TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Análisis de los datos del transporte público de Madrid y deducción de la parada de bajada para cada trayecto

Estudante: Gema Díaz Ferreiro

Dirección: Pablo Gutiérrez Asorey

María Delfina Ramos Vidal

Tirso Varela Rodeiro

A Coruña, septiembre de 2022.

A mamá y a Carmen

Agradecimientos

Por una parte, quiero agradecer a mis tutores Tirso, Delfina y Pablo por estar tan presentes a lo largo de todo el proyecto, por todo el tiempo dedicado y por su infinita paciencia. Por otra parte, agradecer a mi familia y amigos por estar ahí apoyándome durante todos estos meses, confiar en mi y animarme a sacarlo adelante cuando ni yo misma pensaba que podía.

Resumen

El objetivo de este trabajo de fin de grado es deducir las paradas de bajada para cada trayecto en la red de transporte público de Madrid.

Para conseguir este objetivo, primero fue necesario realizar un proceso de análisis, limpieza y transformación de los datos facilitados por el Consorcio Regional de Transportes de Madrid. A continuación, se comenzó con el proceso deductivo aplicando diversos métodos. En primer lugar se aplicaron reglas de decisión y después distintas técnicas de Inteligencia Artificial. Por último, se realizó una comparativa entre los resultados obtenidos mediante ambos métodos a partir de la cual pudimos obtener conclusiones.

Durante la investigación se utilizó SQL Server y SQL para llevar a cabo las reglas de decisión. Para la ejecución de las técnicas de Inteligencia Artificial, se utilizó principalmente *Python* y varias librerias para la gestión de datos y aplicación de técnicas de IA.

Se han conseguido resultados muy prometedores llegando a alcanzar un 87% de precisión con el algoritmo de bajadas basado en reglas de decisión y un 75% de precisión con una de las técnicas de Inteligencia Artificial implementadas.

Abstract

This final dregee project aims to predict the alighting stop for each trip in the public transport network of Madrid.

In order to accomplish this objective, it was necessary to carry out through an analysis, cleaning and transformation process of the data given by the Consorcio Regional de Transportes de Madrid. Next, was when the deductive process started by using different techniques. In the first place, we applied various decision rules, and then we used several artifical intelligence techniques. Lastly, we performed a comparison between the results obtained by these two different methods, and we were able to drew conclusions.

Meanwhile, during the investigation process it was necessary to use SQL Server and SQL to implement the decision rules. To execute the artificial intelligence techniques, we mainly used *Python* combined with some software libraries for the data management and the implementation of the AI methods.

We achieved very promising results reaching a precision of 87% with the decision rules algorithm and a precision of 75% with one of the artificial intelligence techniques implemented.

Palabras clave:

- Análisis de datos
- Predecir
- Parada de bajada
- Reglas de decisión
- Inteligencia artificial
- Random forest
- Red neuronal

Keywords:

- Data analysis
- Predict
- Alighting stop
- Decision rules
- Artificial intelligence
- Random forest
- Neural network

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Motivación

Este trabajo de fin de grado se enmarca en el proyecto SIGTRANS[1] y se centra en el primer contacto con los datos, que incluye el tratamiento y análisis de los datos y en extraer conocimiento que nos permita describir y gestionar el movimiento de la población a través del transporte público. SIGTRANS consiste en un Sistema de Información Geográfica iteroperable y semánticamente enriquecido para el análisis y la gestión del transporte público, donde colaboran la Universidad de A Coruña (UDC) y la Universidad Politécnica de Madrid (UPM).

A lo largo de los últimos años, las tecnologías no han dejado de evolucionar en todos los ámbitos que forman parte de nuestra vida cotidiana. El sector de la movilidad no es una excepción, ya que hoy en día disponemos de tecnologías que nos permiten recopilar la información de cada trayecto como: las tarjetas de transporte de los usuarios, sensores de control de acceso, GPS de móviles y coches o estaciones de monitorización del tráfico. Sin embargo, en este proyecto nos centramos en la información obtenida por medio de las tarjetas de transporte de los usuarios.

La actual emergencia climática, hace que el uso de infraestructuras y servicios para una movilidad sostenible cobren más importancia y, es por ello, por lo que debemos tratar de minimizar el volumen de los gases de efecto invernadero emitidos y que vienen provocados en gran parte por las grandes retenciones de tráfico de las grandes ciudades. Para ello, es importante conseguir una red de transporte público más eficiente, algo que es posible mediante la exploración y explotación de los datos de movilidad, puesto que son estos los que nos ayudarán a realizar los cambios necesarios para implantar mejoras en nuestras redes de transporte.

Debido a esto, es de gran importancia adaptar la red de transporte actual a los movimientos reales de las personas, para permitir a las instituciones facilitar la organización de las estructuras y adecuarse correctamente a las demandas de movilidad de la ciudadanía. Para

[1] Referencia de proyecto: PDC2021-120917-C21

conseguir esto, precisamos deducir la parada en la que se baja cada uno de los viajeros que utiliza el transporte público, a partir de los datos reales facilitados por el Consorcio Regional de Transportes de Madrid (CRTM). Esto es un proceso necesario, ya que solo disponemos de la información que se almacena cuando una tarjeta se escanea en un punto de subida o bajada de un transporte público. En la mayoría de los casos, las tarjetas se escanean al subir, pero no al bajar, por lo que se pierde la información relativa al final del trayecto de muchos viajeros. Para abordar la planificación de la red de transporte necesitamos de datos que representen los trayectos completos de la forma más fiel posible, por lo que debemos estudiar técnicas que nos permitan saber donde podrían bajarse esos viajeros. Además, como parte del proceso de deducción, también identificaremos los viajes y etapas concretas realizadas por cada viajero, es decir, identificar si un trayecto es parte de un viaje más largo.

Como conclusión, podemos afirmar que este proyecto de limpieza, transformación, organización y análisis de datos crudos es un paso básico fundamental a la hora de implementar un sistema de información capaz de ayudar a la toma de decisiones que permita mejorar las redes de transporte público.

1.2 Objetivos

El objetivo principal del proyecto es la deducción de las paradas de bajada para cada trayecto realizado por un viajero. Cabe destacar que la realización de este trabajo es gracias a los datos facilitados por el Consorcio Regional de Transportes de Madrid (CRTM)[2].

Como objetivos derivados del expuesto anteriormente, nos encontramos con los siguientes:

- **Análisis** exhaustivo de los datos dados por Madrid y del estándar que siguen, el estándar General Transit Feed Specification (GTFS). Además, dentro de este análisis, también se detallarán las tareas de limpieza llevadas a cabo sobre el conjunto de datos. Finalmente, se expondrán los datos con los que trabajaremos, así como el formato de los mismos.

- **Diseño** detallado del algoritmo de bajadas y las técnicas de Inteligencia Artificial utilizadas. Se explicará todo el proceso de elaboración y la lógica seguida en cada caso. El algoritmo de bajadas estará basado en la aplicación de un conjunto formado por diversas reglas de decisión. Por otra parte, como técnicas de IA se utilizarán: random forest y una red neuronal.

- **Implementación** necesaria para llevar a cabo el desarrollo del proyecto. En este punto, también se debe detallar la extracción de información, las trasformaciones llevadas a cabo y la carga de los datos un gestor de bases de datos relacionales. Se llevará a cabo

[2] Página web del CRTM: https://www.crtm.es/

la explicación del funcionamiento del servidor OpenTripPlanner (OTP), utilizado durante el trabajo para obtener el tiempo que tarda en recorrer un trayecto y el que tarda caminando desde una parada a otra cercana. Por último, será necesario mostrar ejemplos de la implementación realizada en los algoritmos, además también se realizará una explicación detallada sobre la lógica y funcionamiento de los mismos.

- **Resultados** obtenidos a través del uso de las distintas técnicas tratadas. Se expondrán los resultados así como la explicación de las anomalías que nos podríamos encontrar y una conclusión sobre ellos. También se realizará una comparativa entre varios días con características muy diferenciadas, por ejemplo, días laborales contra días festivos.

Capítulo 2

Fundamentos tecnológicos

En este capítulo, se pondrá en contexto el objetivo de este trabajo respecto a las investigaciones ya existentes en el mismo ámbito en el que se enmarca este proyecto. Para ello, se explicarán en especial detalle aquellos artículos en los que se haga un uso significativo de las mismas técnicas que utilizaremos en este proyecto. Por otra parte, también se detallarán las tecnologías que vamos a usar para el desarrollo del trabajo. Por último, se explicará el estándar General Transit Feed Specification (GTFS), puesto que es el estándar que siguen los datos facilitados.

2.1 Estado del arte

Una vez definido el objetivo del proyecto, se han buscado investigaciones existentes con una finalidad similar. De las publicaciones que hemos explorado en el estado del arte, hemos seleccionado dos de ellas que nos han parecido de especial interés, dado que su objetivo se alineaba perfectamente con nuestras necesidades. Los dos artículos, tratan de deducir la parada de bajada para cada trayecto con dos enfoques distintos, pero en los que se usa el mismo conjunto de datos.

El primero de ellos [1] está basado principalmente en la aplicación de reglas de decisión. Para este estudio, el conjunto de datos del cual disponen incluye las transacciones de tarjetas de viajero, tanto en la parada de subida como en la de bajada, pero en la mayor parte de los casos solamente se dispone de la parada de inicio, ya que la información de destino no suele recogerse en la tarjeta. Por otra parte, de este conjunto se excluían ciertas transacciones: transacciones duplicadas, transacciones en las que faltan datos (ruta, tipo transporte, ...) y tarjetas que solamente cuentan con un viaje. Además, en cuanto excluían una transacción de un viajero, se excluían todas las demás transacciones de esa tarjeta.

Tras detallar los datos de los que disponía el estudio y las técnicas de limpieza de datos

aplicadas, pasamos a los supuestos que se contemplaban para tratar de deducir la parada de bajada. En esta publicación, se basaban en dos suposiciones principales:

- **Principio de continuidad**: un alto porcentaje de pasajeros iniciaban la siguiente etapa desde el final de la etapa anterior.
- **Principio de simetría**: un alto porcentaje de pasajeros regresaban a la misma localización en la que habían iniciado el primer viaje de su día.

Una vez definidas las hipótesis iniciales, disponemos de diversos parámetros ajustables que se deben ir modificando para tratar de obtener los mejores resultados posibles, estos son:

- **Tiempo de transbordo admisible**: tiempo de espera entre la bajada de un medio de transporte y la subida en el siguiente. Se elige de manera arbitraria.
- **Distancia caminando admisible**: la parada se encuentra a una distancia caminando apropiada, puede ser 100,200,300...metros. Se elige una distancia u otra en función de la precisión de los resultados que se vayan obteniendo en cada caso.
- **Suposiciones de último destino**: está directamente relacionada con el principio de simetría. Es la suposición que se elige a la hora de que un usuario regrese al área de la que partió.

Posteriormente, se vio que existe una versión mejorada de la primera hipótesis del principio de simetría y que mejora los resultados. Esta es **last trip-leg**, sigue la misma idea de este principio y consiste en buscar la parada más cercana a la parada de origen y que además pertenezca a la última ruta en la que viajó. Para llevar esto a cabo es necesario utilizar la distancia caminando admisible para averiguar las posibles paradas candidatas cercanas a esa primera parada de subida del día.

Para concluir, los errores de estimación del algoritmo definido en esta publicación se deben a determinar que la última parada del día es la misma que la primera del día (algo que a priori se soluciona aplicando last trip-leg) y las inconsistencias entre los horarios de los servicios y las horas reales de subida y bajada.

Por otra parte, el segundo estudio [2] intenta mejorar los resultados obtenidos por la aproximación de reglas de decisión con un enfoque probabilístico usando redes neuronales. Por tanto, dado que el algoritmo para la reglas está ya abordado, vamos a centrarnos principalmente en las técnicas de Inteligencia Artificial (IA).

Debemos recalcar que para este estudio también se utiliza el mismo conjunto de datos que en el anterior *paper*. Y además, trata de investigar los errores de cálculo para deducir la parada de destino y mejora la precisión del algoritmo del estudio anterior.

Para la deducción, se utiliza un enfoque sin reglas como el deep learning, ya que esperaban que pudiese deducir la parada de bajada con más precisión. Para ello, se crea una lista de paradas de bajada potenciales (paradas coherentes en tiempo y espacio respecto a la parada de subida) para cada transacción que estará compuesta por: 10 paradas previas a la parada de bajada potencial y 10 paradas posteriores. Después, mediante deep learning se elige la parada de destino (potencialmente la más precisa) cuando es la más probable dentro de la lista que se ha creado de paradas potenciales.

Para el deep learning, se utiliza una arquitectura de red neuronal con una sola capa oculta, ya que este tipo de red tiene la capacidad de predecir una salida basada en una combinación compleja y no lineal de variables de entrada.

Una vez aplicada la técnica de IA, se trata de reducir el error de deducción de la parada de bajada y mejorar la precisión. Para ello, se examina la validez de una suposición muy importante: los viajeros se bajan en la parada que está más próxima a su siguiente parada de subida, y una evaluación de las paradas de bajada reales declaró que esta suposición no es necesariamente cierta para todos los casos.

Como conclusión, se relaja este supuesto considerando como posibles paradas de bajada un conjunto de paradas próximas a la parada más cercana a la siguiente subida. Haciendo esto, y aumentando el tamaño del conjunto de entrenamiento, se puede concluir que el algoritmo mejora su precisión.

2.2 Tecnologías utilizadas

Para este proyecto, se han precisado utilizar múltiples tecnologías. Estas van, desde herramientas para la gestión, análisis y modificación de datos utilizando una base de datos relacional, a otras para la gestión de grandes volúmenes de datos y la aplicación de técnicas de IA sobre los mismos. Además, también hemos de destacar el uso de varias librerías para la realización de tareas concretas.

- **Structured Query Language (SQL)**: lenguaje utilizado para realizar consultas a una base de datos relacional, y poder así manipular y acceder a los datos que se encuentran almacenados.
- **SQL Server 2019** [3]: gestor de bases de datos relacionales.
- **Python 3.8.10** [4]: lenguaje de programación de alto nivel con una gran capacidad de análisis y gestión de datos gracias al gran número de librerías que incorpora para este fin.
- **Sublime Text 3** [5]: editor de texto utilizado para escribir todos los script de *python*.

- **Servidor OpenTripPlanner (OTP)**: Servidor utilizado para la obtención de tiempos entre paradas. Se detalla su funcionamiento en la sección 6.2.
- **CSV** [6]: librería de *python* para leer y gestionar datos almacenados en archivos CSV.
- **Pandas** [7]: librería de *python* utilizada para manipular y analizar grandes volúmenes de datos.
- **Sklearn** [8]: librería de *python* que incorpora técnicas de machine learning.
- **TensorFlow** [9]: librería dedicada al aprendizaje automático que permite implementar una red neuronal.
- **Keras** [10]: librería de *python* que incorpora los módulos necesarios para implementar una red neuronal.
- **Matplotlib** [11]: librería para la visualización de datos y obtención de gráficas.
- **Draw.io** [12]: plataforma en línea, utilizada para la elaboración de diagramas.
- **Latex** [13]: sistema de creación de textos, diseñado principalmente para la elaboración de textos estructurados.
- **Overleaf** [14]: editor de texto Latex en línea que permite la escritura colaborativa. Está asociado con las escritura de documentos de carácter científico.

2.3 Estándar GTFS

Los datos facilitados por el Consorcio Regional de Transportes de Madrid (CRTM) [1] siguen el estándar General Transit Feed Specification (GTFS) [2] el cual define un formato común a la hora de publicar la información relacionada con el transporte, y que pueda así ser entendido por toda persona ajena a la agencia que emite cada uno de los datos.

A continuación se describen las entidades mas relevantes para el proyecto:

- *Calendar*: indica la información sobre la disponibilidad diaria (es decir, los horarios de la red), en función del día de la semana, de los servicios de una red. Además, una red de transporte puede tener múltiples horarios al mismo tiempo.
- *CalendarDates*: indica la información sobre excepciones a la planificación general de servicios de Calendar, como podrían ser los días festivos.

