documentation.” [Online]. Available: https://matplotlib.org/. [Accessed: 24-May-2020].

[53] Microsoft, “Microsoft Visual Studio 2017.”

[54] D. Ho, “Notepad++.”

[55] Microsoft, “Microsoft Excel.”

[56] EMT Madrid, “Modelo de datos de la información de uso del servicio BiciMAD,” pp. 1–8, 2017.

[57] Google, “Google Earth Pro.” [Online]. Available: https://www.google.com/intl/es/earth/. [Accessed: 21-May-2020].

[58] Agencia Estatal de Meteorología, “AEMET OpenData.” [Online]. Available: http://www.aemet.es/es/datos\_abiertos/AEMET\_OpenData. [Accessed: 21-May-2020].

[59] The Heidelberg Institute for Geoinformation Technology, “Openroute Service.” [Online]. Available: https://openrouteservice.org/. [Accessed: 21-May-2020].

[60] Microsoft, “Especificaciones y límites de Excel - Excel.” [Online]. Available: https://support.office.com/es-es/article/especificaciones-y-límites-de-excel-1672b34d-7043-467e-8e27-269d656771c3. [Accessed: 24-May-2020].

[61] Google Cloud, “Routes y Directions | Google Maps Platform.” [Online]. Available: https://cloud.google.com/maps-platform/routes/?refresh=1&hl=es. [Accessed: 24-May-2020].

[62] Ministerio de Agricultura Alimentación y Medio ambiente, “Real Decreto 2032/2009, de 30 de diciembre, por el que se establecen las unidades legales de medida,” *Bol. Of. del Estado*, p. 13, 2016.

[63] Bureau International des Poids et Mesures, “Résolution 10 de la 22e CGPM,” 2003.

[64] A. A. Sodemann, M. P. Ross, and B. J. Borghetti, “A review of anomaly detection in automated surveillance,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1257–1272, 2012.

[65] P. Sanagapati, “Anomaly Detection - Credit Card Fraud Analysis,” *Kaggle*, 2019. [Online]. Available: https://www.kaggle.com/pavansanagapati/anomaly-detection-credit-card-fraud-analysis. [Accessed: 24-May-2020].

[66] V. A. Siris and F. Papagalou, “Application of anomaly detection algorithms for detecting SYN flooding attacks,” *Comput. Commun.*, vol. 29, no. 9 SPEC. ISS., pp. 1433–1442, 2006.

[67] S. Asaithambi, “Why, How and When to Scale your Features - GreyAtom,” *Medium*, 2017. [Online]. Available: https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e. [Accessed: 24-May-2020].

[68] M. Á. Medina, “La esperada ampliación de Bicimad no despega,” *EL PAÍS*, 2019.

[69] S. Dray and J. Josse, “Principal component analysis with missing values: a comparative survey of methods,” *Plant Ecol.*, vol. 216, no. 5, pp. 657–667, 2015.

[70] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, 2015.

[71] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, 2010.

[72] J. Amat-Rodrigo, “Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE,” *cienciadedatos.net*, 2017. [Online]. Available: https://www.cienciadedatos.net/documentos/35\_principal\_component\_analysis. [Accessed: 28-May-2020].

[73] “python - Plot PCA loadings and loading in biplot in sklearn (like R’s autoplot),” *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/39216897/plot-pca-loadings-and-loading-in-biplot-in-sklearn-like-rs-autoplot. [Accessed: 28-May-2020].

[74] L. Li, “Principal Component Analysis for Dimensionality Reduction,” *Towards Data Science*. [Online]. Available: https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad. [Accessed: 28-May-2020].

[75] Na8, “Comprende Principal Component Analysis,” *Aprende Machine Learning*. [Online]. Available: https://www.aprendemachinelearning.com/comprende-principal-component-analysis/. [Accessed: 28-May-2020].

[76] A. Hyvärinen, “Fast and robust fixed-point algorithms for independent component analysis,” *IEEE Trans. Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[77] N. Correa, T. Adali, and V. D. Calhoun, “Performance of blind source separation algorithms for fMRI analysis using a group ICA method,” *Magn. Reson. Imaging*, vol. 25, no. 5, pp. 684–694, 2007.

[78] J. S. Kidwell and L. H. Brown, “Ridge Regression as a Technique for Analyzing Models with Multicollinearity,” *J. Marriage Fam.*, vol. 44, no. 2, p. 287, 1982.

[79] R. Tibshirani, “Regression Shrinkage and Selection via the Lasso,” *Wiley R. Stat. Soc.*, vol. 58, no. 1, pp. 267–288, 2016.

[80] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *J. R. Stat. Soc.*, vol. 67, pp. 301–320, 2005.

[81] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, “Online Passive-Aggressive Algorithms,” *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, 2016.

[82] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[83] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[84] H. Drucker and C. Cortes, “Boosting Decision Trees,” *Adv. Neural Inf. Process. Syst.*, no. July, pp. 479–485, 1996.

[85] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, vol. 13. 2005.

[86] “scikit-learn documentation,” *Scikit Learn*. [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 28-May-2020].

[87] S. S. Skiena, *The data science design manual*. 2017.

[88] R. A. Fisher, “THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS,” *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, 1936.

[89] C. Gini, *Variabilità e mutabilità; contributo allo studio delle distribuzioni e delle relazioni statistiche. [Fasc. I.]*. Tipogr. di P. Cuppini, 1912, 1912.

[90] Scikit Learn, “1.10. Decision Trees — scikit-learn 0.23.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/tree.html#tree. [Accessed: 28-May-2020].

[91] D. Harrison and D. Rubinfeld, “Hedonic Housing Prices and the Demand for Clean Air,” *J. Environ. Econ. Manage.*, vol. 5, no. 1, p. 81, 1978.

[92] T. Parr and P. Grover, “How to visualize decision trees,” *explained.ai*. [Online]. Available: https://explained.ai/decision-tree-viz/. [Accessed: 28-May-2020].

[93] G. Zhang and Y. Lu, “Bias-corrected random forests in regression,” *J. Appl. Stat.*, vol. 39, no. 1, pp. 151–160, 2012.

[94] “3.2.4.3.4. sklearn.ensemble.ExtraTreesRegressor — scikit-learn 0.23.1 documentation,” *Scikit Learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html. [Accessed: 28-May-2020].

[95] “3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.23.1 documentation,” *Scikit Learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: 28-May-2020].

[96] N. Halder, “How to find optimal parameters for CatBoost using GridSearchCV for Regression in Python,” *Medium*, 2019. [Online]. Available: https://medium.com/@nilimeshhalder/how-to-find-optimal-parameters-for-catboost-using-gridsearchcv-for-regression-in-python-ef778b60d95d. [Accessed: 28-May-2020].

[97] M. Sokovic, D. Pavletic, and K. Kern-Pipan, “Quality Improvement Methodologies – PDCA Cycle , RADAR Matrix , DMAIC and DFSS,” *J. Achiev. Mater. Manuf. Eng.*, vol. 43, no. 1, pp. 476–483, 2010.

[98] Apache Software Fundation, “Apache Hadoop.” [Online]. Available: https://hadoop.apache.org/. [Accessed: 28-May-2020].

[99] J. Wang and Z. Liu, “Parallel data mining optimal algorithm of virtual cluster,” in *Proceedings - 5th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2008*, 2008, vol. 5, pp. 358–362.

[100] Microsoft, “Servicios de informática en la nube | Microsoft Azure.” [Online]. Available: https://azure.microsoft.com/es-es/. [Accessed: 28-May-2020].

[101] Amazon, “AWS | Cloud Computing - Servicios de informática en la nube.” [Online]. Available: https://aws.amazon.com/es/. [Accessed: 28-May-2020].

[102] Google, “Servicios de cloud computing | Google Cloud.” [Online]. Available: https://cloud.google.com/?hl=es. [Accessed: 28-May-2020].

[103] International Business Machines Corporation, “IBM Cloud - España.” [Online]. Available: https://www.ibm.com/es-es/cloud. [Accessed: 28-May-2020].

[104] D. Opitz and R. Maclin, “Popular Ensemble Methods: An Empirical Study,” *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, 1999.

[105] T. Hill, M. O’Connor, and W. Remus, “Neural network models for time series forecasts,” *Manage. Sci.*, vol. 42, no. 7, pp. 1082–1092, 1996.

[106] Red Eléctrica de España, “Demanda de energía eléctrica en tiempo real, estructura de generación y emisiones de CO2.” [Online]. Available: https://demanda.ree.es/visiona/home. [Accessed: 28-May-2020].

[107] “REData API | Red Eléctrica de España.” [Online]. Available: https://www.ree.es/es/apidatos. [Accessed: 28-May-2020].

[108] Red Eléctrica de España, “Demanda eléctrica y actividad económica: ¿Cambio de paradigma?,” 2019.