[1] https://www.crtm.es/
[2] https://gtfs.org/es/

- *Shapes*: agregación de puntos que siguen una secuencia que representa un recorrido entre dos localizaciones, a cada cual se le asigna un identificador único llamado *shapeId*.
- *Trips*: indican los posibles recorridos de una línea (ida y vuelta). Se utiliza un campo directionId para indicar el sentido del recorrido (0 o 1) y un shapeId para señalar la geometría del mismo. Existe una fila para cada vez que el servicio realiza el recorrido en una semana.
- *Routes*: se corresponde con las líneas que existen para la red de transporte.
- *Stops*: existen tres elementos que podemos identificar como stops. Estos son:
  - Paradas en las que un usuario se puede subir o bajar de un medio de transporte.
  - Estaciones donde se encuentran múltiples paradas.
  - Accesos a estaciones.
- *StopTimes*: indican las horas de llegada y salida programadas de un transporte a una parada para un recorrido (trip) concreto.
- *Frequencies*: informa sobre los intervalos de las salidas de un trasporte en una parada. Además, se indican las horas que limitan el intervalo. Se utilizan en servicios basados en intervalos, como el metro.

Por último, debemos tener en cuenta el proceso seguido para diferenciar ida y vuelta. En el conjunto de datos no se dividen las líneas en función de su sentido, sino que una línea se representa en la tabla *routes* con una única fila. Y mediante la tabla *trips* se obtiene el sentido del recorrido con el campo *directionId*. Además, también podemos obtener el orden de las paradas de una línea para la ida y la vuelta mediante la tabla *stop_times*.

Capítulo 3

Metodología y planificación

En este capítulo, vamos a detallar la metodología de desarrollo elegida, así como la aplicación de esta en nuestro proyecto. También se tratará la planificación inicial del proyecto y el seguimiento realizado sobre la misma. Este último punto, estará apoyado de unos diagramas de Gantt, en los que podremos ver la desviación en tiempo, y para la cual explicaremos las causas.

3.1 Metodología de desarrollo

La metodología elegida para el desarrollo del proyecto ha sido la metodología iterativa incremental [15]. Esta, se caracteriza por dividir todo el desarrollo del proyecto en distintos bloques temporales, o iteraciones.

A lo largo de las distintas iteraciones, se trata de repetir un proceso de trabajo similar, para así conseguir un resultado final más completo. Para llevarla a cabo, cada uno de los requisitos definidos debe completarse en una sola iteración, en la cual se realizan todas las tareas necesarias tratando de realizar un esfuerzo mínimo. Con esto, conseguimos ir desarrollando todas las funcionalidades del proyecto poco a poco, sin correr el riesgo de que las funciones fundamentales se vean comprometidas. Y, además, nos permite tener un producto funcional en cada iteración, sin la necesidad de esperar hasta el final de proyecto para obtener resultados.

La máxima de esta metodología es que este producto irá evolucionando poco a poco, conforme se vayan realizando las entregas incrementales. Por tanto, en cada revisión tendremos nuevos requisitos completados, así como requisitos anteriores mejorados. Debemos recalcar, que esta metodología dirige el desarrollo de forma que ninguno de los requisitos fundamentales queden incompletos, puesto que estos son los que obtienen máxima prioridad.

Es importante resaltar, que hemos usado una metodología basada en *SCRUM*, que es una metodología iterativa incremental. No podemos denominarla *Scrum* como tal, puesto que no existe un equipo de trabajo con roles diferenciados, y, en este caso, al ser un proyecto que se

ha llevado a cabo por una sola persona, consiste en una adaptación de *Scrum*.

Tras la reunión inicial entre tutores y alumna, se decidió utilizar esta metodología ya que viendo los requisitos planteados, inicialmente poco robustos dado que se trata de un proyecto de investigación, parecía la más adecuada. Con ella, logramos priorizar las tareas principales del proyecto: diseño de algoritmo de bajadas y algoritmos de Inteligencia Artificial. Además, también se acordaron reuniones cada 2-3 semanas entre tutores y alumna, para llevar a cabo un seguimiento del trabajo realizado. Estas reuniones se utilizaban para: resolver dudas, realizar correcciones y decidir las nuevas funcionalidades a desarrollar y como abordarlas.

3.1.1 Metodología en el proyecto

Una vez decidida la metodología a utilizar, se realizó la planificación de todas las tareas para llevar a cabo el trabajo. En cada etapa de desarrollo, se aplicaron las fases de: análisis, diseño, implementación y resultados. Estas son las descritas a lo largo de todo el proyecto:

- **Análisis**: se detallan en profundidad los datos facilitados por el Consorcio Regional de Transportes de Madrid y las pasos necesarios para realizar la limpieza y adaptación de los mismos. También se establecen las técnicas a utilizar para la deducción de paradas de bajada.
- **Diseño**: se diseñan los distintos algoritmos de deducción: el algoritmo de bajadas basado en reglas de decisión y las técnicas de Inteligencia Artificial utilizadas.
- **Implementación**: se implementan los algoritmos, características o mejoras diseñadas en el paso anterior.
- **Resultados**: se obtiene la precisión en la deducción de las paradas de bajada para cada iteración.

3.2 Planificación y seguimiento

En esta sección se detalla la planificación inicial del proyecto, así como el seguimiento de la misma. Al final, se muestran dos diagramas de Gantt en los cuales se indican las fechas estimadas y reales de cada iteración, así como los motivos de la desviación en tiempo respecto a la planificación inicial.

3.2.1 Planificación

El primer punto a abordar, es decidir cómo vamos a cumplir los objetivos fijados en la Sección 1.2. Para ello, es necesario planificar las funcionalidades y tareas necesarias para poder cumplir estas metas propuestas.

Uno de los primeros pasos realizados, fue hacer una reunión inicial entre tutores y alumna, para decidir el alcance final del proyecto. En esta reunión se estableció la fecha máxima de entrega y a partir de ahí se definieron las tareas necesarias para cumplir con los objetivos y la prioridad de las mismas.

Una vez definido el alcance, se dividió el proyecto en varias iteraciones:

1. **Primera iteración**: en esta iteración se engloban todas las tareas de documentación y análisis previo al desarrollo del propio trabajo:
    - Estudio del estado del arte sobre las técnicas de deducción de paradas de bajada a través de la lectura de varios artículos de investigación relevantes para este proyecto.
    - Análisis de los datos enviados por el Consorcio Regional de Transportes de Madrid. Se estudió la forma en la que venían dados, para así poder entender las adaptaciones que fueron necesarias realizar para poder trabajar con ellos.
2. **Segunda iteración**: en esta iteración se elaboran las reglas de decisión para crear el algoritmo de bajadas, explicado en detalle en la Sección 5.1. En este algoritmo, se realizará la predicción de las paradas de bajada candidatas y se diferenciarán los trayectos en etapas y viajes.
3. **Tercera iteración**: en esta iteración se elabora el algoritmo para realizar la deducción de las paradas de bajada mediante técnicas de Inteligencia Artificial, explicado en detalle en la Sección 5.2. Como técnicas de IA se utilizan: random forest y redes neuronales.
4. **Cuarta iteración**: en esta iteración se desarrolla una interfaz para mostrar los datos de manera interactiva. Consistirá en un visor cartográfico, con el cual podemos mostrar sobre el mapa de Madrid los viajes reales combinando las subidas facilitadas y las paradas de bajada deducidas.
5. **Elaboración de la memoria**: esta parte del proyecto consiste en la elaboración de un documento donde se detallan todos los procesos llevados a cabo para la elaboración del trabajo.

Para poder llevar a cabo el proyecto de manera satisfactoria, se necesita de la utilización de varios recursos:

- **Recursos humanos**: consisten en las personas capacitadas para realizar las tareas de un proyecto. En este proyecto el equipo de trabajo está formado por los tres tutores y la alumna. Los tutores llevaron a cabo la tarea de limpieza y adecuación de los datos facilitados por el CRTM. Por otra parte, el mantenimiento y preparación del servidor,

explicado en la Sección 6.2, fue llevado a cabo por una persona ajena al proyecto. Por último, la alumna fue responsable de realizar el análisis, diseño, implementación y resultados.

- **Recursos materiales**: constituye las herramientas, máquinas o equipos utilizados. Este proyecto se ha desarrollado empleando el ordenador portátil de la alumna.

Podemos ver el diagrama de Gantt de la planificación con lo tiempos estimados en la figura 3.1.

3.2.2 Seguimiento

El seguimiento llevado a cabo respecto a la planificación descrita en el apartado anterior, sigue el siguiente orden:

1. Estudio del estado del arte sobre la deducción de paradas de bajada. Después de la lectura de diversos artículos de investigación, se hizo una reunión para ver como incluir y adaptar la lógica de los mismos en nuestro proyecto.

2. Elaborar las reglas de decisión para clasificar los viajes y etapas. En primera instancia, se elaboraron de manera teórica y se discutieron en una reunión.

3. Descargar el entorno para la base de datos, en nuestro caso SQL Server, e importar los datos necesarios para empezar a hacer pruebas.

4. Implementar las reglas de decisión para clasificar las transacciones en viajes y etapas. Posteriormente, se hizo otra reunión en la cual se discutían ya todas las reglas implementadas, así como, la completitud y la exactitud de las mismas.

   Durante este momento, también fue necesario comenzar a realizar peticiones al servidor, explicado en el capitulo 6.2, para calcular los datos necesarios para llevar a cabo las reglas. Por tanto, fue necesario crear un script en *Python*, para poder obtener los datos.

5. Implementar las reglas de decisión para deducir las paradas de bajada. Se realizó una reunión para comprobar, nuevamente, la exactitud y completitud de las reglas.

6. Script con las técnicas de Inteligencia Artificial:

   - Primero, se realizó la limpieza, codificación y normalización de los datos.
   - Después, se implementó la técnica de IA random forest.
   - Por último, se implementó la red neuronal.

Durante este último punto, las reuniones fueron más frecuentes por requerimientos de la estudiante.

Es importante recalcar, que los pasos 4, 5 y 6 se llegaron a elaborar todos simultáneamente en etapas avanzadas del proyecto. Esto sucedió principalmente, debido a la necesidad de obtener los datos con las peticiones al servidor. Estas, eran muy costosas computacionalmente y, por tanto, muy lentas. En base a esto, se decidió realizar el desarrollo del script de IA en paralelo con el algoritmo de bajadas, para así tratar de retrasar el proyecto lo mínimo posible.

La tabla 3.1 muestra los tiempos reales dedicados a cada una de las tareas mencionadas en el seguimiento. Debemos destacar que se incluye también la tarea *capacitación*, la cual contempla todo el tiempo de estudio y documentación por parte de la alumna para poder llevar a cabo todas las demás tareas.

Duración de cada tarea	
Estudio del estado del arte (papers)	02:15 h
Análisis datos Madrid	02:00 h
Capacitación	12:20 h
Entorno de desarrollo	03:20 h
Script peticiones servidor y preparación datos	27:10 h
Script viajes/etapas	21:40 h
Script paradas bajada	11:45 h
Script IA (random forest y red neuronal	15:05 h

Tabla 3.1: Duración real de las tareas realizadas en el seguimiento

Podemos ver el diagrama de Gantt relativo al seguimiento sobre la planificación inicial en la figura 3.2.

Como podemos ver en la figura 3.1 y la figura 3.2, ha habido una desviación importante en el seguimiento respecto de la planificación inicial. Esto se debe a que, aún paralelizando las tareas, el proyecto sufrió retraso en tiempo debido a los cálculos necesarios a realizar con el servidor. Hubo varios errores en los cálculos que provocaron que hubiese que repetirlos con todo el gasto de recursos y tiempo que conlleva. Y también el propio tiempo de espera de la obtención de los datos, hizo que en algunos puntos se parase el desarrollo, ya que no se podía continuar ni obtener resultados, hasta que las peticiones terminasen.

Este hecho, además de provocar el retraso en tiempo, impidió llevar a cabo algunas de las tareas marcadas en la planificación inicial del proyecto 3.2.1. Esto afectó a la cuarta iteración,

donde se realizaba el desarrollo del visor GIS; y ante la imposibilidad de llevarlo a cabo, se decidió marcar como trabajo futuro. En el capitulo 8.2 se describe de manera sencilla en qué consiste, y se propone una implementación básica para desarrollarlo.

A pesar de todo, la iteración 4 no constituía una de las tareas críticas del desarrollo del proyecto. Por lo que finalmente, no dispondremos de una interfaz visual para mostrar los datos, pero hemos logrado cumplir con todos los objetivos marcados en el capitulo 1.2. En este punto, nuestro proyecto es capaz de deducir paradas de bajada con una precisión más que aceptable, tal y como podemos ver en los resultados obtenidos en el capítulo 7. Además, se logró entregar el proyecto en la fecha límite establecida durante la primera reunión.

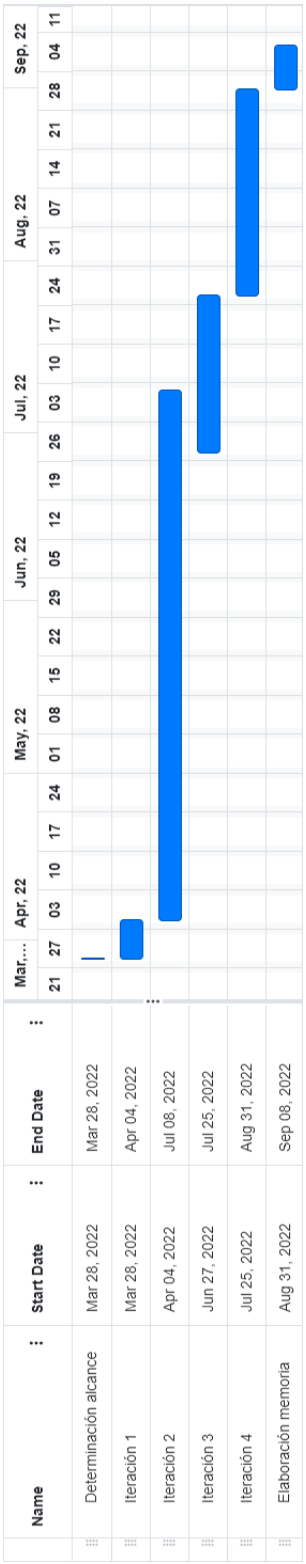

Figura 3.1: Diagrama de Gantt de la planificación inicial

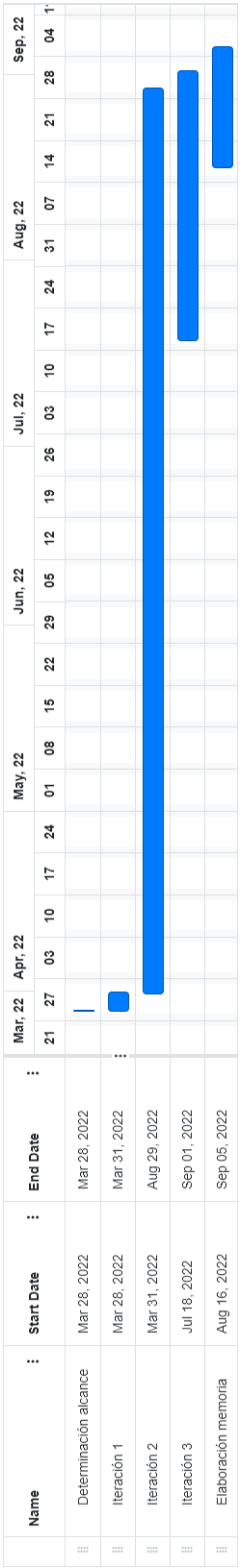

Figura 3.2: Diagrama de Gantt del seguimiento

Capítulo 4

Análisis

En este capítulo, se detallará el análisis exhaustivo que hemos realizado sobre los datos facilitados por el Consorcio Regional de Transportes de Madrid (CRTM)[1]. Además, también se detallarán a grandes rasgos las técnicas que se van a utilizar para la deducción de las paradas de bajada.

4.1 Análisis del conjunto de datos

En esta sección, explicaremos el volumen de datos facilitados por el CRTM y que parte de esto conjunto usaremos para nuestro trabajo. Además, también detallaremos la limpieza que ha sido necesaria efectuar sobre los datos, así como el contenido de los mismos.

4.1.1 Introducción

Todos los datos utilizados para este proyecto provienen del CRTM y siguen el estándar General Transit Feed Specification (GTFS) explicado en la sección 2.3. La información corresponde a todas las transacciones realizas para cada tarjeta de usuario en el transporte público de Madrid, durante los años 2019 y 2021; no disponemos de datos del 2020 debido a la situación anómala en el transporte causada por la pandemia. El conjunto de datos posee más de 250 GiB, por lo que para evitar que las tareas de prueba y análisis se entorpecieran debido al manejo de volúmenes de información tan grandes, se ha decidido usar para este proyecto solamente información relativa al mes de enero de 2019.

Los datos proporcionados sobre las redes de transporte público de Madrid, contienen información sobre los siguientes medios de transporte: Metro, Autobuses interurbanos del municipio de Madrid (EMT), Autobuses urbanos, Autobuses interurbanos de la Comunidad de Madrid, Metro ligero/Tranvía y Cercanías. Además, también hay que recalcar que estos datos

[1] https://www.crtm.es/

están conformados por todo tipo de billetes que ofrece la red de transporte de Madrid: bonos de la tercera edad, bonos turísticos, billetes de viaje único, ...

Poseemos datos de subidas gracias a que se ha generalizado el uso de tarjetas de viajero relacionadas con un usuario especifico en sustitución de los billetes sencillos. Estas tarjetas de viajero se utilizan para validar subidas de usuario a los distintos medios de transporte, no obstante, la mayoría de los medios de transporte no realizan ninguna validación en la bajada. Debido a esto, en nuestros datos encontramos principalmente subidas. Pero, también disponemos de algunas **bajadas**, esto ocurre porque en la mayoría de estaciones de cercanías y algunas estaciones puntuales de otros medios de transporte es necesario validar la tarjeta nuevamente en el momento de bajarse del transporte. Esta información, la usaremos en las técnicas de Inteligencia Artificial para entrenar y medir la precisión con la que deduce el algoritmo. Sin embargo, para las reglas de decisión solamente utilizaremos las transacciones catalogadas como subidas.

Antes de pasar con la siguiente sección, es importante explicar los principios en los que se basará nuestra investigación, ya conocidos en el estado del arte (ver Sección 2.1). Debemos tenerlos en cuenta en el momento de hacer la limpieza de los datos, porque debemos de hacerla de manera adecuada teniendo en cuenta que estos dos principios deben cumplirse en todo momento. Estos son:

- **Principio de continuidad**: indica que un usuario se bajará en la parada más cercana a su próxima parada de subida.
- **Principio de simetría**: indica que un usuario volverá al final del día al lugar desde donde realizó su primera subida. No necesariamente debe ser exactamente la misma parada, sino que puede ser una cercana al origen que coincida en tipo de transporte y linea con la última parada de subida de ese mismo día.

4.1.2 Limpieza de los datos

Una vez aclarados los datos que poseemos para nuestro algoritmo, debemos realizar tareas de limpieza para que en ningún caso, datos anómalos o duplicados comprometan la precisión de la predicción. Para ello, debemos descartar los siguientes datos:

- **Tarjetas turísticas**: como queremos tratar de calcular los movimientos habituales de los usuarios para poder analizar patrones generales de movimiento y uso, este tipo de tarjetas deben descartarse, ya que en ellas se producirán viajes que no siguen un patrón tan claro y, por tanto, podrían contaminar los resultados.
- **Billetes sencillos**: no están asociados a ninguna tarjeta de usuario y, por tanto, no proporcionan ninguna información que podamos utilizar para reconstruir los movimientos

de los usuarios. Esto implica que no se pueden asociar transacciones anteriores y posteriores del mismo usuario.

- **Transacciones duplicadas**: pueden ocurrir por un fallo a la hora de validar la tarjeta o a la hora de almacenar los datos de una transacción.
- **Transacciones que carezcan de un dato importante**: esto ocurriría, por ejemplo, si no disponemos de la parada de subida. Como consecuencia de ello, perderíamos la relación entre la transacción anterior y la actual, rompiendo así el principio de continuidad y haciendo que sea imposible predecir ese viaje. Esto se debe a un error en los datos, mediante el cual no se registró correctamente la parada en la que se efectuó la transacción. En estos casos se descartan todas las transacciones de ese día para esa tarjeta.

Un caso de especial interés, que no debemos descartar pero sí tratar de forma especial, es el caso de varias transacciones para la misma tarjeta, pero cuyas subidas ocurren con unos pocos segundos de diferencia, es decir, varias personas pagaron con la misma tarjeta. En este caso, en vez de descartar todas las transacciones para esa tarjeta, nos quedaremos solo con la primera de ellas eliminando todas las demás y manteniendo así, el principio de continuidad entre paradas de subida anteriores y posteriores.

4.1.3 Contenido de nuestros datos

A continuación, debemos definir una serie de conceptos previos, para poder comprender cómo vamos a transformar y almacenar la información facilitada por Madrid, de forma que sea coherente y entendible para nosotros:

- **Estación**: lugar donde paran una o varias líneas de un solo tipo de transporte. Por ejemplo: cuatro caminos, que es atravesada por las líneas 1,2 y 6.
- **Parada**: dentro de una estación, indica que en una linea, perteneciente a un tipo de transporte, se suban o bajen viajeros. Por ejemplo: Sol, parada de cercanías de la línea C-3.
- **Línea**: línea que recorre un medio de transporte. Por ejemplo: linea 402 de autobús interurbano.
- **Etapa** : una etapa es un desplazamiento del usuario dentro de la red de transporte entre una parada de subida y su posterior parada de bajada.

- **Viaje**: un viaje es el conjunto de etapas agrupadas coherentemente (tanto en la dimensión espacial como temporal), en el cual el origen es la primera etapa donde se sube el usuario, y el destino es donde se baja en la última etapa.
- **Origen y destino**: origen (la parada de subida de la primera etapa del viaje) se refiere al inicio del viaje y destino (la parada de bajada de la última etapa del viaje) al final de un viaje.

Basándonos en estas definiciones, tenemos un posible diagrama de datos 4.1 que recoge toda la información ya facilitada, así como los datos que debemos deducir con nuestros algoritmos (diferenciar viajes/etapas y paradas de bajada candidatas).

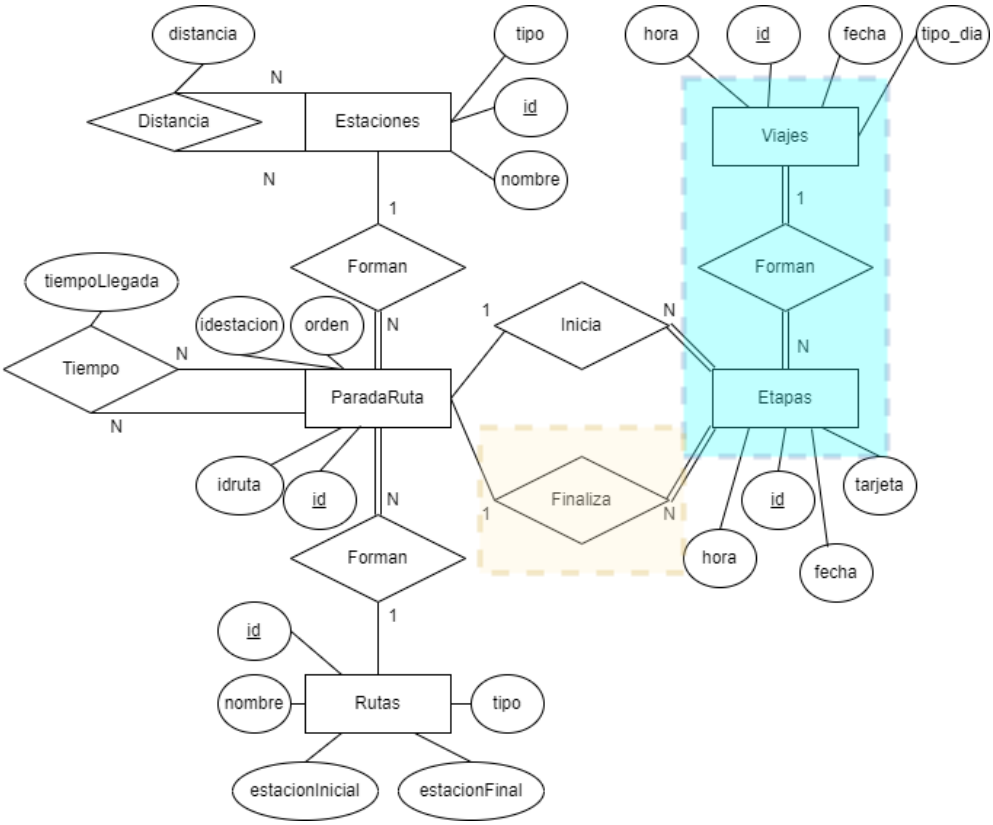

Figura 4.1: Modelo propuesto para relacionar la topología de la red con los propios viajes de usuarios. La parte a completar aparece destacada, en amarillo la estimación de bajadas y en azul la agrupación de etapas en viajes

En los datos facilitados por el CRTM, nos podemos encontrar con los siguientes elementos:

- Una tabla con la información de los viajes realizados por los viajeros. Definida en la sección 4.1.3.

- Una leyenda con los distintos códigos que encontramos como atributos de viajes. Entre estos códigos encontramos códigos para diferenciar medios de transporte o códigos de validación que nos permitieron diferenciar el tipo de transacción.

- Tres tablas que definen la topología de la red de transporte público, definidas en la sección 4.1.3 y la sección 4.1.3. Siendo estas las paradas y líneas correspondientes de trenes, buses interurbanos y buses EMT. Solo disponemos de las líneas en el caso de los buses interurbano y EMT. Además también disponemos de los datos del Consorcio Regional de Transportes de Madrid, que están disponibles de forma abierta en su web y que describen la red de transporte de acuerdo al estándar GTFS.

Viajes

En la tabla de viajes, cada tupla se corresponde con cada validación de una tarjeta o billete en un servicio de transporte público. Tras un análisis preliminar de los datos, observamos que gracias a un campo denominado código de validación, que identificaba las circunstancias específicas de cada validación de una tarjeta o billete, nos era posible clasificar cada transacción en tres tipos: subida, bajada o transbordo. Entendemos por transbordo a un cambio de vehículo obligado por características especiales de la línea, es decir, para continuar viajando en esa línea de la red de transporte, el viajero está obligado a transbordar a otro vehículo. Además, tras realizar la clasificación, fue posible validarla con las autoridades del Consorcio Regional de Transportes de Madrid.

La figura 4.2 muestra una fila de ejemplo, y como atributos más importantes de esta tabla, podemos destacar:

- *user_id*: id de la tarjeta de transporte de un usuario.

- *timestamp*: fecha y hora a la que se detectó la validación de una tarjeta en un punto de pago.

- *dpaypoint*: código que identifica el punto de pago donde se validó la tarjeta. Este identificador normalmente se relaciona con un medio de transporte, una línea y una parada.

- *cod_titulo*: código que indica el tipo de abono asociado a la tarjeta.

- *cod_tlv*: dependiendo del código, indica si la validación ocurrió en metro, cercanías, metro ligero o autobús.

- *cod_transacc*: identifica el tipo de transacción realizada. Existen código para identificar entradas, salidas y transbordos.

user_id	timestamp	dpaypoint	cod_titulo	cod_perfil	cod_descuento	cod_tlv	cod_transacc
3FE3C14BB97DF0F0CEB0EE9A5E3B0359	2019-08-13T23:51:02	02_L8_P8	1052	2	0	C0	1

Figura 4.2: Tupla de ejemplo de la tabla viajes

Por otra parte, tenemos una definición distinta de las tablas que definen la red de transporte, dependiendo del tipo de transporte. Estas tablas tienen varios de sus atributos en común, por lo que pasamos a definirlos:

- *CORONATARIFARIA*: zona tarifaria en la que se encuentra la parada. Las zonas tarifarias son: A, B1, B2, B3, C1, C2.
- *ZONIFICACION*: código de la zona en la que se localiza la parada.
- *IDACTOR*: código del operador del servicio de transporte.
- *IDPARADA*: identificador de la parada.
- *DPAYPOINT*: combinación de los dos atributos anteriores que indica el lugar concreto donde se realizó la validación de la tarjeta.
- *DENOMINAPARADA*: nombre de la parada.

Trenes

En esta tabla se almacena la definición de las paradas de metro, tren, cercanías y metro ligero. En ella, cada tupla indica una parada definida por varios atributos. Los más relevantes son y propios únicamente de esta tabla son:

- *NUMODO*: identifica el tipo de transporte. Siendo: 4 metro, 5 cercanías y 10 metro ligero/tranvía.
- *IDFESTACION*: identificador único de la parada. Se corresponde con el identificador almacenado en los datos abiertos disponibles en la web del Consorcio Regional de Transportes de Madrid y sigue el estándar GTFS.

Sin embargo, en este caso nos podemos encontrar con algunos problemas a la hora de relacionar los viajes con la parada correctamente como:

- Existían filas duplicadas en la tabla para los casos de metro y cercanías refiriéndose a distintas paradas, pero todas relacionadas con el mismo *dpaypoint*. Esto ocurre por como funcionan los puntos de pago de Madrid para los medios de transporte: el viajero valida su tarjeta en un punto de pago y tiene acceso a todas las paradas de dicha estación. Por

esto, el *dpaypoint* es el mismo para todas las paradas, y el problema que nos genera es que a priori no podemos saber en qué parada de metro/cercanías se subió el viajero. Por tanto, en estos medios de transporte, podemos entender el *dpaypoint* como un dato ligado solamente a la estación y no a la parada.

- Existían filas con campos vacíos, incluido a veces el *dpaypoint*. Afortunadamente estas filas se referían siempre a paradas de una estación para las cuales teníamos al menos otra fila indicando su *dpaypoint*.

Para resolver este problema, se consideró a toda la red de metro como una única línea y la totalidad de la red de cercanías como otra única línea.

Autobuses

En estas tablas se almacena la definición de las paradas de autobuses, tanto interurbanos como EMT. En ella, cada tupla indica una parada definida por varios atributos. Tanto para autobuses interurbanos como EMT, nos encontramos los mismo atributos en sus tablas. Los mas relevantes y propios únicamente de autobuses son:

- *IDFEMPRESA*: empresa que gestiona la concesión.
- *CONCESION*: indica la zona que cubre la concesión.
- *IDFLINEAGESTRA*: equivalente al identificador de linea almacenado en los datos abiertos disponibles en la web del Consorcio Regional de Transportes de Madrid y sigue el estándar GTFS.

En el caso de los autobuses, cada fila de la tabla refiriéndose a una parada específica sí tenía un *DPAYPOINT* único, por lo tanto y al contrario que el caso de la tabla de trenes, pudimos considerar todas las lineas de autobuses en la red de transporte público como entidades separadas.

4.2 Partes del proyecto

Basándonos en las aproximaciones encontradas en el estado del arte, este proyecto utiliza dos métodos distintos para tratar de deducir las posibles paradas de bajada; con el objetivo de comparar los resultados de ambos métodos. Por una parte, contamos con el uso de las reglas de decisión y por otra, con la utilización de técnicas de Inteligencia Artificial.

4.2.1 Reglas de decisión

Las reglas de decisión podrían definirse como una serie de condiciones encadenadas que clasifican los datos contenidos en una base de datos. Una vez obtenido un resultado, la regla procede a escribirlo en un campo de destino.

Disponemos de dos conjuntos de reglas claramente diferenciados:

- Reglas para agrupar las etapas en viajes. Estas tienen como objetivo comprobar la coherencia espacial y temporal entre etapas consecutivas para discernir si dichas etapas deben considerarse parte de un mismo viaje o no. La clasificación dentro de un viaje se realiza mediante los identificadores: *Inicio viaje*, *Fin viaje*, *Etapa* y *Trayecto único*.