# ANEXOS

## Anexo I: Código Python de consultas de trayectos a API de Openroute Service

import sys

import pandas

import pandas as pd

import csv

import time

import requests

#Importación de datos de trayectos únicos registrados

print('Inicio del proceso:',time.asctime( time.localtime(time.time())))

trayectos = pd.read\_csv("C:\\MYPATH\\TFG\\API\\trayectos\\trayectosunicos\_coordenadas.csv", delimiter=';')longitud=len(trayectos)

SalidaAPI = pd.DataFrame(columns=['id', 'origen', 'final','status','reason','text'])

path\_destino='C:\\MYPATH\\TFG\\API\\trayectos\\output\\'

#API restringe 2000 consultas/key-día. Se preparan varios ciclos con varios accesos.

inicio\_hoy=0 #A configurar en cada grupo de ciclos

fin\_hoy=3000 #A configurar en cada grupo de ciclos

#CICLO 1

inicio=0 #A configurar en cada ciclo

fin=1500 #A configurar en cada ciclo

APIkey='xxxxxxxxxx1110001cf6248a3a53c0b7aaa4baebd56b3xxxxxxxxxx' #Código de acceso (API key). Contenido omitido, puede puede obtener una en https://openrouteservice.org/

# Bucle donde se suceden todas las consultas lanzadas en cada ciclo

for i in range(inicio,(fin),1):

#Cuerpo de la consulta

body = {"coordinates":[[trayectos.loc[i][4],trayectos.loc[i][3]],[trayectos.loc[i][6],trayectos.loc[i][5]]] }

headers = {'Accept': 'application/json, application/geo+json, application/gpx+xml, img/png; charset=utf-8','Authorization': APIkey}

call = requests.post('https://api.openrouteservice.org/v2/directions/cycling-electric', json=body, headers=headers) #Dirección URL de la consulta. Se especifica modalidad de transporte.

print(call.status\_code, call.reason)

print(call.text)

SalidaQuery=pd.DataFrame([[trayectos.loc[i][0],trayectos.loc[i][1],trayectos.loc[i][2],call.status\_code,call.reason,call.text]],columns=['id', 'origen', 'final','status','reason','text'])

SalidaAPI=SalidaAPI.append(SalidaQuery) #Resultado incluido en el DataFrame Global

time.sleep(1.4) #Tiempo de espera necesario para no exceder 40 consultas por minuto (restricción de aplicación)

print('Completadas',i,'de',len(trayectos),'consultas.') #Salida del progreso del ciclo

SalidaAPI.to\_csv(path\_destino + 'salidaAPI('+str(inicio\_hoy)+'-'+str(fin\_hoy)+')resto.csv', encoding='utf-8', index=False, sep=';') #Almacenamiento local en CSV

#CICLO 2

inicio=1500

fin=3000

APIkey='xxxxxxxxxx243501deh648aa2001ba4h13245622baddaxxxxxxxxxx' #Código de acceso (API key). Contenido omitido, puede puede obtener una en https://openrouteservice.org/

for i in range(inicio,(fin),1):

body = {"coordinates":[[trayectos.loc[i][4],trayectos.loc[i][3]],[trayectos.loc[i][6],trayectos.loc[i][5]]] }

headers = {'Accept': 'application/json, application/geo+json, application/gpx+xml, img/png; charset=utf-8','Authorization': APIkey}

call = requests.post('https://api.openrouteservice.org/v2/directions/cycling-electric', json=body, headers=headers)

print(call.status\_code, call.reason)

print(call.text)

SalidaQuery=pd.DataFrame([[trayectos.loc[i][0],trayectos.loc[i][1],trayectos.loc[i][2],call.status\_code,call.reason,call.text]],columns=['id', 'origen', 'final','status','reason','text'])

SalidaAPI=SalidaAPI.append(SalidaQuery)

time.sleep(1.4)

print('Completadas',i,'de',len(trayectos),'consultas.')

SalidaAPI.to\_csv(path\_destino + 'salidaAPI('+str(inicio\_hoy)+'-'+str(fin\_hoy)+')resto.csv', encoding='utf-8', index=False, sep=';') #Almacenamiento local en CSV. Sobrescribe al anterior.

#Tantos ciclos por día como códigos de acceso, o códigos pendientes de consultar, se dispongan.

print('Fin del proceso:',time.asctime( time.localtime(time.time())))

## Anexo II: Código Python correspondiente a conversión a CSV, filtrado y unificación de datos estáticos de desplazamientos de BiciMAD

import sys

import pandas

import pandas as pd

import csv

import time

import datetime

import numpy as np

pd.set\_option('display.max\_columns', 50)

pd.set\_option('display.max\_rows', 500)

#PARTE 1: Conversión formato

#Bucle de transformación de ficheros deformato JSON a CSV

print('Inicio del proceso:',time.asctime( time.localtime(time.time())))

meses\_registrados= ('201704','201705','201706','201707','201708','201709','201710','201711','201712','201801','201802','201803','201804','201805','201806','201807','201808','201809','201810','201811','201812','201901','201902','201903','201904','201905','201906')

path\_origen= 'C:\\MYPATH\\TFG\\Datos\\JSON\\'

path\_destino = 'C:\\MYPATH\\TFG\\Datos\\DataSet\\'

datosGlobal=pd.DataFrame()

for mensualidad in meses\_registrados:

#Lectura de ficheros JSON haciendo uso de 'chunksize'

chunks = []

for chunk in pd.read\_json(path\_origen + mensualidad +'\_Usage\_Bicimad.json',chunksize=20000,encoding="latin-1", lines=True, orient='values',dtype={'\_id':str,'ageRange':int, 'idplug\_base':str, 'idplug\_station' : str, 'idunplug\_base': str, 'idunplug\_station': str, 'travel\_time': int, 'unplug\_hourTime':str, 'user\_day\_code': str, 'user\_type':str , 'zip\_code':str}):

chunks.append(chunk)

df = pd.concat(chunks, axis=0)

#Condición de eliminación del campo TRACKS, si existe

if df.columns.isin(['track']).any():

df = df.drop('track', 1) #valor 1 indica que actúa sobre las columnas, 0 para elimitar filas

print('El campo TRACK ha sido suprimido de',mensualidad+'\_Usage\_Bicimad')

#PARTE 2: filtrado

#Filtro campo '\_id'

df['\_id'] = df['\_id'].str.replace('$', "",regex=True)

df['\_id'] = df['\_id'].str.replace("{'oid': '", "",regex=True)

df['\_id'] = df['\_id'].str.replace("'}", "",regex=True)

#Filtro campo 'unplug\_hourTime'

df['unplug\_hourTime'] = df['unplug\_hourTime'].str.replace('$', "",regex=True)

df['unplug\_hourTime'] = df['unplug\_hourTime'].str.replace("{'date': '", "",regex=True)

df['unplug\_hourTime'] = df['unplug\_hourTime'].str.replace("'}", "",regex=True)

#Supresión de registros duplicados

df.drop\_duplicates(keep = 'first', inplace = True)

#PARTE 3: Creación de variables temporales y clave primaria

#Campo 'unplug\_hourTime' convertido en formato Datetime

df['unplug\_hourTime']=pd.to\_datetime(df['unplug\_hourTime'], utc=False)

#TIEMPOS

df=pd.DataFrame()

df=df

df['year'] = df['unplug\_hourTime'].dt.year

df['month'] = df['unplug\_hourTime'].dt.month

df['day'] = df['unplug\_hourTime'].dt.day

#Fecha como clave primaria con meteo

df['datetime']=df['unplug\_hourTime'].dt.year.astype(str)+'/'+df['unplug\_hourTime'].dt.month.astype(str)+'/'+df['unplug\_hourTime'].dt.day.astype(str)

#Visualización del resultado

df.head()

df.tail()

#PARTE 4: Almacenamiento local

#Copia de datos en nuevo DataFrame (en caso de recuperación inmediata, ahorra el tiempo de lectura)

df1=pd.DataFrame()

df1=df

#Generación de fichero CSV

df.to\_csv(path\_destino+'parcial\_trayectosBiciMAD\_v1.csv', encoding='utf-8', index=False)

## Anexo III: Código Python correspondiente a asociación de datos en un solo conjunto (desplazamientos + distancia y duración + meteo)

import sys

import pandas

import pandas as pd

import csv

import time

import datetime

pd.set\_option('display.max\_columns', 50)

#Importación datos de desplazamientos de BiciMAD.