- Reglas para deducir la parada de bajada de las distintas etapas. En todo momento la parada de bajada candidata generada, cumple con las siguientes condiciones:

  - La parada de bajada es la mas cercana a la subida inmediatamente posterior que pertenezca al mismo medio de transporte que la subida actual.
  - Si el tipo de transporte es autobús (interurbano o EMT), pertenece a la misma línea.

4.2.2 Técnicas de Inteligencia Artificial

Entendemos por Inteligencia Artificial (IA) al uso de algoritmos realizados con el fin de crear máquinas inteligentes que posean capacidades que les permitan llevar a cabo tareas sin la necesidad de supervisión humana.

La aplicación de técnicas de IA, no se realiza para predecir nuevas paradas de bajada, sino para validar el principio de continuidad que dice que una persona que viaje en transporte público, continuará su viaje en transporte público subiéndose al siguiente vehículo en una parada cercana a su anterior parada de bajada.

En este proyecto, hemos aplicado dos técnicas de IA distintas. Hemos elegido random forest porque es una técnica que a priori resulta muy apropiada para aplicar al tipo de datos que tenemos, ya que nuestra variable objetivo es la parada de bajada para la cual existen cientos de clases, y este algoritmo es capaz de manejar la clasificación multiclase directamente. Por otra parte, la red neuronal la hemos elegido puesto que son capaces de aprender y modelar relaciones complejas y no-lineales, por lo que se adaptan muy bien a situaciones reales como es el caso de los datos del transporte público de Madrid, donde la relación entre los datos no está tan clara. Además, en uno de los artículo de investigación [2] se utilizaba esta técnica y arrojaba buenos resultados al estudio.

- **Random forest**: consiste en una técnica clásica de Inteligencia Artificial en la cual se utilizan un gran número de árboles de decisión que trabajan conjuntamente para clasificar un conjunto de datos.
- **Red Neuronal**: está inspirada en el funcionamiento del cerebro humano, y está formada por un conjunto de neuronas artificiales. En este caso, optamos por una red neuronal prealimentada con una única capa, de forma que la configuración de la red neuronal constará de: una capa de entrada, una capa oculta y una capa de salida.

Capítulo 5

Diseño

En este capítulo, vamos a explicar detalladamente el funcionamiento y la lógica de los algoritmos desarrollados para agrupar las transacciones en viajes y deducir las paradas de bajada. Estos son: el algoritmo de bajadas y la aplicación de técnicas de Inteligencia Artificial. También se explicarán los distintos conjuntos de datos que se necesitan utilizar para cada algoritmo y sus motivos.

5.1 Algoritmos de las reglas de decisión

Los algoritmos para agrupar viajes y deducir paradas de bajada se ha realizado en torno a las reglas de decisión (introducidas en la Sección 4.2.1).

Como primer paso para alcanzar los objetivos de este proyecto, debemos calcular las etapas y los viajes de cada usuario. Una vez cumplido este objetivo, pasaremos a la deducción de paradas de bajada para cada etapa. Entendemos que una etapa comprende desde que un usuario se sube a un método de transporte hasta que se baja; y un viaje es una agrupación coherente, tanto en la dimensión espacial como temporal, de etapas. Este paso previo de clasificación de viajes es necesario porque nuestro objetivo final es tener los distintos viajes con las respectivas paradas de bajada para cada etapa. Pero también consiste en un paso necesario previo para poder deducir correctamente las paradas de bajada, ya que debemos saber qué transacciones están relacionadas entre si. Por tanto, al agruparlas en viajes tendremos trayectos en los que sabemos que son aplicables los principios de simetría y continuidad (explicados en la sección 4.1.1) y además, nos permitirá analizar la movilidad en la red de transporte público de los usuarios a nivel individualizado.

Par tanto, el proceso deductivo ha precisado de dos scripts bien diferenciados. El primero de ellos, se encarga de distinguir entre los distintos viajes que puedan existir; y una vez diferenciados, señalar las posibles etapas que pueda poseer cada uno. El segundo de ellos, utiliza la diferenciación de viaje y etapas calculada previamente para tratar de deducir las paradas

de bajada candidatas, basándonos en el principios previamente mencionados.

Para llevarlo a cabo, lo primero que ha sido necesario hacer es cargar toda la información a utilizar en nuestro gestor de bases de datos relacionales, SQL server.

5.1.1 Algoritmo de Viajes y Etapas

Para crear el algoritmo que diferencie entre los distintos viajes y etapas, ha sido necesario crear múltiples reglas de decisión. Estas van a ser capaces de distinguir si una transacción pertenece a un viaje o constituye alguna de las etapas de un viaje. Para esta clasificación utilizaremos las siguientes etiquetas:

- Inicio de viaje: se trata de la transacción de subida en origen de un viaje.
- Fin de viaje: se trata de la transacción de subida hacia el destino de un viaje.
- Etapa: se trata de una transacción intermedia de un viaje. Esta rodeada de otras etapas o inicios/fines de viaje.
- Trayecto único: ocurre cuando solamente existe una transacción asociada a un viaje concreto. Siempre consistirá en una transacción de subida.

Tras esto, pasamos a definir los elementos fundamentales utilizados en la versión final del algoritmo y que debemos tener en cuenta para poder hacer la distinción correctamente:

- Una tabla que posea la relación entre paradas que se encuentren a 200 metros de distancia entre ellas. Lo cual es necesario para que podamos comprobar que las etapas encadenadas en un viaje siguen el principio de continuidad espacial (ver Sección 4.1.1).
- Comprobar si las transacciones a comparar ocurren en tiempos admisibles, cumpliendo así el principio de continuidad en la dimensión temporal. Estos debemos obtenerlos tal y como se explica en la sección 6.2, y son los siguientes:
  - Tiempo que tarda caminando desde una parada a cualquier otra parada que se encuentre a 200 metros o menos de distancia.
  - Tiempo que tarda en llegar desde un parada de subida a la parada de subida inmediatamente posterior, utilizando para ello un método de transporte.
  - Tiempo de espera, el cual será un umbral establecido a mano. Podemos ir variándolo para ver cual es el que arroja los mejores resultados posibles.

Una vez calculados los tiempos y con la tabla de paradas separadas entre sí por 200 metros o menos, podremos aplicar el criterio principal en el que se basarán las reglas. Este se muestra en la figura 5.1 y es: para que una parada de subida (A) pertenezca al mismo viaje que otra

(B), la parada B debe tener paradas a 200 metros que puedan ser paradas candidatas de bajada de A, y además, A y B deben ocurrir en un tiempo admisible; es decir, el tiempo entre las dos transacciones no puede ser menor a la suma del tiempo caminado, tiempo de trayecto y tiempo de espera.

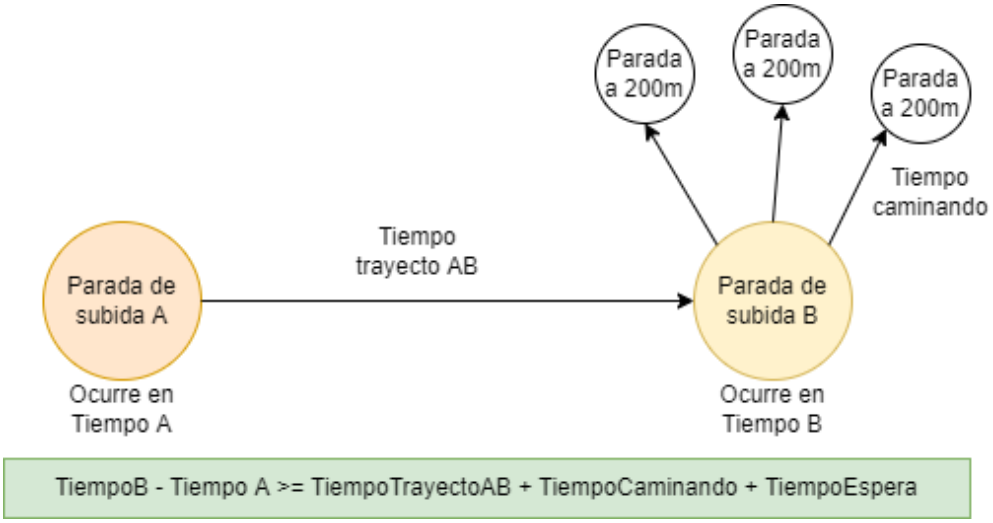

Figura 5.1: Ejemplificación del criterio principal utilizado para la división entre viajes y etapas

Una vez tenemos esto claro, realizamos un experimento preliminar partiendo de un conjunto de datos de unas 100 filas, en el cual podamos ver como están afectando las reglas de decisión que vayamos añadiendo y si se está clasificando correctamente. El criterio general a la hora de crear las distintas reglas ha sido: partir de los casos más simples e ir pasando a los mas complejos una vez vemos cómo los otros van afectando a nuestros datos, posteriormente veremos explicados tanto un ejemplo complejo como uno sencillo. Una vez hecho esto, tras múltiples ejecuciones y comprobaciones, nos daremos cuenta de que existen también casos muy concretos y difíciles de identificar, y para los cuales es necesario crear reglas y condiciones muy especificas. Estas últimas, a priori pueden parecer que tienen una ocurrencia despreciable, pero si trabajamos con un volumen de datos más grande veremos como sí constituyen un número de casos relevantes. Un ejemplo de estos casos, sería el que corresponde a la siguiente regla:

```
IF (trayecto not primeroDia AND trayecto not ultimoDia AND
    subidaSig not parada200m AND tiempoTrayectoAnterior =
    NoAdmisible) THEN trayecto único
```

Listing 5.1: Ejemplo de regla de decisión muy especifica para un trayecto única

Para entender un poco más el razonamiento utilizado a lo largo de todo este script, el caso de la primera transacción del día sería un ejemplo de una regla sencilla y fácilmente

identificable. En ella, la deducción aplicada es la siguiente: el primer trayecto del día solamente puede ser *Inicio de viaje* o *Trayecto único*, por lo que si la subida siguiente tiene paradas a 200m que puedan ser paradas candidatas de bajada de la primera subida de ese día y ocurre en un tiempo admisible será *Inicio de viaje*; y si alguna de las dos condiciones no se cumple, será *Trayecto único*.

5.1.2 Algoritmo de paradas de bajada candidatas

Una vez tenemos la definición de viajes y etapas clasificadas en las etiquetas: *Inicio de viaje*, *Fin de viaje*, *Etapa* y *Trayecto único*; las utilizaremos para tratar de deducir la parada de bajada en cada caso.

Antes de nada, debemos tener en cuenta que si se trata de un trayecto único no hay ninguna deducción posible que podamos realizar, ya que no se cumple ni el principio de continuidad, ni el principio de simetría. Por tanto, nuestras suposiciones se basaran en los casos en los que la transacción se clasifique de alguna de las otras maneras.

Primero de todo, a la hora de enfocar nuestra deducción, debemos tener en cuenta el tipo de transporte utilizado en la parada de subida; y para los autobuses urbanos y EMT, la línea en la que se ha realizado el trayecto.

La condición de la línea solo se aplica a autobuses, porque cercanías y metro se han colapsado previamente en una única red común, salvo algunas excepciones (Sección 4.1.3). Debido a esto, no es relevante que en esos medios de transporte la parada de bajada pertenezca a la misma línea que la subida, puesto que es algo que ya de por sí ocurrirá en la mayoría de los casos.

Tras esto, comenzamos a aplicar los principios en los que se basa nuestro algoritmo, los cuales son el principio de simetría y el principio de continuidad.

Principio de continuidad

El principio de continuidad indica que: tras subirse a un método de transporte, un viajero se bajará en la parada mas cercana en un radio de, por ejemplo 200 metros, de la siguiente parada de subida. Recalcando que esto solamente ocurre, si y solo si la transacción pertenece al mismo viaje que la transacción siguiente, por lo que solo aplica en el caso de *Inicio de viaje* y *Etapa*.

Una vez tenemos este razonamiento, debemos aplicarlo a todos los posibles medios de transporte en los que haya podido subirse un usuario y, por tanto, la parada donde se bajará debe también pertenecer al mismo tipo de transporte. Esto nos permite acotar más las diferentes paradas de bajada candidatas pero, en todo caso, tal y como se dijo antes, se cogerá la parada de bajada candidata más cerca de la siguiente subida que cumpla esta condición.

Además, en el caso de los autobuses (interurbano y EMT) debe cumplirse que pertenezcan también a la misma línea.

Principio de simetría

El principio de simetría indica que: un viajero al final de un día, volverá a la misma zona de la parada de origen y, por tanto, la parada de bajada de la última transacción será igual a la primera parada subida de ese día. A la hora de aplicar esta hipótesis en el proyecto, se han hecho pequeñas modificaciones, ya que en nuestro script solamente la aplicaremos cuando existan al menos dos viajes para un usuario en un mismo día, es decir, dos inicios de viaje con sus respectivos finales de viaje. Además, debemos recalcar que utilizando este principio estaremos deduciendo paradas de bajada solamente para las transacciones etiquetadas como *Fin de viaje*.

Una vez acotado el espectro de opciones en los que este principio es aplicable, debemos tener en cuenta que el método de transporte para la posible parada bajada debe ser consistente respecto a la última subida del día. Para ello, comprobamos que la última subida y la primera de un día utilizan el mismo método de transporte, en caso de ser así, la última bajada será la parada de origen; si esto no ocurre, la bajada será una parada a 200 metros o menos del origen que coincida en método de transporte con esa última transacción y si se trata de un autobús también deberá coincidir la línea.

5.2 Técnicas de IA

El segundo enfoque utilizado para validar el principio de continuidad, ha sido la aplicación de diferentes ténicas de Inteligencia Artificial, tal y como se adelantaba en la sección 4.2.2. Para llevar este algoritmo a cabo, se ha utilizado el lenguaje de programación Python en su versión 3.8.10, asi como diferentes librerias:

- Librerías de gestión de datos: csv y pandas.
- Librerías orientadas a la aplicación de algoritmos de IA: sklearn.
- Librerías más concretas y orientas a la implementación de redes neuronales: tensorflow y keras.
- Librearías para la visualización de datos: matplotlib.

La deducción de paradas de bajada de los usuarios del transporte público es un problema de clasificación multiclase con aprendizaje supervisado. Es **clasificación**, porque el resultado que queremos obtener debe pertenecer a una de las clases existentes. Es **supervisado**, porque los datos están etiquetados y disponemos de la soluciones deseadas para poder entrenar

el algoritmo. También, porque queremos modelar la relación y dependencia entre la variable dependiente y las variables de entrada, para así predecir nuevas paradas de bajada en el futuro basándonos en esas relaciones. Esto se hace porque queremos comprobar que existe una relación entre la parada de bajada de un viaje y la siguiente parada de subida, es decir, podemos deducir que la gente que viaje en trasporte público continúa su viaje en transporte público desde una parada cercana. Por otra parte, normalmente, los problemas de clasificación son binarios porque clasifican los datos en dos clases, pero en este caso la variable dependiente es un id de parada de bajada que puede tomar más de 117 valores, por lo que se transforma en un problema de clasificación **multiclase**.

Primero de todo, tenemos que tener en cuenta el conjunto de datos con el que vamos a trabajar en esta sección. Para abordar un problema de clasificación, vamos a aplicar técnicas de aprendizaje supervisado, por lo que debemos utilizar datos que tengan una etiqueta por la cual clasificar estos datos. En ausencia de una variable objetivo, los algoritmos de aprendizaje supervisado no pueden relacionar los datos de entrada con el *output* (salida) deseado. En este caso, consideramos que nuestra variable objetivo es la parada de bajada, por lo que solo tendremos en consideración para el entrenamiento de este algoritmo aquellas transacciones para los que tengamos los datos correspondientes a la parada de bajada, sino, no podremos enseñarle a los algoritmos a clasificar. Con esto en claro, el conjunto de datos se divide en variables independientes y variables dependientes. La variable dependiente es el *output* que vamos a intentar predecir y las variables independientes son los atributos que usaremos como *input* (entrada) para el proceso de aprendizaje.

Solamente podemos utilizar aquellas transacciones de las cuales dispongamos de su transacción de bajada correspondiente. Esto es necesario, puesto que para entrenar a los algoritmos, debemos pasarle un conjunto de entrenamiento X (input), donde solamente haya información de la subidas. Y un conjunto de entrenamiento Y (output), donde solamente haya información del dato que queramos obtener, en este caso las bajadas.

Para realizar la deducción se han utilizado dos técnicas de IA. Estas son: random forest y una red neuronal. Pero, antes de aplicar alguna de las técnicas, es necesario preparar los datos, realizando las tareas de preprocesamiento apropiadas necesaria y reorganizando los datos para que encajen en la estructura requerida por los algoritmos.

5.2.1 Preparación de los datos

Antes de poder pasar a aplicar cualquiera de las dos técnicas de Inteligencia Artificial mencionadas anteriormente, debemos preparar los datos de una manera determinada. Para ello, debemos codificarlos, colocarlos de una manera determinada y finalmente normalizarlos para cada algoritmo [16].

El primer paso del pre-procesamiento de los datos debe ser la limpieza de los mismos, que

incluye la gestión de los valores nulos en las observaciones. Nuestro conjunto de datos incluye valores nulos en la columnas *tipo_transporte*, *linea* y *stop_id*. Estos tres atributos son críticos para el aprendizaje de nuestro algoritmo, puesto que el *stop_id* es la variable dependiente que queremos predecir. Y en el caso de *tipo_transporte* y *linea*, son atributos estrechamente relacionados con la parada de bajada puesto que una parada de bajada siempre va a estar incluida en la lista de de paradas de una línea determinada y una línea corresponde a un tipo de transporte concreto. Puesto que son tres atributos críticos y nuestro conjunto de datos dispone de millones de filas de datos, decidimos que la mejor forma de gestionar estos valores nulos es eliminar todas las ocurrencias en las que aparezca un valor nulo, teniendo en cuenta que no supone una pérdida significativa comparado con el número total de observaciones de las que disponemos para el aprendizaje.

Codificación

Una vez hecho esto, pasamos a la **codificación** de los datos. En el conjunto de datos, nos encontramos con distintos tipos de variables:

- **Variables numéricas**: las variables numéricas son las que están representadas con, por ejemplo, un float o un int. En nuestros datos, ejemplos de estas variables son: *periodo_hora*, *latitud* o *longitud*.
- **Variables categóricas**: las variables categóricas son descriptivas, y están representadas por cadenas de carácteres. En nuestros datos, ejemplos de estas variables son: *stop_id* o *tipo_tranporte*.

Un paso necesario para el proceso de preparación de datos es codificar las variables de una manera que sean entendibles para el algoritmo. Esto ocurre porque los modelos de aprendizaje automático se basan en procesos matemáticos y muchos de ellos no son capaces de interpretar cadenas de caracteres, por lo que debemos transformarlos en números que el algoritmo sea capaz de manejar. Esto significa, transformar las variables categóricas en numéricas. Para ello se han seguido diferentes criterios:

- En el caso de variables nominales con pocas clases, codificaremos empezando en 0 ayudándonos de la función *factorize*, puesto que tan solo queremos una representación numérica que identifique valores únicos. Esto ocurre para las variables: *tipo_trasporte*, puesto que solo existen 5 posibles clases: metro, metro ligero, bus EMT, bus interurbano y cercanías; y *linea* (8 líneas).
- Por otra parte, las variables con muchas clases, o que ya posean números muy altos para algunas de sus clases, debemos de tratarlas de manera distinta. Esto es necesario, puesto que si disponemos de 1000 clases distintas y comenzamos a codificarlas en 0,

podríamos provocar que el algoritmo diese más importancia a las clases con número más altos realizando así una clasificación errónea. Para ello las siguientes variables se han abordado de esta manera:

- La variable *cod_titulo*, posee en su mayoría ya variables numéricas que parten del millar. Debido a esto, las clases que sí son una cadena de caracteres, comienzan a codificarse a partir del número 3000.
- La variable *stop_id*. En este caso, todas sus clases son categóricas, y empezaremos a codificarlas a partir del 1000, para no sesgar al algoritmo.

Reestructuración del conjunto de datos

Dado que queremos verificar que se cumple el principio de continuidad utilizando técnicas de Inteligencia Artificial (IA), debemos estructurar los datos de forma que podamos evaluar en una única observación la continuidad de toda una etapa. Una etapa comienza en el momento en el que un viajero se sube en una parada, se baja en otra parada distinta, y se vuelve a subir en una última parada parada dar comienzo a una nueva etapa, que enlazamos con la anterior.

El conjunto de datos inicial almacena la información de las tarjetas de transporte cada vez que se valida la tarjeta en una parada de subida o de bajada, de forma que cada línea del conjunto de datos representa una validación única de la tarjeta. Sabemos que la primera validación de una tarjeta corresponde a la subida de un viajero en un medio de transporte, y la validación de la tarjeta siguiente identifica la parada de bajada de dicho trayecto. La etapa inmediatamente posterior comienza con un nuevo escaneo de subida y así sucesivamente. Para representar la continuidad de las etapas en una única observación del conjunto de datos, agrupamos en una misma fila la información del viajero, la información de la primera validación de subida, la parada de bajada de ese trayecto, y la información de subida de la etapa inmediatamente posterior. De esta forma, enlazamos en una única instancia una etapa con el comienzo de la etapa siguiente. Esto es un proceso necesario porque cuando el algoritmo toma los datos lo hace fila a fila, y si no estuvieran colocados de esta manera no los relacionaría de la forma que necesitamos para verificar que cumpla el principio.

Normalización

Tras la reestructuración, será necesario separar la variable objetivo del resto del conjunto de datos:

- **Conjunto de entrada**: conjunto que se le pasa como entrada al algoritmo. En él, dispondremos solamente de la información relativa a las transacciones de subida, tanto la actual como la inmediatamente posterior.

- **Variable objetivo**: variable objetivo, que será la salida del algoritmo. En él, dispondremos solamente de la información que queramos que nuestro algoritmo aprenda a clasificar, en este caso las paradas de bajada.

Antes de ejecutar los algoritmos, debemos de **normalizar** los datos de los conjuntos X e Y. Entendemos por normalizar, transformar los datos numéricos a valores comprendidos entre 0 y 1. Este es un proceso necesario puesto que los valores están medidos en diferentes escalar y se ajustan a una escala común para que la comparación sea más fiable.

Para random forest, normalizaremos todos los valores del conjunto X, menos las variables *stop_id*. Para ello nos ayudaremos de la función *MinMaxScaler*, la cual además nos permite decidir el rango al que escalar. Por otra parte, el conjunto Y, que contiene el *stop_id* de la transacción de bajada, lo dejamos codificado tal y como estaba. Lo mismo ocurre con las variables *stop_id* de las dos transacciones de subida que poseemos. Esto lo hacemos para que no haya incoherencias entre las id del conjunto X y el conjunto Y.

División en subconjuntos

Después, en el momento de entrenar el algoritmo se divide el conjunto de datos en dos. Un subconjunto de entrenamiento y un subconjunto de validación, formado por el 25% de las observaciones del conjunto total. Para entrenar al algoritmo, utilizaremos el conjunto de entrenamiento; y para comprobar que clasifica correctamente una vez entrenado, el conjunto de validación.

A lo largo del proyecto, nos referiremos al conjunto de entrada como X, y a la variable objetivo como Y. Al dividirlos en los subconjuntos de entrenamiento y de validación, el subconjunto de entrenamiento será: X_train e Y_train; y el de validación: X_test e Y_test.

5.2.2 Random forest

Random forest [17] es una de las técnicas clásicas de machine learning y de aprendizaje supervisado más utilizadas, y de las más efectivas. Pertenece al grupo de los métodos de conjunto, o *ensemble*, los cuales combinan múltiples modelos de aprendizaje automático para crear modelos con una capacidad predictiva mayor. Además, es una técnica que ha demostrado arrojar muy buenos resultado para la clasificación y es capaz de soportar la clasificación multiclase. Por esto, es por lo que se ha decidido utilizar esta técnica, porque en un principio es apropiada para las características del conjunto de datos que tenemos.

Random forest consiste en una colección de arboles de decisión que trabaja sobre un conjunto de datos en el que sus elementos están randomizados. Cada árbol de decisión del conjunto es un poco diferente del resto, por lo cual cada uno puede predecir de una manera bastante precisa, pero con tendencia a sobreentrenarse con parte del conjunto de datos. Por tanto, al te-

ner muchos arboles, cada uno se sobreentrenará de una manera distinta, y podremos reducirlo haciendo una media de sus resultados.

5.2.3 Red Neuronal

Las redes neuronales [17], consisten en técnicas de *deep learning*, o aprendizaje profundo, supervisadas. En nuestra caso usaremos redes neuronales alimentadas hacia delante, que consisten en la versión mas clásica y sencilla de las mismas.

Las redes neuronales, son adecuadas para los problemas de clasificación y para trabajar con grandes conjuntos de datos gracias a que son capaces de crear modelos complejos. En nuestro caso, también hemos elegido esta técnica, porque ha demostrado tener muy buenos resultados en problemas de este tipo. De hecho, uno de los artículos de investigación tratados [2], basa su investigación en torno a las redes neuronales.

Una red neuronal es un conjunto de neuronas artificiales, cuya estructura imita a las del cerebro humano. Es capaz de resolver problemas tanto de clasificación como de regresión complejos. Su funcionamiento, consiste en la corrección constante de sus errores para lograr cada vez mejores resultados. Es importante no caer en el sobreentrenamiento, para no sesgar al algoritmo y, para ello, entrenaremos el conjunto por ciclos, fijándonos en el error de clasificación cometido en cada etapa hasta alcanzar un error aceptable.

Nuestra red neuronal cosiste en: una capa de entrada, una capa oculta y una capa de salida. Para una primera aproximación consideramos que es suficiente con una red neuronal de una sola capa oculta, puesto que no queremos añadirle complejidad al problema y, además, se ha demostrado en trabajo anteriores que esta configuración es suficiente para arrojar resultados prometedores [2].

Capítulo 6

Implementación y pruebas

En este capítulo, vamos a explicar cómo se ha hecho la transformación y gestión de los datos del Consorcio Regional de Transportes de Madrid (CRTM), para que estos sean utilizables en nuestro proyecto. Por otra parte, se explicará el funcionamiento del servidor OpenTripPlanner (OTP) para el cálculo de las duraciones entre paradas. Y por último, se mostrarán y explicarán en detalle ejemplos del código desarrollado a lo largo del proyecto.

6.1 ETL

Entendemos por Extract, transform and load (ETL) a un proceso de integración de datos, mediante el cual cohesionamos los datos pertenecientes de diferentes fuentes o aplicaciones. Además, a estos datos también se les aplicarán técnicas de limpieza para posteriormente cargarlos en una base de datos apropiada.

Las partes de ETL, tal y como define el propio acrónimo, son las siguientes:

- **Extracción**: es la primera fase y es altamente importante, ya que nos indica la forma correcta de extraer los datos que vamos a utilizar y cómo prepararlos adecuadamente para la siguiente fase.
- **Transformación**: consiste en relacionar los datos con la idea de negocio que vamos a tratar, para así tener a nuestra disposición los datos que necesitamos y de la manera en la que los necesitamos.
- **Carga**: es la última fase y consiste en cargar en un sistema adecuado, por ejemplo una base de datos, los datos obtenidos en la fase anterior.

Una vez definidas las partes de ETL, veremos la aplicación directa que ha tenido en nuestro proyecto.

Para la fase de **extracción**, hemos recopilado los datos facilitados directamente por el Consorcio Regional de Transportes de Madrid (ver Sección 4.1.1. Una vez hecho esto, seleccionamos los datos con lo que vamos a trabajar finalmente, ya que a priori, sería muy complicado procesar todo el volumen de datos recibido (aproximadamente 250 GiB). Por tanto, se elige Enero del 2019, como el mes del que vamos a utilizar los datos, y más concretamente se utilizaran solo los de unos días concretos, puesto que serán suficientes para probar las validez de las soluciones desarrolladas en este trabajo. Los días utilizados han sido:

- Para las reglas de decisión: los subconjuntos de Enero de los días 1 y 2, y 21 y 22. Se utilizan estos dos conjuntos, para poder comparar la diferencias entre los resultados de días laborables y vacaciones.
- Para las técnicas de Inteligencia Artificial también disponemos de los días 1 y 2, y 21 y 22 de Enero. Pero, además de estos usaremos datos de la semana completa del 21 al 27 de Enero, puesto que, para preparar los datos correctamente se termina descartando mucha información, y así podremos disponer de una muestra final de datos más grande.

Después de definir completamente los conjuntos de datos con los que vamos a trabajar, debemos realizar las tareas de **limpieza y transformación** necesarias. Principalmente serán las siguientes:

- Ignorar todos los *dpaypoint* redundantes en metro y cercanías, es decir, considerar un único *dpaypoint* para todas las paradas en una misma estación de metro y/o cercanías (explicado con mas detalle en la sección 4.1.3).
- Colapsar metro y cercanías en una única macro red. Esto se realiza puesto que con la información facilitada por el CRTM, sabemos qué estación utiliza un usuario, pero una vez está dentro de ella, no sabremos en cual de los posibles lineas que ofrece ha podido subirse.
- Clasificar los códigos de validación en tres categorías tras el estudio de los mismos: entrada, salida y transbordo. Se detallan en el anexo A.
- Encontrar relaciones entre las tablas para poder integrar información adicional de General Transit Feed Specification en nuestros algoritmos.

Finalmente, una vez organizada y transformada las información en base a nuestros requerimientos, debemos **cargarla** en un sistema que nos permita trabajar con ella libremente, en nuestro caso hemos elegido el gestor de bases de datos SQL Server 2019.

6.2 Funcionamiento del servidor Open Trip Planner

Para calcular los tiempos utilizados en el apartado 5.1.1 se ha utilizado OpenTripPlanner (OTP)[1] en la versión 2.1.0. Para ello, se ha extraído una porción del mapa de Open Street Map[2], en concreto la zona de Madrid y alrededores puesto que donde se encuentran las paradas con las que estamos trabajando. Después, se ha procesado con Osmosis[18], el cual genera un fichero Protocolbuffer Binary Format (PBF) que luego utilizará OTP.

Por otra parte, también es necesario descargar los datos General Transit Feed Specification de Madrid desde la página del Consorcio Regional de Transportes de Madrid[3] en formato ZIP y, junto con los ficheros PBF generados anteriormente, se colocan todos en mismo directorio para poder compilarlo con OTP y construir el grafo.

Una vez tenemos el servicio de OTP funcionando correctamente, lo único que nos queda hacer es lanzar las consultas a través de la API, y para esta tarea podemos ayudarnos de la documentación dispuesta para este fin[19].

6.3 Código desarrollado

En esta sección, se describen algunas reglas de decisión explicada en detalle. Estas reglas, pertenecen tanto al script de viajes y etapas como al script de deducción de paradas de bajada. También se detallan las consultas que han sido necesarias realizar para obtener la tabla con la relación de paradas separadas a una distancia máxima de 200 metros.

6.3.1 Script de Viajes y Etapas

Antes de nada, tenemos que destacar que existió una primera versión del algoritmo que utilizaba como momento de la transacción la columna *periodo_hora*. Este atributo, se calculó inicialmente puesto que creíamos que nos ayudaría a simplificar los cálculos de los viajes. Esto en cambio generaba errores, ya que cada *periodo_hora* representaba 10 minutos, y al hacer el cálculo había transacciones muy próximas que se encontraban en diferentes periodos. Por ejemplo, se podía llegar a hacer ver que dos transacciones podían haber ocurrido hasta con 20 minutos de diferencia, cuando la realidad era una diferencia de 1 minuto.