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\API\\trayectos\\output\\parcial\_trayectosBiciMAD\_v1.csv', chunksize=2000000, low\_memory=False,dtype={'\_id':str,'year':int, 'month':int, 'day':int, 'datetime':str, ageRange':int, 'idplug\_base':str, 'idplug\_station' : str, 'idunplug\_base': str, 'idunplug\_station': str, 'travel\_time': int, 'unplug\_hourTime': str, 'user\_day\_code': str, 'user\_type':str , 'zip\_code':str}):

chunks.append(chunk)

df = pd.concat(chunks, axis=0)

df.head() #Visualización

#Importación datos consulta API Openroute Service

#Requiere de procesamiento previo: unificación de ficheros de respuestas de consulta, separación en columnas y reemplazo (compatible con Excel)

#Nota: se recomienda previa aplicación de la subsanación de omitidos (p.e.: imputación de valores omitidos)

ORSquery=pd.read\_csv('C:\\MYPATH\\TFG\\API\\trayectos\\output\\parcial\_ORS\_globalv1.csv', sep=';', low\_memory=False,dtype={'id':str,'keyTry':str, 'origen':str, 'destino' : str, 'reason': str, 'coords': str, 'distance': str, 'duration': str})

ORSquery.head() #Visualización

ORSmerge=ORSquery.drop(['id','origen','destino','status','reason','coords'],axis=1)

ORSmerge.head() #Visualización

#Creación de nuevo campo que sirve como clave primaria entre desplazamientos y ORS:

df['keyTry']=df['idunplug\_base']+'-'+df['idplug\_base']

df['keyTry'].head() #Visualización

df=df.sort\_values('unplug\_hourTime') #Registros ordenados cronológicamente

#Asociación entre desplazamientos y respuesta de ORS

df2=pd.merge(df,ORSmerge)

df2.head() #Visualización

#Importación registros meteorológicos de AEMET.

#Consulta a través de https://opendata.aemet.es/, almacenamiento y preprocesamiento posterior (compatible con excel).

chunks = []

for chunk in pd.read\_csv('D:\\CS PC MAXI\\Documentos\\ETSII\\TFG\\API\\AEMET\\reportesAEMET(reducido).csv', chunksize=2000000, low\_memory=False,sep=';'):

chunks.append(chunk)

meteo = pd.concat(chunks, axis=0)

#Cambio de separador decimal de comas a puntos y supresión código en conjunto (compatible variable tipo float)

meteo['meteo\_tmed'] = meteo['meteo\_tmed'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_prec'] = meteo['meteo\_prec'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_tmin'] = meteo['meteo\_tmin'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_tmax'] = meteo['meteo\_tmax'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_dir'] = meteo['meteo\_dir'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_velmedia'] = meteo['meteo\_velmedia'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_racha'] = meteo['meteo\_racha'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_presMax'] = meteo['meteo\_presMax'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_presMin'] = meteo['meteo\_presMin'].astype(str).str.replace(',', ".",regex=True)

#Nota: la creación de un registro de año / mes / día en el conjunto de meteo debe ir en consonancia con el formato de fecha considerado en la clave primaria de los desplazamientos.

#Por ejemplo: 2017/04/01 mantine formato diferente a 2017/4/1.

df=pd.merge(df,meteo) #Asociación entre desplazamientos y registros meteo de AEMET

df=df.drop\_duplicates() #Supresión de registros duplicados

df.to\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\completo\_desplazamientoORSmeteo\_v1.csv', encoding='utf-8', index=False, sep=';')

## Anexo IV: Código Python correspondiente a generación de gráficos para el análisis descriptivo de los datos empleados

import sys

import pandas

import pandas as pd

import csv

import time

import datetime

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

pd.set\_option('display.max\_columns', 50)

pd.set\_option('display.max\_rows', 500)

#Lectura de CSV donde se alamacenan los registros completos

chunks = []

for chunk in pd.read\_csv('D:\\CS PC MAXI\\Documentos\\ETSII\\TFG\\Datos\\DataSet\\completo\_desplazamientoORSmeteo\_v1.csv', sep=',', chunksize=2000000, low\_memory=False):

chunks.append(chunk)

df = pd.concat(chunks, axis=0)

#df['unplug\_hourTime']=pd.to\_datetime(df['unplug\_hourTime'], utc=False)

#Pasa emplearse df2 como copia de df. Con ello se ahorra tiempo en caso de querer recuperar la información original al no tener que leer de nuevo el CSV

df2=df

#Exploración de número de registros nulos por columna contemplada en el conjunto de datos.

for i in df2.columns:

print(i+':'+ str(len(df2[df2[i].isnull()])))

#Se define df3 como muestra aleatoria de 100.000 registros de df2 (sampleado)

df3=df2.sample(n=100000,random\_state=1) #random\_state como semilla de generador de números aleatorios

#Descripción de variables <<< Ilustración 6.1 >>>

df2.describe()

#Diagrama de SAMPLEADO Age\_range <<< Ilustración 6.2>>>

x\_labels = ['No identificado','< 17 años','17-18 años','19-26 años','27-40 años','41-65 años','>65 años']

df3=df2['ageRange'].sample(n=100000,random\_state=1).value\_counts().sort\_index().head(10)

df2['ageRange'].sample(n=100000,random\_state=1).value\_counts().sort\_index().plot(kind='bar',color='#002e5d').set\_xticklabels(x\_labels)

plt.title('Número de desplazamientos por rango de edad \n muestra n=100.000')

plt.ylabel('Número de desplazamientos')

plt.show()

#Diagrama de bigotes sobre muestra de 100.000 registros (con outliers) <<< Ilustraciones 6.3 y 6.4 >>>

df2.sample(n=100000,random\_state=1).boxplot(column=['travel\_time', 'duration'])

plt.title('Diagrama de caja de variables travel\_time y duration \n muestra n=100.000')

plt.xlabel('')

plt.ylabel('Segundos')

plt.show()

#Tiempo real frente predicho por Openroute Service <<< Ilustración 6.5>>>

df2.sample(n=10000,random\_state=1).plot.scatter(y='travel\_time', x=['duration'], figsize=(10,10), grid=True,s=0.5)

x = [1,3000]

y = [1,3000]

plt.plot(x,y, color="red", label = 'línea de referencia: tiempo previsto igual al de desplazamiento', linewidth=1 )

plt.title('Duración prevista por Openroute Service frente a tiempo empleado \n muestra n=10.000')

plt.xlabel('Duración prevista')

plt.ylabel('Tiempo empleado')

plt.show()

#Histograma de muestreo de variable travel\_time. <<< Ilustración 6.6 >>>

df2['travel\_time'].sample(n=100000,random\_state=1).plot.hist(bins=50000, xlim=(0,2000),color='#002e5d')

plt.title('Histograma de tiempos de viaje \n muestra n=100.000')

plt.xlabel('Segundos')

plt.ylabel('Frecuencia')

plt.show()

#Histograma de registros menores a 60s en variable travel\_time registros . <<< Ilustración 6.7 >>>

df4=df2[df2.travel\_time < 60]

df4.info()

df4['travel\_time'].plot.hist(bins=4000, xlim=(0,60),color='#002e5d')

plt.title('Histograma de tiempos de viaje < 60 segundos')

plt.xlabel('Segundos')

plt.ylabel('Frecuencia')

plt.show()

#Diagrama frecuencia por meses (muestreo) <<<Ilustración 6.8>>>

#Se eliminan anteriores al 1 de julio de 2017 (contempla julio 2017 - junio 2019 24 meses)

df5=df2.drop(df2[df2.datetime <2017063100].index)

x\_labels = ['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre']

df5['month'].sample(n=100000,random\_state=1).value\_counts().sort\_index().plot(kind='bar',color='#002e5d').set\_xticklabels(x\_labels)

plt.title('Número de desplazamientos por meses \n muestra n=100.000 \n (julio 2017 - junio 2019)')

plt.ylabel('Número de desplazamientos')

plt.show()

#Diagrama frecuencia por días de la semana (muestreo) <<<Ilustración 6.9>>>

#Se eliminan anteriores al 1 de julio de 2017 (contempla julio 2017 - junio 2019 24 meses)

#Diagrama de SAMPLEADO weekday

df3=df2.drop(df2[df2.datetime <2017040300].index)

x\_labels = ['Lunes','Martes','Miércoles','Jueves','Viernes','Sábado','Domingo']

df4=df3['weekday'].sample(n=100000,random\_state=1).value\_counts().sort\_index()

df5=pd.DataFrame({df4['Monday'], df4['Tuesday'],df4['Wednesday'],df4['Thursday'],df4['Friday'],df4['Saturday'],df4['Sunday']})

df5.sort\_index().plot(kind='bar',color='#002e5d',legend=False).set\_xticklabels(x\_labels)

plt.title('Número de desplazamientos por día de la semana \n muestra n=100.000')

plt.ylabel('Número de desplazamientos')

plt.show()

#Temperaturas máximas y mínimas por días <<<Ilustración 6.10>>>

df2['bydate']=df2['year'].astype(str)+'-'+df2['month'].astype(str)+'-'+df2['day'].astype(str)

s1=df2.groupby(['bydate'])['meteo\_tmax'].agg(lambda x: x.unique().mean()).plot(linewidth=0.5,color='#ff0000')

s2=df2.groupby(['bydate'])['meteo\_tmin'].agg(lambda x: x.unique().mean()).plot(linewidth=0.5,color='#002e5d')

plt.title('Temperaturas máximas y mínimas por día')

plt.ylabel('Temperatura (ºC)')

plt.xlabel('Fecha')

plt.legend()

plt.show()