Tras este breve inciso, debemos definir las tablas y atributos que tenemos antes de explicar los ejemplos, para saber cual es la información con la que estamos trabajado y a qué corresponde en cada caso:

- **enero2019**: tabla que contiene los datos de las transacciones para las cuales queramos

[1] https://www.opentripplanner.org/
[2] https://www.openstreetmap.org/
[3] https://www.crtm.es/

obtener resultados, por ejemplo, transacciones de los días 21 y 22 de Enero. Las columnas relevantes de esta tabla son:

  - **t_user**: identificador único de la tarjeta de un usuario.
  - **fecha**: fecha en la que tiene lugar la transacción.
  - **hora**: hora a la que tiene lugar la transacción.
  - **fecha_y_hora**: fecha y hora de la transacción actual formateada como *dateTime*. Esto se hace para que no haya problemas en caso de que una transacción pertenezca a un mismo viaje, pero ocurra después de las 00:00.
  - **stop_id**: parada donde se sube el usuario.
  - **duracion_aParadaSiguiente**: tiempo que se tarda en llegar desde la subida de la transacción actual a la parada de subida de la transacción siguiente, utilizando para ello un medio de transporte.

- **v_datosTransac**: vista con la que añadimos datos de las trasacciones adyacentes a la trasacción actual. Por ejemplo: parada de subida de la siguiente transacción o hora a la cual tienen lugar la transacción anterior y posterior. Las columnas relevantes de esta tabla son:

  - **duracion_desdeParadaAnterior**: tiempo que se tarda en llegar desde la parada de subida de la transacción anterior a la parada de subida de la transacción actual, utilizando para ello un medio de transporte.
  - **fecha_y_horaAnterior**: fecha y hora de la transacción inmediatamente anterior a la actual formateada como *dateTime*.
  - **fecha_y_horaPosterior**: fecha y hora de la transacción inmediatamente posterior a la actual formateada como *dateTime*.
  - **subida_sig**: parada de subida de la transacción inmediatamente posterior a la actual.
  - **duracionWalkActual**: tiempo máximo caminando entre la parada de subida actual, y cualquiera de las paradas a 200 metros que se relacionan con ella.
  - **duracionWalkSig**: tiempo máximo caminando entre la parada de subida de la transacción siguiente, y cualquiera de las paradas a 200 metros que se relacionan con ella.

- **distanciaParadas200m**: tabla en la que tenemos la relación de paradas separadas entre sí a una distancia máxima de 200 metros, y cuanto se tarda en llegar caminando de una a otra. Las columnas relevantes de esta tabla son:

  - **duracion**: tiempo que tarda en llegar caminando de un parada a otra, siempre y cuando estas se encuentren separadas por una distancia máxima de 200 metros.
  - **id_origen**: parada la cual tiene, al menos, una parada a 200 metros de distancia en la tabla *distanciaParadas200*.

- **@tiempo**: tiempo de espera. Se elige de manera arbitraria y para la ejecución es de 20 minutos.

Una vez, explicada cada tabla y atributo que nos podamos encontrar en los ejemplos, podemos proceder a la explicación detalla de los mismos.

El ejemplo 1, que vemos a continuación, trata de una regla de decisión para identificar finales de viaje. Para ello, lo primero que se hace es seleccionar los datos que necesitemos de la tabla *v_datosTransac* y posteriormente un *join* a la tabla que contiene las paradas a 200 metros de distancia. Este *join*, se hace sobre el *stop_id* de la transacción actual al *id_origen*, puesto que de esta forma obtendremos resultados si la parada de subida actual tiene paradas cercanas a 200 metros o menos y, por tanto, la transacción anterior puede estar relacionada con la actual.

Después, se marcan las demás condiciones que se deben cumplir para que sea fin de viaje. En este caso son:

- La diferencia entre la fecha y la hora de la transacción anterior y la actual, debe ser menor o igual a la suma de *duracionWalkActual*, *duracion_desdeParadaAnterior* y el tiempo de espera (*@tiempo*).

- La diferencia entre la fecha y la hora de la transacción actual y la siguiente, debe ser estrictamente mayor a la suma de *duracionWalkSig*, *duracion_aParadaSiguiente* y el tiempo de espera (*@tiempo*).

Debemos tener en cuenta, que en todo momento comprobamos si *fecha_y_horaAnterior* y *fecha_y_horaPosterior* son distintas de nulo. Esto se hace porque en esta regla concreta clasificamos transacciones que no son la última del día, es decir, que se localizan entre otros viajes. Esto es lógico, porque para obtener esos dos datos, se filtra por usuario y día; entonces si fuese la primera transacción del día la *fecha_y_horaAnterior* tendría valor nulo.

```
1 /*
2 * Si el trayecto ocurre en el medio de un día, la parada de subida
    del trayecto actual tiene paradas a 200m,
3 * el trayecto anterior ocurre en un tiempo admisible AND el
    trayecto siguiente ocurre en un tiempo NO admisible
4 * es fin de viaje.
5 */
6 UPDATE [dbo].[enero2019]
```

```
7   SET [TIPO_TRAYECTO] = 'Fin Viaje'
8   from enero2019 t1 inner join (select t1.t_user, t1.fecha,
      t1.hora, stop_id,
9   max(duracion) as duracion from v_datosTransac t1
10  inner join distanciaParadas200 t2 on t1.stop_id = t2.id_origen
11  where (t1.fecha_y_horaAnterior is not null and
      datediff(second,t1.fecha_y_horaAnterior,t1.fecha_y_hora) <=
      t1.duracionWalkActual + t1.duracion_desdeParadaAnterior +
      @tiempo)
12  and (t1.fecha_y_horaPosterior is not null and
      datediff(second,t1.fecha_y_hora,t1.fecha_y_horaPosterior) >
      t1.duracionWalkSig + t1.duracion_aParadaSiguiente + @tiempo)
13  group by t1.t_user, t1.fecha, t1.hora, stop_id)
14  t3 on t3.t_user = t1.t_user and t3.fecha = t1.fecha and t3.hora =
      t1.hora;
```

Listing 6.1: Código del algoritmo descrito para identificar fines de viaje

El ejemplo 2, que vemos a continuación, trata de una regla de decisión para identificar trayectos únicos. Para ello, lo primero que hace es seleccionar los datos necesarios de la tabla *v_datosTrasanc*.

Después, existen dos posibilidades por la cuales, con tal de que una se cumpla, la regla clasificará una transacción como *Trayecto único*. Estas son:

- La parada de subida de la transacción actual y de la transacción siguiente no tienen paradas a 200 metros o menos que puedan ser parada de bajada respecto de la parada de subida anterior. Esto se comprueba viendo si esas paradas existen como *id_origen* dentro de la tabla *distanciaParadas200m*. Con eso conseguimos ver si una transacción esta realmente aislada, porque si la parada actual no tiene paradas a 200 metros implica que no puede ser *Etapa* o *Fin de viaje*; y si la parada de subida siguiente tampoco tiene paradas a 200 metros, significa que la transacción actual tampoco puede ser *Inicio de viaje*.

- Comprobación de que la transacción ocurre en un tiempo no admisible:

  - La diferencia entre la fecha y la hora de la transacción anterior y la actual, debe ser estrictamente mayor a la suma de *duracionWalkActual*, *duracion_desdeParadaAnterior* y el tiempo de espera (*@tiempo*).
  - La diferencia entre la fecha y la hora de la transacción actual y la posterior, debe ser estrictamente mayor a la suma de *duracionWalkSig*, *duracion_aParadaSiguiente* y el tiempo de espera (*@tiempo*).

Es importante recalcar de nuevo, que se realiza la comprobación de si *fecha_y_horaAnterior* y *fecha_y_horaPosterior* son distintas de nulo. Esto es necesario, porque la función de esta regla es clasificar trayectos únicos para transacciones que se encuentren en el medio de un día, es decir, que no sea ni la primera ni la última transacción de un día.

```
/*
* Si el trayecto ocurre en el medio de un día, (la parada de subida
    del trayecto actual AND siguiente NO
* tienen paradas a 200m) OR (el trayeto anterior AND siguiente
    ocurren en un tiempo NO admisible es
* trayecto único)
*/
UPDATE [dbo].[enero2019]
  SET [TIPO_TRAYECTO] = 'Trayecto único'
  from enero2019 t1 inner join (select t1.t_user, t1.fecha,
    t1.hora, stop_id from v_datosTransac t1
  where (t1.subida_sig not in (select id_origen from
    distanciaParadas200) and
  t1.stop_id not in (select id_origen from distanciaParadas200)) or
  ((t1.fecha_y_horaAnterior is not null and
    datediff(second,t1.fecha_y_horaAnterior,t1.fecha_y_hora) >
    t1.duracionWalkActual + t1.duracion_desdeParadaAnterior +
    @tiempo)
  and (t1.fecha_y_horaPosterior is not null and
    datediff(second,t1.fecha_y_hora,t1.fecha_y_horaPosterior) >
    t1.duracionWalkSig + t1.duracion_aParadaSiguiente + @tiempo))
  group by t1.t_user, t1.fecha, t1.hora, stop_id) t2 on t2.T_USER =
    t1.T_USER AND t2.FECHA = t1.FECHA AND t2.HORA = t1.HORA;
```

Listing 6.2: Código del algoritmo descrito para identificar trayectos únicos

6.3.2 Script de paradas de bajada

Tal y como hicimos en la sección anterior, necesitamos explicar las tablas y entidades que van a aparecer en los ejemplos para tener todo el contexto necesario para entenderlos con claridad. Las tablas y atributos que nos podemos encontrar son los siguientes:

- **enero2019**: tabla que contiene os datos de las transacciones para las cuales queramos obtener resultados. Las columnas relevantes de esta tabla son:
  - **t_user**: identificador único de la tarjeta de un usuario.
  - **fecha**: fecha en la que tiene lugar la transacción.
  - **hora**: hora a la que tiene lugar la transacción.

  - **tipo_transporte**: indica el medio de transporte en que se ha subido un usuario. Puede ser: autobús interurbano, autobús EMT, metro, cercanías o metro ligero.
  - **stop_id**: parada donde se sube el usuario.
  - **line_id**: identificador de la linea por la que pasa un tipo de transporte.
  - **nombre_parada**: nombre de la parada de subida.
  - **tipo_trayecto**: indica si un trayecto es de tipo *Inicio de viaje*, *Fin de viaje*, *Etapa* o *Trayecto único*.

- **v_datosTransac**: vista con la que añadimos a la transacción actual la parada de subida de la transacción posterior. Como columna relevante de esta tabla tenemos:
  - **subida_sig**: parada de subida de la transacción inmediatamente posterior a la actual.

- **distanciaParadas200**: tabla en la que tenemos la relación de paradas separadas entre sí a una distancia máxima de 200 metros, y cuanto se tarda en llegar caminando de una a otra. Las columnas relevantes de esta tabla son:
  - **duracion**: tiempo que tarda en llegar caminando de un parada a otra, siempre y cuando estas se encuentren separadas por una distancia máxima de 200 metros.
  - **id_origen**: parada la cual tiene, al menos, una parada a 200 metros de distancia en la tabla *distanciaParadas200*.
  - **id_destino**: parada que está a 200 metros de otra. Conforma la segunda parte de la relación de *id_origen* de la tabla *distanciaParadas200*.
  - **nombre_destino**: nombre de la parada que posee el id *id_destino*.
  - **ruta_destino**: linea a la que pertenece la parada *id_destino*.

- **primeraYUiltimaVariosViajes**: tabla temporal que devuelve el primer inicio de viaje y el último fin de viaje de cada día, si para ese día existen al menos dos viajes, es decir, dos inicios y dos finales de viaje.

- **v_subidaPrimera_es_bajadaUltimo**: vista que utiliza el primer inicio y fin de viaje de la tabla *primeraYUiltimaVariosViajes* y asocia cada subida del primer inicio de viaje como parada de bajada del cada último fin de viaje. Las columnas relevantes de esta tabla son:
  - **ID_anterior**: parada de subida de la transacción inmediatamente anterior a la actual. En decir, el id del primer inicio de viaje de un día el cual tenga al menos dos viajes.

  - **nombre_anterior**: nombre de la parada de subida de la transacción inmediatamente anterior a la actual. En decir, el nombre del primer inicio de viaje de un día el cual tenga al menos dos viajes.
  - **tipo_transporte_anterior**: medio de transporte utilizado en la transacción inmediatamente anterior a la actual. En decir, el tipo de transporte del primer inicio de viaje de un día el cual tenga al menos dos viajes.

Por otra parte, para identificar el medio de transporte utilizado, nos podemos valer de varios atributos:

- *tipo_tranporte* de la tabla *enero2019*. Identifica los medios de transporte por su nombre: *BUS EMT*, *BUS INTERURBANO*, *METRO METRO LIGERO* O *CERCANÍAS*.
- *id_destino* de la tabla *distanciaParadas200*. En este caso, no tenemos el nombre, sino que tiene la mismo estrucutra que el *stop_id*, pero fijándonos en su estructura podemos ver de que tipo de transporte se trata:
  - BUS INTERURBANO: sigue la estructura par_8.
  - BUS EMT: sigue la estructura par_6.
  - METRO: sigue la estructura par_4, est_4, par_90 o est_90.
  - CERCANÍAS: sigue la estructura par_5.
  - METRO LIGERO: sigue la estructura par_10.

Tras haber definido el contexto necesario para entender correctamente los ejemplos propuestos a continuación, pasamos a definirlos de manera detallada.

El primero de los ejemplos que podemos ver consiste en la aplicación directa del principio de continuidad (previamente explicado en la sección 5.1.2. En este caso, se aplica a las transacciones cuyo medio de transporte es autobús EMT. En la *query* tenemos una subconsulta anidada que crea una tabla a la que luego accederemos, por lo que para explicar el funcionamiento, vamos a empezar por descomponer esta subconsulta.

Tal y como vemos en la linea 18 del código, la tabla resultante de este subconsulta recibe el nombre de *paradaSeleccionada*, así que vamos a referirnos así a ella a lo largo de toda la explicación. Lo que estamos haciendo es tratar de seleccionar la parada a 200 metros o menos de la siguiente parada de subida con la menor duración caminando desde esa siguiente parada de subida, la cual pertenece a la tabla *distanciaParadas200*. Después de esto, se le aplican las condiciones necesarias, para que además de poseer la duración mínima cumpla con los requerimientos oportunos:

- Dado que se trata del principio de continuidad, solamente puede aplicarse a tipos de trayecto etiquetados como *Etapa* o *Inicio de viaje*.

- Debemos comprobar el medio de transporte y, en este caso, esta regla clasifica transacciones de bus EMT. Para ello, debemos comprobar que *tipo_transporte* es efectivamente 'BUS EMT' y que la parada de la tabla *distanciaParadas200m* con la menor duración corresponda a BUS EMT. Para hacer esto último, debemos de fijarnos en la estructura del id de la parada, el cual debe tener un 6 como primer dígito, es decir, par_6.

- Como se trata de bus EMT, también debemos tener en cuenta la linea. Para ello, nos fijamos en el valor de *line_id* en la transacción de subida y comprobamos que este sea el mismo a *ruta_destino* que será el id de linea correspondiente a la parada a 200 metros con menor duración respecto a la parada de subida actual.

Después de esta subconsulta tendremos la *paradaSeleccionada* que tendrá las transacciones originales y la duración mínima de lo que tardaría en llegar a la parada de bajada candidata. Pero, todavía necesitamos relacionar la subida actual con la parada de bajada a 200 metros o menos de la subida siguiente, y a esta parada de bajada es a la que corresponde la duración mínima caminando obtenida con *paradaSeleccionada*. Esto lo hacemos haciendo un *join* con la tabla *distanciaParadas200m* relacionando la *subida_sig* con *id_origen* (parada a 200 metros o menos), cuya duración sea mínima y, por tanto, coincida con la que hemos calculado en la subconsulta de *paradaSeleccionada*.

```
/*
* BUS EMT
* SI (tipoTrayecto = etapa OR tipoTrayecto = Inicio Viaje)
* ENTONCES paradaBajada = parada200m con distancia minima de
    siguiente parada Y del mismo tipo que la parada de subida
* Se esta calculando para todos, pero en el caso de los buses o
    metro, no estoy teniendo en cuenta si esa parada pertenece a la
* misma linea (no tengo este dato en la tabla de distancias)
*/
UPDATE [dbo].[enero2019_grande]
  SET [ID_BAJADA] = t3.id_destino, [NOMBRE_PARADA_BAJADA] =
    t3.nombre_destino
  from enero2019_grande t1 inner join (select t_user, fecha, hora,
    tipo_transporte, stop_id, nombre_parada,
  tipo_trayecto, paradaSeleccionada.duracion, id_destino,
    nombre_destino from (select T_USER, FECHA, HORA,
    TIPO_TRANSPORTE, stop_id, line_id, NOMBRE_PARADA,
  TIPO_TRAYECTO,subida_sig, min(duracion) as duracion from
    v_datosTransac t1
  inner join distanciaParadas200 t2 on t1.subida_sig = t2.id_origen
  where (TIPO_TRAYECTO = 'Etapa' or TIPO_TRAYECTO = 'Inicio viaje')
  and SUBSTRING(id_destino, 5, 1) = '6' and Tipo_transporte = 'BUS
    EMT'
  and (t1.line_id != '' and t1.line_id = t2.ruta_destino)
```

```
17    group by T_USER, FECHA, HORA, TIPO_TRANSPORTE, stop_id, line_id,
        NOMBRE_PARADA,
18    TIPO_TRAYECTO, subida_sig) paradaSeleccionada
19    inner join distanciaParadas200 t2 on
        paradaSeleccionada.subida_sig = t2.id_origen and
        paradaSeleccionada.duracion = t2.duracion)
20    t3 on t3.t_user = t1.t_user and t3.fecha = t1.fecha and t3.hora =
        t1.hora;
```

Listing 6.3: Código del algoritmo descrito para deducir paradas de bajada cuando la subida se ha realizado al tipo de transporte bus EMT

El segundo ejemplo, dispuesto a continuación, detalla la vista necesaria para poder aplicar todas las reglas relacionadas al principio de simetría, explicado en la sección 5.1.2. De hecho, en esta vista reside toda la lógica de este principio, puesto que las reglas siguientes utilizaran la información obtenida en la vista y aplicarán las condiciones de tipo de transporte y línea, en el caso de los autobuses, para obtener la parada de bajada en cada caso.

Para empezar, tenemos que segmentar la vista a la hora de explicarla, puesto que se utilizan múltiples tipos de query. Primero, comenzaremos con la tabla temporal *primeraYUltimaVariosViajes*. Dentro de ella, nos encontramos con dos consultas principales relacionadas con una cláusula *union*. La primera de ellas relativa a los inicios de viaje y la segunda a los finales de viaje. Vamos a comenzar analizando la referente a los inicios de viaje y, dentro de ella, la subconsulta que abarca desde la linea 12 hasta la 14, y está nombrada como *t3* (mas2iniciosViaje).

La tabla *t3* (mas2iniciosViaje) que obtenemos, contiene las transacciones cuyo tipo de trayecto es *Inicio de viaje*, si para un día y un usuario concreto existen al menos dos trayectos con esta etiqueta.

Después, pasamos a la tabla *t4* (primerInicioViaje), que comienza en la linea 11 y termina en las 16. En esta consulta, lo que estamos haciendo es coger los datos calculados de *t3* (mas2iniciosViaje); y de todos los Inicios de viaje seleccionados, cogemos solamente el que ocurra el primero en un día y para un usuario concreto.

Por último, seleccionamos todas las transacciones de *enero2019* y hacemos un *join* con la tabla *t4* (primerInicioViaje), para así, seleccionar solamente las transacciones que estamos buscando, que son las primeras transacciónes *Inicio de viaje* de un usuario para los días con al menos dos inicios de viaje.

Una vez explicada la primera de las consultas, que compone la primera clausula de la función *union*, tenemos la segunda cuyo funcionamiento es muy similar. En esta realizamos lo mismo para las transacciones etiquetadas como *Fin de viaje*, con la única variación de que seleccionamos la transacción del usuario que sea final de viaje y la última de un día, para días

con al menos dos finales de viaje.

Con esto, damos por terminada la explicación de la tabla temporal *primeraYUltimaVariosViaje* y procedemos a detallar el funcionamiento de la consulta que va a devolver el resultado final de la vista *v_subidaPrimera_es_bajadaUltimo*. En esta consulta lo que hacemos esencialmente es asociar como parada de bajada del último fin de viaje de un usuario en un día, el *stop_id* y el *nombre_parada* del primer inicio de viaje de ese día y ese usuario. Para ello, utilizamos la funcion *lag* de SQL Server, que devuelve el dato que busquemos de la transacción localizada en la posición inmediatamente anterior a la actual, filtrando por día y usuario. De esta manera, como solamente tenemos para cada día y usuario el primer inicio de viaje y el último fin de viaje, le asignamos a ese fin de viaje los datos necesarios del inicio de viaje.

```
GO
/*
* Selecciona el primer inicio de viaje y el último fin de viaje (si en un dia existen mas de un viaje) y asocia la subida del primer inicio
* de viaje del día como bajada del último fin de viaje del día
*/
Create View v_subidaPrimera_es_bajadaUltimo as
  /*
  * Devuelve el primer inicio de viaje y ultimo fin de viaje de cada día si para ese día existen al menos dos viajes (dos inicios y dos finales).
  */
  with primeraYUltimaVariosViajes as (select t1.* from enero2019_grande t1
  inner join (select t2.t_user, t2.fecha,  min(hora) as hora from enero2019_grande t2
  inner join (select t_user, fecha, count(*) as cantidad from enero2019_grande where TIPO_TRAYECTO = 'Inicio Viaje'
  group by t_user, fecha
  having count(*) > 1) t3 on t2.t_user = t3.t_user and t2.fecha = t3.fecha
  where TIPO_TRAYECTO = 'Inicio Viaje'
  group by t2.fecha, t2.T_USER) t4 on t1.t_user = t4.T_USER and t1.fecha = t4.FECHA and t1.hora = t4.hora
  union
  select t5.* from enero2019_grande t5
  inner join (select t6.t_user, t6.fecha,  max(hora) as hora from enero2019_grande t6
  inner join (select t_user, fecha, count(*) as cantidad from enero2019_grande where TIPO_TRAYECTO = 'Fin Viaje'
  group by t_user, fecha
  having count(*) > 1) t7 on t6.t_user = t7.t_user and t6.fecha =
```

```
    t7.fecha
23  where TIPO_TRAYECTO = 'Fin Viaje'
24  group by t6.fecha, t6.T_USER) t8 on t5.t_user = t8.T_USER and
    t5.fecha = t8.FECHA and t5.hora = t8.hora)
25  /*
26  * Se asocia a los ultimos trayectos final de viaje, el id y
    nombre de la parada del primer inicio de viaje
27  */
28  select t_user, fecha, hora, tipo_transporte, stop_id,
    nombre_parada, tipo_trayecto,
29  lag(stop_id, 1) over (partition by t_user, fecha order by fecha)
    ID_anterior,
30  lag(nombre_parada, 1) over (partition by t_user, fecha order by
    fecha) Nombre_Anterior,
31  lag(tipo_transporte, 1) over (partition by t_user, fecha order by
    fecha) as tipo_transporte_anterior
32  from primeraYUltimaVariosViajes
33 GO
```

Listing 6.4: Código del algoritmo descrito para obtener la relación entre la primera y última transacción de un día para un usuario

6.3.3 Tabla de paradas a 200 metros

Para obtener las paradas separadas a 200 metros entre si, se utilizan los datos en abierto del Consorcio Regional de Transportes de Madrid (CRTM) que siguen el estándar General Transit Feed Specification (GTFS).

Inicialmente, se realizó una consulta más sencilla mediante la cual se obtenía la relación entre paradas que estuviesen a una distancia máxima de 200 metros. Para ello, fue necesario hacer un *cross join* de la tabla de paradas para poder relacionar todas las paradas con todas. Después de esto, se aplicó la condición para que la distancia entre las paradas no superase los 200 metros y se excluyeron las relaciones de paradas consigo mismas para evitar datos redundantes.

```
1 select distinct o.stop_id as id_origen, o.stop_name as
    nombre_origen, o.stop_loc as punto_origen, d.stop_id as
    id_destino, d.stop_name as nombre_destino, d.stop_loc as
    punto_destino,ST_Distance(o.stop_loc::geography,
    d.stop_loc::geography) AS distancia
2 from t_stops o cross join t_stops d
3 where o.stop_id != d.stop_id and
    (ST_Distance(o.stop_loc::geography, d.stop_loc::geography) <=
    200)
```

Listing 6.5: Consulta básica para la obtención de la tabla *distanciaParadas200*

Después de la primera versión de la tabla, se vio que para algunos casos era necesario tener la línea que correspondía a cada parada. Esto es necesario en el caso del autobús interurbano y el autobús EMT, ya que ahí si que son relevantes las distintas líneas existentes.

Esta consulta, accedía a demasiadas tablas, lo cual la hacía muy compleja y que requiriese de una capacidad computacional muy alta. Para tratar de minimizar este impacto, y hacer que la petición fuese más rápida, se crea una tabla auxiliar intermedia, *stop_routes*.

En esta tabla, tenemos la relación entre el *stop_id* y la línea o *route_id* al que pertenece.

```
select distinct o.stop_id as id_origen, o.stop_name as
    nombre_origen, o.stop_loc as punto_origen, tro.route_id as
    ruta_origen, d.stop_id as id_destino, d.stop_name as
    nombre_destino, d.stop_loc as punto_destino, trd.route_id as
    ruta_destino,ST_Distance(o.stop_loc::geography,
    d.stop_loc::geography) AS distancia
from t_stops o join t_stop_times tsto on tsto.stop_id = o.stop_id
               join t_trips tto on tto.trip_id = tsto.trip_id
               join t_routes tro on tro.route_id = tto.route_id
               cross join t_stops d
               join t_stop_times tstd on tstd.stop_id = d.stop_id
               join t_trips ttd on ttd.trip_id = tstd.trip_id
               join t_routes trd on trd.route_id = ttd.route_id
where o.stop_id != d.stop_id and
    (ST_Distance(o.stop_loc::geography, d.stop_loc::geography) <=
    200)
```

Listing 6.6: Consulta para la obtención de la tabla *distanciaParadas200* que contiene la línea asociada a cada parada

6.3.4 Técnicas de inteligencia artifical

Antes de nada, debemos explicar los parámetros que se van a utiliza en el momento de dividir los conjuntos X e Y en los conjuntos de entrenamiento y de prueba (*X_train, X_test, Y_train e Y_test*:

- Aplicamos un *random state* de 42. Se elige este valor, porque de no indicar nosotros un valor fijo, cada vez que ejecutásemos el código se generaría un conjunto de datos diferente al aleatorizar los datos en la división, de forma que sería imposible replicar los experimentos. Al fijar un número como semilla del randomizador nos aseguramos de que siempre se genere el mismo conjunto de datos, y elegimos 42 por su extendido uso en la comunidad.

- Aplicamos un tamaño del conjunto de test de un 25% respecto al conjunto X o Y total, y elegimos este tamaño porque es el predefinido por *sklearn*. Tras probarlo consideramos

que devuelve valores aceptables proporcionando un conjunto test diverso y adecuado para comprobar la precisión del algoritmo.

Random forest

Para obtener los resultado del algoritmo Random forest, se ha seguido la siguiente configuración de sus parámetros:

- Para random forest, utilizamos un tamaño de 100 arboles de decisión. Tomaos este valor porque es la configuración predeterminada de *sklearn*, y tras probarlo decidimos que obtuvimos resultados aceptables.

```
x_train_RF, x_test_RF, y_train_RF, y_test_RF =
    train_test_split(x_RF, y_RF, test_size=0.25, random_state=42)

clf = RandomForestClassifier(n_estimators = 100)


clf.fit(x_train_RF, y_train_RF)
```

Listing 6.7: División de X e Y en los subconjuntos de validación y entrenamiento y implementación de random forest

Red Neuronal

Para obtener los resultado de la red neuronal, se ha seguido la siguiente configuración de sus parámetros:

- El número de neuronas de la **capa de entrada**, es igual al número de columnas en nuestro conjunto datos. En nuestro caso, es de 24.
- En la literatura se encuentran diversas recomendaciones para elegir un buen número de neuronas en la **capa oculta**, como que debe ser uno entre el número de neuronas de la capa de entrada y el número de neuronas de la capa de salida y no debe superar el doble del tamaño de la capa de entrada. Por esto, decidimos que un buen tamaño para la capa oculta se obtiene doblando el número de neuronas de la capa de entrada, por lo que constará de 48.
- El número de neuronas de la **capa de salida** es igual al número de clases que hay en el conjunto Y. Es decir, el número posible de paradas de bajada que existen en las transacciones de bajada que tenemos. En este caso es de 118.
- Las **funciones de activación** que usamos son las siguientes:

  - Para la **capa oculta** usamos la función de activación *Rectified Linear Unit (ReLU)*. Primero, se hicieron pruebas con las funciones *Sigmoid* y *Tanh* y la ReLU es la que mejores resultados arrojaba en nuestro caso. Además, ReLU es la función más popular a día de hoy porque supera algunas limitaciones de las otras funciones de activación porque es menos susceptible al problema de desvanecimiento de gradiente, el cual dificulta el entrenamiento de las redes neuronales.
  - Para la **capa de salida** usamos la función de activación *softmax*. Normalmente se usa la función *softmax* como la última función de activación de la red neuronal porque normaliza la salida en una distribución de probabilidades más fácil de interpretar. Además, da soporte a problemas de clasificación multiclase, como es nuestro caso.

- En la compilación del algoritmo, usamos como **función de optimización** *ADAM*. Tras probar otras funciones de activación como *SGD* o *RMSprop*, *ADAM* es la que ha arrojado mejores resultados para nuestro problema.

- En la compilación del algoritmo, usamos como **función de pérdida** *sparse categorical crossentropy*. Es la más adecuada para problemas de clasificación multiclase, y permite calcular la pérdida entre dos probabilidades (siendo 1 cuando un ejemplo pertenece a una clase en concreto y 0 cuando pertenece a cualquiera de las otras).

- La ejecución se hace en conjuntos de 32 filas. Esto se hace, porque preferimos que la red neuronal tome conjuntos pequeños puesto que los conjuntos grandes tienden a generalizar peor, dando peores resultados.

- Se realizan 10 ciclos o *epochs*. Se controla la precisión del subconjunto de validación respecto a la precisión general para evitar el sobreetrenamiento. Por tanto, en 10 se consigue que las dos precisiones mencionadas anteriormente sean parecidas y adecuadas.

- La fracción de datos del conjunto de entrenamiento a usar como conjunto de validación es del 20%. Se utiliza este tamaño, puesto que es una práctica extendida dividir el conjunto de validación siguiente un ratio 80/20.

```
input_layer_size = n
hidden_layer_size = input_layer_size*2
output_layer_size = len(yle.classes_)

X_train_NN, X_test_NN, Y_train_NN, Y_test_NN =
    train_test_split(X_NN, Y_NN, test_size=0.25, random_state=42)

```

```
model = Sequential()
model.add(Dense(hidden_layer_size, input_dim=input_layer_size,
    activation = 'relu'))
model.add(Dense(output_layer_size, activation = 'softmax'))

model.compile(optimizer = 'ADAM', loss =
    'sparse_categorical_crossentropy', metrics=['accuracy'])
```

Listing 6.8: Implementación de las capas de la red neuronal, así como las funciones de activación, perdida y optimización

Capítulo 7

Resultados

En este capítulo vamos a mostrar los resultados obtenidos después de aplicar los diferentes algoritmos desarrollados y explicados durante el proyecto. Para ello, mostraremos la precisión de cada una de la técnicas a la hora de realizar la deducción de las paradas de bajada. También haremos una comparativa entre los resultados que arrojan días de diario y días festivos, para ver las posibles diferencias que podamos encontrar fruto de la impredecibilidad de los viajes realizados en días especiales.