#Número de desplazamientos por día y hora <<<Ilustración 6.11 (superior)>>>

df2['bydate']=df2['unplug\_hourTime'].astype(str).str[0:13]

df2['bydate'].value\_counts().sort\_index().plot(linewidth=0.5,color='#002e5d')

plt.title('Desplazamientos por hora')

plt.ylabel('Número de desplazamientos')

plt.show()

#Número de desplazamientos por días <<<Ilustración 6.11 (inferior)>>>

df2['bydate']=df2['unplug\_hourTime'].astype(str).str[0:10]

df2['bydate'].value\_counts().sort\_index().plot(linewidth=0.5,color='#002e5d')

plt.title('Desplazamientos por día')

plt.ylabel('Número de desplazamientos')

plt.show()

#Duración mediana por días <<<Ilustración 6.12>>>

df2['bydate']=df2['unplug\_hourTime'].astype(str).str[0:10]

df3=df2.groupby('bydate')['travel\_time'].median()

df3[(df3 < 10000) & (df3 > 300)].plot(linewidth=0.5,color='#002e5d')

plt.title('Duración mediana por día')

plt.xlabel('Fecha')

plt.show()

#Flujo neto (enganches-desenganches) en la estación 1 (Puerta del Sol)

id\_estacion=1

flujo=df2[np.logical\_or(df2['idunplug\_station'].astype(int)==id\_estacion , df2['idplug\_station'].astype(int)==id\_estacion)]

flujo['diaHora']=flujo['bydate']+'T'+flujo['datetime'].astype(str).str[8:10]

flujo\_entrada=flujo[flujo['idplug\_station'].astype(int)==id\_estacion]['diaHora'].value\_counts()

flujo\_salida=flujo[flujo['idunplug\_station'].astype(int)==id\_estacion]['diaHora'].value\_counts()

flujo\_entrada=pd.DataFrame({'diaHora':flujo\_entrada.index, 'entradas':flujo\_entrada.values})

flujo\_salida=pd.DataFrame({'diaHora':flujo\_salida.index, 'salidas':flujo\_salida.values})

flujo=flujo\_entrada.merge(flujo\_salida, on='diaHora')

flujo['neto']=flujo['entradas']-flujo['salidas']

flujo.groupby(['diaHora'])['neto'].agg(lambda x: x.unique().mean()).plot(linewidth=0.5,color='#002e5d')

plt.title('Flujo neto de estación 1 (Puerta del Sol) \n (diferencia entre enganches y desenganches para cada hora)')

plt.ylabel('Flujo neto')

plt.xlabel('Fecha y hora')

plt.legend('')

plt.show()

## Anexo V: Código Python correspondiente a imputación de valores omitidos en conjunto de registros meteorológicos

import sys

import pandas

import pandas as pd

import csv

import numpy as np

#Carga de registros meteorológicos

#Requiere preprocesamiento (compatible con Excel) entre la descarga y su carga en CSV.

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\API\\AEMET\\reportesAEMET(reducido).csv', chunksize=2000000, low\_memory=False,sep=';'):

chunks.append(chunk)

meteo = pd.concat(chunks, axis=0)

#Cambio de separador decimal, comas por puntos y supresión código 'Ip' en conjunto

meteo['meteo\_tmed'] = meteo['meteo\_tmed'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_prec'] = meteo['meteo\_prec'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_tmin'] = meteo['meteo\_tmin'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_tmax'] = meteo['meteo\_tmax'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_dir'] = meteo['meteo\_dir'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_velmedia'] = meteo['meteo\_velmedia'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_racha'] = meteo['meteo\_racha'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_presMax'] = meteo['meteo\_presMax'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_presMin'] = meteo['meteo\_presMin'].astype(str).str.replace(',', ".",regex=True)

meteo['meteo\_tmed'] = meteo['meteo\_tmed'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_prec'] = meteo['meteo\_prec'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_tmin'] = meteo['meteo\_tmin'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_tmax'] = meteo['meteo\_tmax'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_dir'] = meteo['meteo\_dir'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_velmedia'] = meteo['meteo\_velmedia'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_racha'] = meteo['meteo\_racha'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_presMax'] = meteo['meteo\_presMax'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo['meteo\_presMin'] = meteo['meteo\_presMin'].astype(str).str.replace('Ip', '',regex=True).astype(float)

meteo[meteo['meteo\_tmin'].isnull()] #Se obtienen índices 141 y 154

#Registros con índice 141 y 154:

#Estrategia de imputación: t\_max=(tmax\_dia-anterior+tmax\_dia-posterior)/2 t\_min=(tmin\_dia-anterior+tmin\_dia-posterior)/2 tmedia=(tmax-tmin)/2

meteo['meteo\_tmax'][141]=(meteo['meteo\_tmax'][141-1]+meteo['meteo\_tmax'][141+1])/2

meteo['meteo\_tmin'][141]=(meteo['meteo\_tmin'][141-1]+meteo['meteo\_tmin'][141+1])/2

meteo['meteo\_tmed'][141]=(meteo['meteo\_tmax'][141]+meteo['meteo\_tmin'][141])/2

meteo['meteo\_tmax'][154]=(meteo['meteo\_tmax'][154-1]+meteo['meteo\_tmax'][154+1])/2

meteo['meteo\_tmin'][154]=(meteo['meteo\_tmin'][154-1]+meteo['meteo\_tmin'][154+1])/2

meteo['meteo\_tmed'][154]=(meteo['meteo\_tmax'][154]+meteo['meteo\_tmin'][154])/2

meteo.info()

#Imputación de valores nulos por variables

#Estrategia de imputación: regresión lineal de valores nulos a partir de otras valores

from sklearn.linear\_model import LinearRegression

regression = LinearRegression()

#Creación de variables día, mes y año.

for i in range(0,len(meteo),1):

meteo['day'][i]=meteo['meteo\_date'][i][0:2] #año

meteo['month'][i]=meteo['meteo\_date'][i][3:5] #mes

meteo['year'][i]=meteo['meteo\_date'][i][6:10] #día

meteo['day']=meteo['day'].astype(int)

meteo['month']=meteo['month'].astype(int)

meteo['year']=meteo['year'].astype(int)

#Regresión 1: Pmax a partir de temperaturas y fechas

#División en conjunto de entrenamiento (X\_train y variable objetivo Y\_train) y de evaluación (X\_test como insumo e Y\_test como resultado)

x\_train = meteo[meteo['meteo\_presMax'].notnull()].drop(['meteo\_date','meteo\_prec','meteo\_dir','meteo\_velmedia','meteo\_racha', 'meteo\_racha','meteo\_presMax','meteo\_presMin'], axis=1)

y\_train = meteo[meteo['meteo\_presMax'].notnull()]['meteo\_presMax']

x\_test = meteo[meteo['meteo\_presMax'].isnull()].drop(['meteo\_date','meteo\_prec','meteo\_dir','meteo\_velmedia','meteo\_racha', 'meteo\_racha','meteo\_presMax','meteo\_presMin'], axis=1 )

y\_test = meteo[meteo['meteo\_presMax'].isnull()]['meteo\_presMax']

#Entrenamiento del modelo de regresión lineal

regression=LinearRegression().fit(x\_train, y\_train)

#Predicción del modelo a partir de datos de entrada X\_test

predicted = regression.predict(x\_test)

#Imputación de valores predichos a los valores nulos de la variable objetivo del conjunto de datos

meteo.meteo\_presMax[meteo.meteo\_presMax.isnull()] = predicted

#Regresión 2: Pmin a partir de temperaturas, fechas y Pmax

#División en conjunto de entrenamiento (X\_train y variable objetivo Y\_train) y de evaluación (X\_test como insumo e Y\_test como resultado)

x\_train = meteo[meteo['meteo\_presMin'].notnull()].drop(['meteo\_date','meteo\_prec','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMin'], axis=1)

y\_train = meteo[meteo['meteo\_presMin'].notnull()]['meteo\_presMin']

x\_test = meteo[meteo['meteo\_presMin'].isnull()].drop(['meteo\_date','meteo\_prec','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMin'], axis=1)

y\_test = meteo[meteo['meteo\_presMin'].isnull()]['meteo\_presMin']

#Entrenamiento del modelo de regresión lineal

regression=LinearRegression().fit(x\_train, y\_train)

#Predicción del modelo a partir de datos de entrada X\_test

predicted = regression.predict(x\_test)

#Imputación de valores predichos a los valores nulos de la variable objetivo del conjunto de datos

meteo.meteo\_presMin[meteo.meteo\_presMin.isnull()] = predicted

difPres\_medio=(meteo['meteo\_presMax']-meteo['meteo\_presMin']).mean()

for i in range(0,len(meteo),1):

if (meteo['meteo\_presMax'][i]-meteo['meteo\_presMin'][i]) < 0:

meteo['meteo\_presMin'][i]=(meteo['meteo\_presMax'][i]-difPres\_medio)

#Regresión 3: Precipitaciones a partir de temperaturas, fechas, Pmax y Pmin

meteo[meteo['meteo\_prec'].isnull()]

#Precipitación sin registros de viento nulos:

#Los valores nulos de precipitación no coinciden, en dichos registros,con valores nulos en el resto de campos. Se predice precipitaciones entrenando con valores de viento no nulos.

#División en conjunto de entrenamiento (X\_train y variable objetivo Y\_train) y de evaluación (X\_test como insumo e Y\_test como resultado)

x\_train = meteo[meteo['meteo\_prec'].notnull() & meteo['meteo\_dir'].notnull() & meteo['meteo\_velmedia'].notnull() & meteo['meteo\_racha'].notnull()].drop(['meteo\_date','meteo\_prec'], axis=1)

y\_train = meteo[meteo['meteo\_prec'].notnull() & meteo['meteo\_dir'].notnull() & meteo['meteo\_velmedia'].notnull() & meteo['meteo\_racha'].notnull()]['meteo\_prec']

x\_test = meteo[meteo['meteo\_prec'].isnull()].drop(['meteo\_date','meteo\_prec'], axis=1)

y\_test = meteo[meteo['meteo\_prec'].isnull()]['meteo\_prec']

#Entrenamiento del modelo de regresión lineal

regression=LinearRegression().fit(x\_train, y\_train)

#Predicción del modelo a partir de datos de entrada X\_test

predicted = regression.predict(x\_test)

#Imputación de valores predichos a los valores nulos de la variable objetivo del conjunto de datos

meteo.meteo\_prec[meteo.meteo\_prec.isnull()] = predicted

#Regresión 4: Vel media a partir de variables anteriores

#División en conjunto de entrenamiento (X\_train y variable objetivo Y\_train) y de evaluación (X\_test como insumo e Y\_test como resultado)

x\_train = meteo[meteo['meteo\_velmedia'].notnull()].drop(['meteo\_date','meteo\_dir','meteo\_racha','meteo\_velmedia'], axis=1)

y\_train = meteo[meteo['meteo\_velmedia'].notnull()]['meteo\_velmedia']

x\_test = meteo[meteo['meteo\_velmedia'].isnull()].drop(['meteo\_date','meteo\_dir','meteo\_racha','meteo\_velmedia'], axis=1)

y\_test = meteo[meteo['meteo\_velmedia'].isnull()]['meteo\_velmedia']

#Entrenamiento del modelo de regresión lineal

regression=LinearRegression().fit(x\_train, y\_train)

#Predicción del modelo a partir de datos de entrada X\_test

predicted = regression.predict(x\_test)

#Imputación de valores predichos a los valores nulos de la variable objetivo del conjunto de datos

meteo.meteo\_velmedia[meteo.meteo\_velmedia.isnull()] = predicted

#Regresión 5: Dirección del viento a partir de variables anteriores

#División en conjunto de entrenamiento (X\_train y variable objetivo Y\_train) y de evaluación (X\_test como insumo e Y\_test como resultado)

x\_train = meteo[meteo['meteo\_dir'].notnull()].drop(['meteo\_date','meteo\_dir','meteo\_racha'], axis=1)

y\_train = meteo[meteo['meteo\_dir'].notnull()]['meteo\_dir']

x\_test = meteo[meteo['meteo\_dir'].isnull()].drop(['meteo\_date','meteo\_dir','meteo\_racha'], axis=1)

y\_test = meteo[meteo['meteo\_dir'].isnull()]['meteo\_dir']

#Entrenamiento del modelo de regresión lineal

regression=LinearRegression().fit(x\_train, y\_train)

#Predicción del modelo a partir de datos de entrada X\_test

predicted = regression.predict(x\_test)

#Imputación de valores predichos a los valores nulos de la variable objetivo del conjunto de datos

meteo.meteo\_dir[meteo.meteo\_dir.isnull()] = predicted

meteo.info()

#Regresión 6: Racha de viento a partir de variables anteriores

#División en conjunto de entrenamiento (X\_train y variable objetivo Y\_train) y de evaluación (X\_test como insumo e Y\_test como resultado)

x\_train = meteo[meteo['meteo\_racha'].notnull()].drop(['meteo\_date','meteo\_racha'], axis=1)

y\_train = meteo[meteo['meteo\_racha'].notnull()]['meteo\_racha']

x\_test = meteo[meteo['meteo\_racha'].isnull()].drop(['meteo\_date','meteo\_racha'], axis=1)

y\_test = meteo[meteo['meteo\_racha'].isnull()]['meteo\_racha']

#Entrenamiento del modelo de regresión lineal

regression=LinearRegression().fit(x\_train, y\_train)

#Predicción del modelo a partir de datos de entrada X\_test

predicted = regression.predict(x\_test)

#Imputación de valores predichos a los valores nulos de la variable objetivo del conjunto de datos

meteo.meteo\_racha[meteo.meteo\_racha.isnull()] = predicted

#Visibilización del resultado obtenido

meteo.info()

#Almacenamiento local (formato CSV) de registros meteorológicos sin valores nulos

meteo.to\_csv('C:\\MYPATH\\TFG\\API\\AEMET\\reportesAEMET(sin missingdata).csv', encoding='utf-8', index=False)

#Asociación de registros subsanados en conjuntos de desplazamientos

#Carga del conjunto datos global, previo a la mitigación de valores nulos.

chunks = []

for chunk in pd.read\_csv('D:\\CS PC MAXI\\Documentos\\ETSII\\TFG\\Datos\\DataSet\\DatosGlobalTrymeteo2.csv', chunksize=20000, low\_memory=False,sep=','):

chunks.append(chunk)

df = pd.concat(chunks, axis=0)

df2=df

#Comprobación de valores nulos existentes

for i in df2.columns:

print(i+':'+ str(len(df2[df2[i].isnull()])))

#Supresión de variables metereológicas de registros de desplazamiento y nueva asociación a registros meteorológicos subsanados

meteo.info()

df3=df2.drop(['meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin'], axis=1)

meteo['meteo\_date']=meteo['day'].map(str)+'/'+meteo['month'].map(str)+'/'+meteo['year'].map(str)

df4=pd.merge(df3,meteo)

len(df3)

df4=df4.drop\_duplicates()

len(df4)

df4.head()

#Comprobación resultado

for i in df4.columns:

print(i+':'+ str(len(df4[df4[i].isnull()])))

#Supresión de variable 'zip\_code'

df4=df4.drop(['zip\_code'],axis=1)

#Almacenamiento local

df4.to\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\parcial\_trayectosBiciMAD\_v2.csv', encoding='utf-8', index=False, sep=';')

## Anexo VI: Código Python correspondiente a generación de conjunto de datos correspondiente a demanda del sistema BiciMAD.

import sys

import pandas

import pandas as pd

import csv

import time

import datetime

import numpy as np

pd.set\_option('display.max\_columns', 50)

pd.set\_option('display.max\_rows', 850)

#Importación de conjuntos de datos de desplazamientos

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\parcial\_trayectosBiciMAD\_v2.csv', sep=';', chunksize=2000000, low\_memory=False):

chunks.append(chunk)

df = pd.concat(chunks, axis=0)

#Conteo de número de desplazamientos por días y exportación del número en un nuevo DataFrame

df3=df.groupby([df['year'], df['month'], df['day']]).count()

df4=pd.DataFrame()

df4[['\_id']]=df3[['\_id']]

#Creación de nuevas columnas. 'Count' corresponde a secuencia cronológica que permita identificar posibles tendencias a lo largo del tiempo.

df4['index1'] = df4.index

df4=df4.sort\_index(by=['year','month','day'])

df4['meteo\_date']=""

df4['meteo\_date']=df4['meteo\_date'].astype(str)

df4['datetime']=""

df4['datetime']=df4['datetime'].astype(str)

df4['year']=""

df4['month']=""

df4['day']=""

df4['weekday\_int']=""

df4['weekday\_str']=""

df4['count']=""

#Generacíón de datetime y meteo\_date (clave primaria) a partir de index1 (tupla)

for i in range(0,len(df4)-1,1):

j=df4['index1'][i]

df4['count'][i]=i

if (len(str(j[1])) == 1):

k='0'+str(j[1])

else:

k=str(j[1])

if (len(str(j[2])) == 1):

l='0'+str(j[2])

else:

l=str(j[2])

df4['datetime'][i]=str(j[0])+k+l

df4['meteo\_date'][i]=(str(l)+'/'+str(k)+'/'+str(j[0]))

df4['year'][i]=j[0]

df4['month'][i]=j[1]

df4['day'][i]=j[2]

df4['weekday\_int'][i]=datetime.datetime(j[0],j[1], j[2]).weekday()+1

df4['weekday\_str'][i]=datetime.datetime(j[0],j[1], j[2]).strftime('%A')

#Importación de registros meteorológicos (sin valores nulos)

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\API\\AEMET\\reportesAEMET(sin missingdata).csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

meteo = pd.concat(chunks, axis=0)

#Asociación a registros meteorológicos con 'meteo\_date' como clave primaria.

df5=pd.merge(df4,meteo.drop(['year','month','day'],axis=1))

#Almecenamiento local (en formato CSV) de conjunto de datos correpsondiente a la demanda del sistema BiciMAD

df5.to\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', encoding='utf-8', index=False)

## Anexo VII: Código Python correspondiente a número de variables a considerar en análisis PCA

import sys

import pandas

import pandas as pd

import csv

import time

import datetime

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

#Importación de datos de demanda de BiciMAD

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

df5 = pd.concat(chunks, axis=0)

#Separación en variable objetivo (Y) y atributos (X) (véase Ilustración 7.7)

#Características que contribuyen

x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin']].astype(float).values

df5x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin']]

#Variable a predecir (desplazamientos por día)

y=df5['\_id'].astype(int).values

#Líneas de código desarrolladas empleando como referencia el recurso bibliográfico disponible en https://www.aprendemachinelearning.com/comprende-principal-component-analysis/

#Estandarización de características (input)

x = StandardScaler().fit\_transform(x)

scaler=StandardScaler()

scaler.fit(x) # Cálculo del valor promedio

X\_scaled=scaler.transform(x)# Resto de valores escalados con base en promedio

#Consideración de todas las componentes principales, tantas como atributos

pca=PCA(n\_components=13) # Otra opción es instanciar pca sólo con dimensiones nuevas hasta obtene

pca.fit(X\_scaled) # obtener los componentes principales

X\_pca=pca.transform(X\_scaled) # convertimos nuestros datos con las nuevas dimensiones de PCA

print("shape of X\_pca", X\_pca.shape)

expl = pca.explained\_variance\_ratio\_

print(expl)

#Varianza explicada por todas las componentes principales (100%)

print('suma:',sum(expl[0:13]))

#####################

#Gráfico de cargas <<<Ilustración 7.10>>>

# Se emplea parte del código propuesto en repositorio Stack Overflow

#https://stackoverflow.com/questions/39216897/plot-pca-loadings-and-loading-in-biplot-in-sklearn-like-rs-autoplot

#Definición de función empleada más adelante para generar el gráfico

def myplot(score,coeff,labels=None):

xs = score[:,0]

ys = score[:,1]

n = coeff.shape[0]

scalex = 1.0/(xs.max() - xs.min())

scaley = 1.0/(ys.max() - ys.min())

#plt.scatter(xs \* scalex,ys \* scaley, c = y)

for i in range(n):

plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)

if labels is None:

plt.text(coeff[i,0]\* 1.15, coeff[i,1] \* 1.15, "Var"+str(i+1), color = 'g', ha = 'center', va = 'center')

else:

plt.text(coeff[i,0]\* 1.15, coeff[i,1] \* 1.15, labels[i], color = 'g', ha = 'center', va = 'center')

plt.xlim(-1,1)

plt.ylim(-1,1)

plt.xlabel("Componente Principal PC{}".format(1))

plt.ylabel("Componente Principal PC{}".format(2))

plt.title("Gráfico de cargas/importancias de las variables en PC1 y PC2 \n ")

plt.grid()

plt.show()

pca.fit\_transform(x)[0:809]

myplot(pca.fit\_transform(x)[0:809],np.transpose(pca.components\_[0:809, :]),df5x.columns)

plt.show()

# Gráfico de proporción de varianza explicada por ponente principal <<<Ilutración 7.12>>>

#Se emplea código original de Lorraine Li disponible en https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad

matriz\_cov = np.cov(X\_pca.T)

autovalores, autovectores = np.linalg.eig(matriz\_cov)

tot = sum(autovalores)

var\_exp = [(i / tot) for i in sorted(autovalores, reverse=True)]

cum\_var\_exp = np.cumsum(var\_exp)

plt.bar(range(1,14), var\_exp, alpha=0.7,

align='center', label='Varianza explicada por PC')

plt.step(range(1,14), cum\_var\_exp, where='mid',

label='Varianza explicada acumulada (escalonada)')

plt.plot(range(1,14), cum\_var\_exp,

label='Varianza explicada acumulada (con pendiente)', linewidth=0.5, color='r')

plt.ylabel('Proporción de varia explicada')

plt.xlabel('Número de componentes principales empleadas')

plt.title('Proporción de varianza explicada por PC')

plt.legend()

plt.grid()

plt.show()

## Anexo VIII: Código Python empleado en la selección de modelo (sin PCA)

import pandas as pd

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

import numpy as np

import matplotlib.pyplot as plt

import sklearn.linear\_model

from sklearn.model\_selection import cross\_val\_score

#Importación de conjunto de datos de demanda del servicio BiciMAD

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

df5 = pd.concat(chunks, axis=0)

pd.set\_option('display.max\_columns', 15)

pd.set\_option('display.max\_rows', 820)

df5['cont']=df5.index.astype(int)

#Características que contribuyen

X=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']].astype(float).values

df5x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']]

#Variable a predecir (desplazamientos por día)

y=df5['\_id'].astype(int).values

from sklearn.decomposition import PCA, FastICA

from sklearn.linear\_model import LinearRegression,LogisticRegression, RidgeCV, Lasso,ElasticNetCV, BayesianRidge,PassiveAggressiveRegressor, SGDRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn import svm

from sklearn.gaussian\_process import GaussianProcessRegressor

from sklearn.gaussian\_process.kernels import DotProduct, WhiteKernel

from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor, GradientBoostingRegressor

#Separación en conjuntos de entrenamiento y evaluación

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

#Declaración de diferentes técnicas a ensayar

MODELOS.items()

MODELOS = {

'Linear regression': LinearRegression(),

'LogRegression': LogisticRegression(),

'Ridge': RidgeCV(),

'BayesianRidge': BayesianRidge(),

'Lasso': Lasso(normalize=True),

'ElasticNet' :ElasticNetCV(cv=5, random\_state=0),

'PassiveAggressiveRegressor': PassiveAggressiveRegressor(),

'Supported Vector Machine Reg' :svm.SVR(),

'K-nn': KNeighborsRegressor(n\_neighbors=5, weights='distance'),

'Random Forest' : RandomForestRegressor(random\_state=0, n\_estimators=10),

'Extra trees': ExtraTreesRegressor(max\_features=int(13),random\_state=0),

'GPR' : GaussianProcessRegressor(kernel=(DotProduct() + WhiteKernel()), random\_state=0),

'GBR' : GradientBoostingRegressor(random\_state=0, n\_estimators=10)

}

#Normalización (a escala) de los datos empleados

from sklearn import preprocessing

scale = StandardScaler()

X\_train\_std = scale.fit\_transform(X\_train)

X\_test\_std = scale.transform(X\_test)

from sklearn.model\_selection import ShuffleSplit

from sklearn.metrics import explained\_variance\_score, r2\_score

#Se declaran algunos diccionarios que permiten almacenar datos predichos y estadísticos sobre la calidad del ajusta

y\_test\_predict = dict()

R = dict()

RR=dict()

resultados=pd.DataFrame()

R2=dict()

ExpVar=dict()

for name, estimator in MODELOS.items():

estimator.fit(X\_train\_std, y\_train)

y\_test\_predict[name] = estimator.predict(X\_test\_std)

resultados[name]=y\_test\_predict[name]

cv = ShuffleSplit(n\_splits=5, test\_size=0.2, random\_state=int(0))

R2[name]=cross\_val\_score(estimator, X, y, scoring='r2', cv=cv, n\_jobs=-1).mean()

ExpVar[name]=cross\_val\_score(estimator, X, y,scoring='explained\_variance', cv=cv, n\_jobs=-1).mean()

#Conversión de diccionarios en DataFrame para facilitar su visualización

R2=pd.DataFrame(R2.items()).T

ExpVar=pd.DataFrame(ExpVar.items()).T

Comparativa=pd.concat([R2,ExpVar]).drop\_duplicates()

#Visualización del ajuste de cada modelo en términos de coef. determinación y varianza explicada

Comparativa.head()

#En términos de coeficiente de determinación <<<Ilustración 7.15>>>

R2.head()

#Generación de histogramas <<<Ilustración 7.16>>>

resultados['real']=y\_test

import matplotlib.pyplot as plt

import numpy as np

#HISTOGRAMA Regresión lineal

resultados['error Linear regression']=(resultados['Linear regression']-resultados['real'])

plt.subplot(231)

plt.title('Regresión lineal')

resultados['error Linear regression'].plot.hist(bins=150, xlim=(-7000,7000), ylim=(0,15),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error')

#HISTOGRAMA Regresión método Lasso

resultados['error Lasso']=(resultados['Lasso']-resultados['real'])

plt.subplot(232)

plt.title('Regresión Lasso')

resultados['error Lasso'].plot.hist(bins=150, xlim=(-7000,7000),ylim=(0,15),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error')

#HISTOGRAMA K-nn

resultados['error K-nn']=(resultados['K-nn']-resultados['real'])

plt.subplot(233)

plt.title('K-vecinos')

resultados['error K-nn'].plot.hist(bins=150, xlim=(-7000,7000),ylim=(0,15),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error')

#HISTOGRAMA Extra Trees

resultados['error Extra trees']=(resultados['Extra trees']-resultados['real'])

plt.subplot(234)

plt.title('Árboles ext. aleatorios')

resultados['error Extra trees'].plot.hist(bins=150, xlim=(-7000,7000),ylim=(0,15),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error')

#HISTOGRAMA Random Forest

resultados['error Random Forest']=(resultados['Random Forest']-resultados['real'])

plt.subplot(235)

plt.title('Bosque aleatorio')

resultados['error Random Forest'].plot.hist(bins=150, xlim=(-7000,7000),ylim=(0,15),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error')

#HISTOGRAMA GBR

resultados['error GBR']=(resultados['GBR']-resultados['real'])

plt.subplot(236)

plt.title('Potenciación del gradiente en árboles')

resultados['error GBR'].plot.hist(bins=150, xlim=(-7000,7000),ylim=(0,15),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error')

plt.tight\_layout(pad=0.1)

plt.show()

resultados.head()

#HISTOGRAMA Extra tree

resultados['error PassiveAggressiveRegressor']=(resultados['PassiveAggressiveRegressor']-resultados['real'])

a[1][1]=resultados['error PassiveAggressiveRegressor'].plot.hist(bins=150, xlim=(-7000,7000),color='#002e5d')

plt.title('Histograma error absoluto de PassiveAggressiveRegressor')

plt.xlabel('Error')

plt.ylabel('Frecuencia')

plt.show()

#Cálculo de raíz de error cuadrático de algunos estimadores

resultados['error K-nn']=((resultados['K-nn']-resultados['real'])\*\*2)\*\*(1/2)

resultados['error Linear regression']=((resultados['Linear regression']-resultados['real'])\*\*2)\*\*(1/2)

resultados['error Ridge']=((resultados['Ridge']-resultados['real'])\*\*2)\*\*(1/2)

#Gráfico de valores predichos frente a reales

resultados.plot.scatter(y='Extra trees', x=['real'], figsize=(10,10), grid=True,s=1)

x = [1,30000]

y = [1,30000]

plt.plot(x,y, color="red", label = 'línea de referencia: tiempo previsto igual al de desplazamiento', linewidth=1 )

plt.plot([0,1],[0,1])

plt.xlabel('real')

plt.ylabel('predicho')

plt.show()

## Anexo IX: Código Python empleado en la definición de hiper-parámetros del modelo de árboles extremadamente aleatorios (ExtraTrees)

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, ShuffleSplit

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor

import time

start\_time = time.time()

pd.set\_option('display.max\_columns', 15)

pd.set\_option('display.max\_rows', 810)

#Importación de conjunto de datos de la demanda del servicio BiciMAD

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

df5 = pd.concat(chunks, axis=0)

df5=pd.DataFrame(df5)

df5=df5.sort\_values(by='datetime')

df5['cont']=df5.index.astype(int)

#Características que contribuyen

X=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']].astype(float).values

df5x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']]

#Variable a predecir (desplazamientos por día)

y=df5['\_id'].astype(int).values

#División en 60% entrenamiento 20% evaluación 20% validación

X, X\_eval, y, y\_eval = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

# train 485 registros, test 162 registros , eval 162 registros

#Se han empleado líneas de código procedentes del recurso https://medium.com/@nilimeshhalder/how-to-find-optimal-parameters-for-catboost-using-gridsearchcv-for-regression-in-python-ef778b60d95d

m\_time = time.time()

print("GridSearch:--- %s segundos ---" % (m\_time - start\_time))

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer

from sklearn.metrics import accuracy\_score

modelo =ExtraTreesRegressor(max\_features=None, max\_depth=None, bootstrap=True, warm\_start=True)

#Declaración de valores que se dean probar en diferentes hiper-parámetros

parametros = {'min\_samples\_leaf': [2,3,4,5,6],

'criterion' : ['mse','mae'],

'oob\_score' : ['False','True'],

'min\_impurity\_decrease' : [0.0, 0.01, 0.05, 0.1, 0.2, 0.3],

'n\_estimators' : [20,100,1000,2000,5000,10000],

'ccp\_alpha' : [0.1, None]}

#Búsqueda secuencial emplendo validación cruzada k=5

grid = GridSearchCV(estimator=modelo, param\_grid = parametros, cv = 5, n\_jobs=-1, verbose=2)

#Advertencia: la siguiente línea de código conlleva la ejecución de un proceso que puede tomar varias horas. La configuración del modelo con los hiperparámetros resultantes se encuentra más adelante.

grid.fit(X, y)

# Representación de resultados obtenidos

print("\n========================================================")

print(" Resultado de búsqueda por Grid" )

print("========================================================")

print("\n Mejor configuración del estimador entre todos los parámetros ensayados:\n",

grid.best\_estimator\_)

print("\n El mejor índice de regresión ensayado:\n",

grid.best\_score\_)

print("\n Mejores parámetros entre los buscados:\n",

grid.best\_params\_)

print("\n ========================================================")

print("GridSearch:--- %s segundos ---" % (time.time() - m\_time))

print("Total:--- %s segundos ---" % (time.time() - start\_time))

#Definición de hiperparámetros, resultante de la búsqueda secuencial, para modelo ExtraTree

''' modelo=ExtraTreesRegressor(bootstrap='False', ccp\_alpha=0.0, criterion='mae',

max\_depth=1850, max\_features=None, max\_leaf\_nodes=None,

max\_samples=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1,

min\_samples\_split=5, min\_weight\_fraction\_leaf=0.0,

n\_estimators=470, n\_jobs=None, oob\_score='True',

random\_state=None, verbose=0, warm\_start=False) '''

## Anexo X: Código Python empleado en la definición de hiper-parámetros del modelo de bosque aleatorio (Random Forest)

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, ShuffleSplit

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor

import time

start\_time = time.time()

pd.set\_option('display.max\_columns', 15)

pd.set\_option('display.max\_rows', 810)

#Importación de conjunto de datos de la demanda del servicio BiciMAD

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

df5 = pd.concat(chunks, axis=0)

df5=pd.DataFrame(df5)

df5=df5.sort\_values(by='datetime')

df5['cont']=df5.index.astype(int)

#Características que contribuyen

X=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']].astype(float).values

df5x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']]

#Variable a predecir (desplazamientos por día)

y=df5['\_id'].astype(int).values

#División en 60% entrenamiento 20% evaluación 20% validación

X, X\_eval, y, y\_eval = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

# train 485 registros, test 162 registros , eval 162 registros

#Se han empleado líneas de código procedentes del recurso https://medium.com/@nilimeshhalder/how-to-find-optimal-parameters-for-catboost-using-gridsearchcv-for-regression-in-python-ef778b60d95d

m\_time = time.time()

print("GridSearch:--- %s segundos ---" % (m\_time - start\_time))

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer

from sklearn.metrics import accuracy\_score

modelo =RandomForestRegressor(max\_depth=None, warm\_start=True)

#Declaración de valores que se dean probar en diferentes hiper-parámetros

parametros = {'min\_samples\_leaf': [2,3,4,5,6],

'criterion' : ['mse','mae'],

'oob\_score' : ['False','True'],

'bootstrap' : ['False','True'],

'max\_features' : ['auto','sqrt', 7, 10],

'min\_impurity\_decrease' : [0.0, 0.01, 0.05, 0.1, 0.2, 0.3],

'n\_estimators' : [20,100,1000,2000,5000],

'ccp\_alpha' : [0.1, None]}

#Búsqueda secuencial emplendo validación cruzada k=5

grid = GridSearchCV(estimator=modelo, param\_grid = parametros, cv = 5, n\_jobs=-1, verbose=2)

#Advertencia: la siguiente línea de código conlleva la ejecución de un proceso que puede tomar varias horas. La configuración del modelo con los hiperparámetros resultantes se encuentra más adelante.

grid.fit(X, y)

# Representación de resultados obtenidos

print("\n========================================================")

print(" Resultado de búsqueda por Grid" )

print("========================================================")

print("\n Mejor configuración del estimador entre todos los parámetros ensayados:\n",

grid.best\_estimator\_)

print("\n El mejor índice de regresión ensayado:\n",

grid.best\_score\_)

print("\n Mejores parámetros entre los buscados:\n",

grid.best\_params\_)

print("\n ========================================================")

print("GridSearch:--- %s segundos ---" % (time.time() - m\_time))

print("Total:--- %s segundos ---" % (time.time() - start\_time))

#Definición de hiperparámetros, resultante de la búsqueda secuencial, para modelo ExtraTree

''' modelo= RandomForestRegressor(bootstrap='False', ccp\_alpha=0.1, criterion='mse',

max\_depth=None, max\_features=10, max\_leaf\_nodes=None,

max\_samples=None, min\_impurity\_decrease=0.3,

min\_impurity\_split=None, min\_samples\_leaf=3,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_jobs=None, oob\_score='False',

random\_state=None, verbose=0, warm\_start=True)'''

## Anexo XI: Código Python empleado en la validación del modelo de árboles extremadamente aleatorios (ExtraTrees) obtenido.

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, ShuffleSplit

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor

import time

start\_time = time.time()

pd.set\_option('display.max\_columns', 15)

pd.set\_option('display.max\_rows', 810)

#Importación de conjuntos de datos de la demanda de servicio BiciMAD

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

df5 = pd.concat(chunks, axis=0)

df5=pd.DataFrame(df5)

df5=df5.sort\_values(by='datetime')

df5['cont']=df5.index.astype(int)

#Características que contribuyen

X=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']].astype(float).values

df5x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']]

#Variable a predecir (desplazamientos por día)

y=df5['\_id'].astype(int).values

#División en 60% entrenamiento 20% evaluación 20% validación

X, X\_val, y, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

# train 485 reg, test 162 reg , eval 162 reg

#ExtraTreesRegressor(n\_estimators=100, criterion='mse', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, bootstrap=False, oob\_score=False, n\_jobs=None, random\_state=None, verbose=0, warm\_start=False, ccp\_alpha=0.0, max\_samples=None)

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer,accuracy\_score,r2\_score

modelo= ExtraTreesRegressor(bootstrap=True, ccp\_alpha=0.1, criterion='mse',

max\_depth=None, max\_features=None, max\_leaf\_nodes=None,

max\_samples=None, min\_impurity\_decrease=0.30,

min\_impurity\_split=None, min\_samples\_leaf=2,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_jobs=None, oob\_score='False',

random\_state=None, verbose=0, warm\_start=True)

modelo.fit(X, y)

y\_test\_predict = modelo.predict(X\_val)

resultados=y\_test\_predict

#Obtención de coeficiente de determinación:

r2\_score(y\_val, resultados)

res=pd.DataFrame()

res['predicho']=resultados

res['real']=y\_val

res['errorAbs']=res['predicho']-res['real']

res['errorCuad']=(res['predicho']-res['real'])\*\*(2)

#Visualización del resultado mediante histograma de error absoluto <<< Ilustración 7.31>>>

plt.title('Histograma de errores absolutos (árboles extremadamente aleatorios)')

res['errorAbs'].plot.hist(bins=150, xlim=(-7000,7000), ylim=(0,20),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error absoluto')

plt.show()

#Descripción analítica de error absouto

res['errorAbs'].describe()

#Representación de valores predichos frente a reales <<<Ilustración 7.30>>>

x = [1,30000]

y = [1,30000]

plt.subplot(121)

res.plot.scatter(y=['predicho'], x=['real'], figsize=(10,10), grid=True,s=1)

plt.plot(x,y, color="red", label = 'línea de referencia: tiempo previsto igual al de desplazamiento', linewidth=1 )

plt.title('Valores predichos frente a reales (árboles extremadamente aleatorios)')

plt.xlabel('Valor real')

plt.ylabel('Valor predicho')

plt.show()

#Almacenamiento del modelo en fichero local

from sklearn.externals import joblib

joblib.dump(modelo, 'C:\\MYPATH\\TFG\\Python\\Modelos\\Modelo ExtraTree v1')

#El modelo puede ser cargado y empleado para la predicción de nuevos valores.

modelo = joblib.load('C:\\MYPATH\\TFG\\Python\\Modelos\\Modelo ExtraTree v1')

#Análisis de fechas de mayor error absoluto

por\_fechas=pd.DataFrame()

por\_fechas=res[res['predicho'] > 10000]

por\_fechas['predicho']=por\_fechas['predicho']+660

por\_fechas['errorAbs']=por\_fechas['predicho']-por\_fechas['real']

por\_fechas['errorAbs'].describe()

por\_fechas.head()

por\_fechas2['real']=y\_val

por\_fechas2['predicho']=modelo.predict(X\_val)

por\_fechas2['errorAbs']=por\_fechas2['predicho']-por\_fechas2['real']

por\_fechas2['RECM']=((por\_fechas2['errorAbs'])\*\*2)\*\*(1/2)

por\_fechas2.head()

por\_fechas2[ por\_fechas2['RECM'] >3000]

## Anexo XII: Código Python empleado en la validación del modelo de bosque aleatorio (RandomForest) obtenido.

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, ShuffleSplit

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor

import time

start\_time = time.time()

pd.set\_option('display.max\_columns', 17)

pd.set\_option('display.max\_rows', 810)

#Importación de conjuntos de datos de la demanda de servicio BiciMAD

chunks = []

for chunk in pd.read\_csv('C:\\MYPATH\\TFG\\Datos\\DataSet\\demandaBiciMAD\_v1.csv', chunksize=200, low\_memory=False,sep=','):

chunks.append(chunk)

df5 = pd.concat(chunks, axis=0)

df5=pd.DataFrame(df5)

df5=df5.sort\_values(by='datetime')

df5['cont']=df5.index.astype(int)

#Características que contribuyen

X=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']].astype(float).values

df5x=df5[['year','month','day','weekday\_int','meteo\_tmed','meteo\_prec','meteo\_tmin','meteo\_tmax','meteo\_dir','meteo\_velmedia','meteo\_racha','meteo\_presMax','meteo\_presMin','cont']]

#Variable a predecir (desplazamientos por día)

y=df5['\_id'].astype(int).values

#División en 60% entrenamiento 20% evaluación 20% validación

X, X\_val, y, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

# train 485 reg, test 162 reg , eval 162 reg

#ExtraTreesRegressor(n\_estimators=100, criterion='mse', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, bootstrap=False, oob\_score=False, n\_jobs=None, random\_state=None, verbose=0, warm\_start=False, ccp\_alpha=0.0, max\_samples=None)

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer, accuracy\_score,r2\_score

modelo= RandomForestRegressor(bootstrap='False', ccp\_alpha=0.1, criterion='mse',

max\_depth=None, max\_features=10, max\_leaf\_nodes=None,

max\_samples=None, min\_impurity\_decrease=0.3,

min\_impurity\_split=None, min\_samples\_leaf=3,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_jobs=None, oob\_score='False',

random\_state=None, verbose=0, warm\_start=True)

modelo.fit(X, y)

y\_test\_predict = modelo.predict(X\_val)

resultados=y\_test\_predict

#Obtención de coeficiente de determinación:

r2\_score(resultados, y\_val)

resRF=pd.DataFrame()

resRF['predicho']=resultados

resRF['real']=y\_val

resRF['errorAbs']=resRF['predicho']-resRF['real']

resRF['errorCuad']=(resRF['predicho']-resRF['real'])\*\*(2)

#Visualización del resultado mediante histograma de error absoluto <<< Ilustración 7.31>>>

plt.title('Histograma de errores absolutos (bosque aleatorio)')

resRF['errorAbs'].plot.hist(bins=150, xlim=(-7000,7000), ylim=(0,20),color='#002e5d')

plt.ylabel('Frecuencia')

plt.xlabel('Error absoluto')

resRF['errorAbs'].describe()

plt.show()

#Descripción analítica de error absouto

resRF['errorAbs'].describe()

#Representación de valores predichos frente a reales <<<Ilustración 7.30>>>

resRF.plot.scatter(y='predicho', x=['real'], figsize=(10,10), grid=True,s=1)

x = [1,30000]

y = [1,30000]

plt.plot(x,y, color="red", label = 'línea de referencia: tiempo previsto igual al de desplazamiento', linewidth=1 )

plt.title('Valores predichos frente a reales (bosque aleatorio)')

plt.plot([0,1],[0,1])

plt.xlabel('Valor real')

plt.ylabel('Valor predicho')

plt.show()

#Almacenamiento del modelo en fichero local

from sklearn.externals import joblib

joblib.dump(modelo, 'C:\\MYPATH\\TFG\\Python\\Modelos\\Modelo Random Forest V1')

#El modelo puede ser cargado y empleado para la predicción de nuevos valores.

modelo = joblib.load('C:\\MYPATH\\TFG\\Python\\Modelos\\Modelo Random Forest V1')