7.1 Resultado de las reglas de decisión y comparativa con las paradas de bajada disponibles

En esta sección se ilustran los resultados obtenidos sobre la predicción de las paradas de bajada, aplicando para ello las reglas de decisión. Nuestro algoritmo de bajada toma como conjunto de datos solamente las transacciones categorizados como subida. Para obtener la precisión de nuestro algoritmo de bajada, debemos comparar las posibles paradas de bajada obtenidas con los datos de bajada reales de los que disponemos. Estos son en su mayoría para los tipo de transporte de cercanías y metro ligero, por lo tanto, el conjunto de validación del que disponemos se vuelve bastante más pequeño. Las validaciones de metro solamente conforman el 1% de las totales, esto ocurre porque en metro solamente se valida la tarjeta en la salida de una estación en momentos puntales; mientras que en cercanías y en metro ligero se valida la bajada en la mayoría de estaciones. Por otra parte, no existen datos de los autobuses en las validaciones puesto que nunca se valida la tarjeta al bajar de este transporte.

También debemos destacar, que para todos los resultados obtenidos, se han elegido solamente transacciones cuyas tarjetas tengan un número alto de apariciones. Tras varios experimentos modificando el umbral, decidimos utilizar el de 22 apariciones ya que conseguíamos un número de filas razonablemente grande pero todavía manejable para nosotros. Esto se hace, puesto que sino el conjunto de datos que obtendríamos para cada día sería muy grande y

muy complicado de manejar a la hora de calcular todas la duraciones necesarias con el servidor descrito en la sección 6.2. Además, al descartar las transacciones con tarjetas con baja aparición, propiciamos que los datos con los que vamos a trabajar muestren los patrones existentes, sobre todo del tipo del principio de simetría. También evitamos muchas transacciones aisladas del tipo *trayecto único*, algo beneficioso para nosotros puesto que lo queremos analizar son los patrones de movilidad continua de usuarios en la red de transporte. Y en el caso de este tipo de transacciones no podemos deducir la parada de bajada, ya que no cumplen con ninguno de los principios en los que se basa nuestro algoritmo.

A la hora de clasificar las predicciones, se ha seguido el siguiente criterio:

- **Éxito total**: la parada predicha es exactamente la misma que la parada de bajada real.
- **Éxito parcial**: la parada predicha está cerca y conectada de la parada de bajada real. Conectada, quiere decir que la parada predicha es accesible desde la parada real a través de alguna de las lineas que la atraviesan. Para considerarse éxito parcial, se ha considerado que la parada predicha esté a dos paradas antes o después de la parada real. Se consideran dos paradas de diferencia un margen de error razonable porque por particularidades de las paradas, consideramos coherente suponer que un viajero podría escoger bajarse en una parada cercana pero distinta a la más óptima en términos de distancia a su siguiente subida.
- **Fracaso**: no se cumple ninguno de los criterios anteriores. La parada predicha no es la misma que la real y, además, no se cumple el umbral de que la parada predicha se encuentra a ±2 paradas de la parada real.

A continuación se muestra una tabla para cada conjunto, en las aparecen los resultados obtenidos, estas son: la figura 7.1, la figura 7.2 y la figura 7.3. En ellas nos encontramos con los siguientes datos:

- Días por los que esta formado el conjunto.
- Tamaño del conjunto de datos.
- Número de paradas de bajada predichas para el conjunto de datos.
- Número de paradas de bajada reales de las que disponemos para realizar la validación.
- Éxitos totales.
- Éxitos parciales.
- Fracasos.

Los éxitos y fracasos, también los expresaremos como un porcentaje, para ver la precisión obtenida respecto al conjunto de validaciones totales disponibles.

Para los ejemplos de los días laborables, hemos cogido la tabla 7.1; y para los ejemplos de días festivos, hemos cogido la tabla 7.3 y la tabla 7.2.

Para los días festivos utilizaremos dos conjuntos, porque el volumen de datos de los días 1 y 2, era bastante menor en comparación con los días 21 y 22. Esto, podría provocar que la comparativa no fuese demasiado precisa, por lo que hemos decidido coger un día festivo más. De esta forma, podemos ver que en número de predicciones la tabla del los días 1, 2 y 3 es más similar a la de los días 21 y 22. A pesar de esto, el número de paradas de bajadas reales disponibles para validar es claramente mayor, esto puede deberse a que en esos días justo tenemos más datos de cercanías, que es el medio de transporte del cual poseemos paradas de bajada reales.

Resultados días 21-22 de Enero	
Tamaño del conjunto de datos en filas	366.199
Parada de bajada predichas	122.658
Predicciones validadas	11.693
Éxitos totales	9.853 (84,26%)
Éxitos parciales	451 (3,86%)
Fracasos	1.389 (11,88%)

Tabla 7.1: Resultados de la comparativa de los días 21-22 de Enero con el conjunto de paradas de bajadas reales disponibles. Utilizando el algoritmo de bajadas.

Resultados días 1-2 de Enero	
Tamaño del conjunto de datos en filas	265.265
Parada de bajada predichas	76.582
Predicciones validadas	11.708
Éxitos totales	9.808 (83,77%)
Éxitos parciales	567 (4,84%)
Fracasos	1.333 (11,39%)

Tabla 7.2: Resultados de la comparativa de los días 1-2 de Enero con el conjunto de paradas de bajadas reales disponibles. Utilizando el algoritmo de bajadas.

Resultados días 1-2-3 de Enero	
Tamaño del conjunto de datos en filas	454.190
Parada de bajada predichas	133.659
Predicciones validadas	20.110
Éxitos totales	17,604 (87,54%)
Éxitos parciales	729 (3,63%)
Fracasos	1.777 (8,83%)

Tabla 7.3: Resultados de la comparativa de los días 1-2-3 de Enero con el conjunto de paradas de bajadas reales disponibles. Utilizando el algoritmo de bajadas.

A partir de los resultados obtenidos, podemos concluir que el principio de continuidad parece cumplirse. Puesto que, cuando el destino del usuario esta claro, casi siempre se baja en la parada más cercana a su parada de subida a la que pueda llegar en un tiempo admisible. En base a esto, podemos ver que por eso no existe un gran volumen de éxitos parciales, ya que no es frecuente que un usuario no se baje en la parada claramente más adecuada (será la mas cercana) para realizar su siguiente subida.

7.2 Resultados técnicas de IA y comparativa con las paradas de bajada disponibles

En esta sección, veremos los resultados obtenidos para las dos técnicas de Inteligencia Artificial que hemos utilizado: random forest y redes neuronales.

Debemos destacar que el conjunto utilizado para obtener estos resultados ha sido el correspondiente a la semana del 21 al 27 de Enero de 2019. En este caso, hemos utilizado una semana entera, puesto que, aunque el conjunto de datos incluye millones de transacciones, muchas de ellas corresponden a viajes para los cuales no se registra información de bajada. Dado que el planteamiento de nuestras predicciones requiere que contemos con datos de subida que cuenten con su correspondiente bajada y seguidos de una subida inmediatamente posterior, el número de ocurrencias se reduce considerablemente. Además, también se descartan muchos datos realizando la limpieza de los datos para poder pasárselos a los algoritmos. Por tanto, si cogiésemos solamente los datos de los días 21 y 22, tal y como hicimos para el algoritmo de bajada, nos quedaríamos con un volumen de datos muy pequeño que podría dar lugar a resultados poco fiables al compararlo con las reglas de decisión.

Del conjunto inicial con casi 2.5 millones de observaciones, nos quedamos con 335.000 lineas tras aplicar la limpieza y colocación de datos. Este conjunto final, es el que aplicaremos

a nuestras dos técnicas de IA.

7.2.1 Resultados Random Forest

Una vez ejecutado el algoritmo de random forest con los parámetros especificados en la Sección 6.3.4, obtenemos la precisión. Para ello, tras tener el algoritmo entrenado, hacemos una predicción para el conjunto *X_test* que habíamos reservado previamente. Tras esto, realizamos la comparativa entre esta predicción con el conjunto *Y_test* obteniendo una **precisión del 75,78%**.

7.2.2 Resultados Red Neuronal

Antes de nada, debemos definir un concepto muy importante en redes neuronales que va a aparecer a lo largo de esta sección. Este es el sobreentrenamiento, ocurre cuando una red neuronal aprende a clasificar de manera muy especifica los datos del conjunto de entrenamiento y pierde la aplicabilidad a otros datos. Esto, resulta en predicciones erróneas para conjuntos de datos futuros.

Una vez ejecutada la red neuronal con los parámetros especificados en la Sección 6.3.4, obtenemos la precisión. Para ello, realizamos la predicción para el conjunto *X_test* con la red entrenada. Después, se comparan los resultados obtenidos, con el conjunto *Y_test* que contiene las paradas de bajada que deberíamos haber obtenido. Con esto, obtenemos una **precisión del 59,39%**.

En la figura 7.1 podemos ver la evolución de la precisión conforme van pasando los ciclos, o *epochs*. La línea azul etiquetada como *train*, indica la precisión del conjunto de entrenamiento de cada ciclo. Por otra parte, la linea naranja etiquetada como *val*, indica la precisión obtenida por el conjunto de validación en cada ciclo. Por tanto, este último valor es al que debemos atender.

Como podemos ver, el entrenamiento finaliza en el ciclo 10, donde las representaciones de *val* y *train* convergen, lo cual nos indica que hemos alcanzado el valor de precisión óptimo sin llegar a sobreentrenar la red. Si continuásemos realizando ciclos, el valor val se estancaría y el valor *train* comenzaría a descender, lo cual nos indicaría sobreentrenamiento.

Por otra parte, en la figura 7.2 encontramos la evolución del error obtenido a través de la función *loss* utilizada, *sparse categorical crossentropy*. Esta función es la más adecuada para problemas de clasificación multiclase cuyas variables categóricas se han codificado.

En la figura, podemos ver dos líneas: la línea azul indica el error obtenido por el conjunto de entrenamiento en cada ciclo; y la linea naranja el error obtenido por la red a lo largo de los diferentes ciclos, es decir el conjunto de validación. Como podemos ver al principio, el error desciende de una manera muy pronunciada, y conforme la red van aprendiendo y mejorando su manera de predecir el error se va minimizando hasta que se mantiene bastante estable.

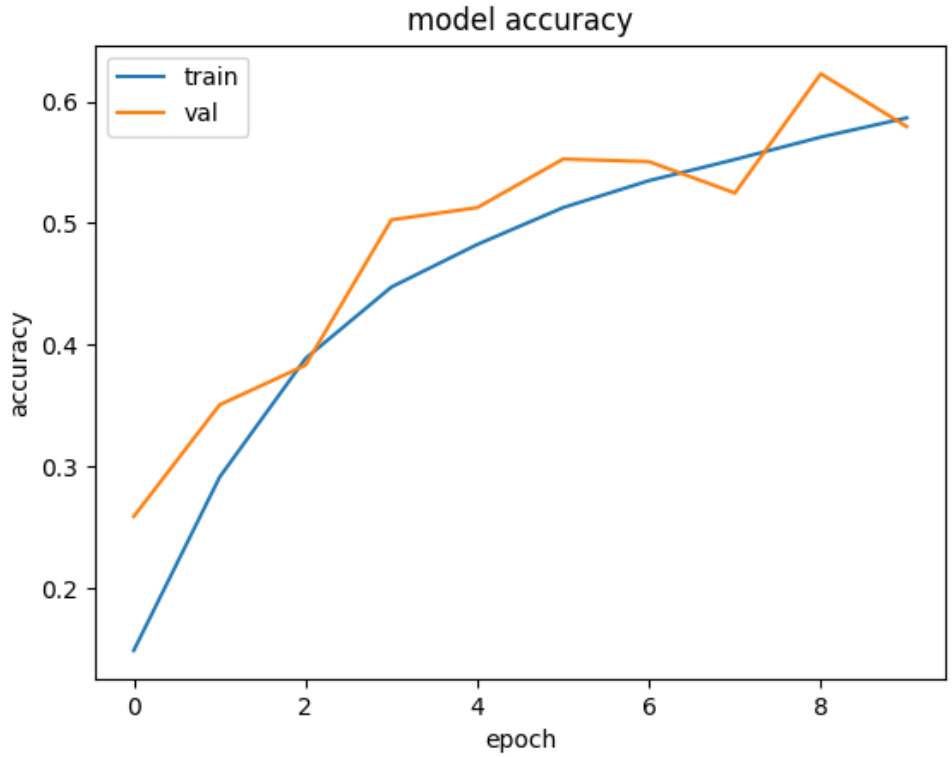

Figura 7.1: Relación entre la precisión de la red neuronal y el conjunto de validación de cada ciclo.

Esto último, también nos indica cuando debemos parar de realizar ciclos, puesto que si nos encontramos con varios ciclos en los cuales la mejora no existe o no es notable, no tiene sentido seguir realizándolos. Por tanto, también nos indica que no debemos realizar más ciclos para evitar el sobreentrenamiento de la red neuronal.

7.3 Comparativa 1,2 enero y 20,21 enero

Por último, hemos elegido dos conjuntos pertenecientes a días con características muy distintas para realizar una comparativa de sus resultados.

Por una parte, tenemos el conjunto de los días 1 y 2 de Enero de 2019. Estos, son días festivos por lo que es bastante probable que se produzcan viajes anómalos y fuera de la rutina habitual de cada usuario.

Por otra parte, tenemos el conjunto de los días 21 y 22 de Enero de 2019. Estos, corresponden al lunes y martes de una semana laboral típica, por lo que esperamos encontrar patrones más regulares en los viajes realizados por los viajeros.

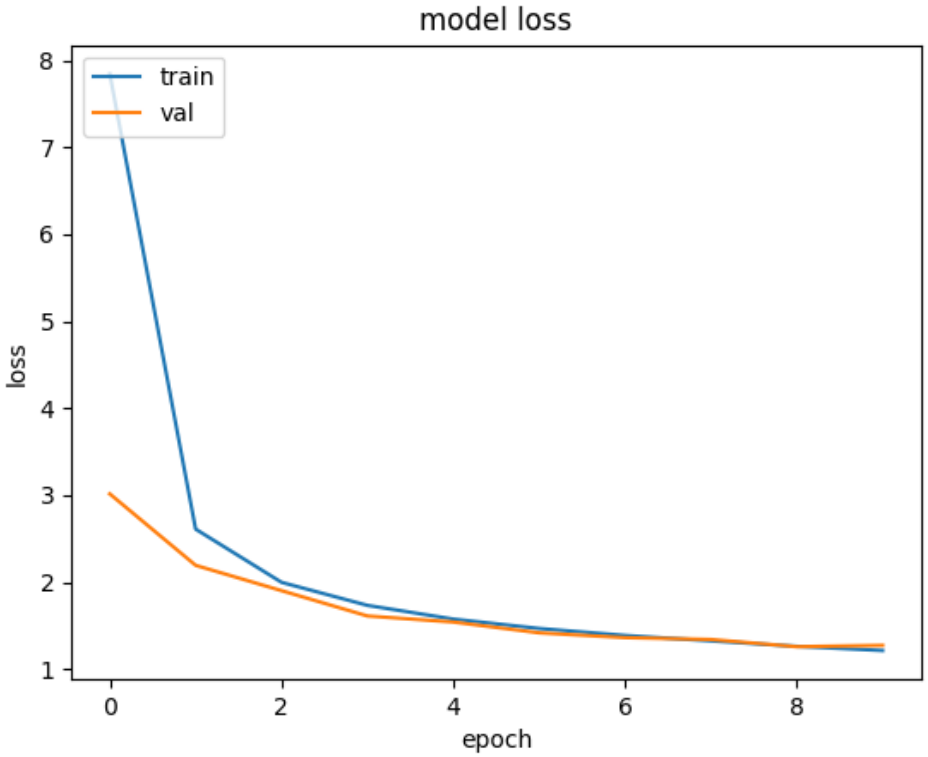

Figura 7.2: Relación entre el error obtenido por la red neuronal y el conjunto de validación de cada ciclo

7.3.1 Algoritmo de bajadas

Primero, analizamos la diferencia en precisión obtenida mediante el algoritmo de bajada:

- En la tabla 7.2, tenemos los resultados para los días 1 y 2 de Enero. Aquí, se validan 11.708 trayectos con un porcentaje de **éxito del 83,77%**.
- En la tabla 7.1, tenemos los resultados para los días 21 y 22 de Enero. Aquí, se validan 11.693 trayecto con un porcentaje de **éxito del 84,26%**.

En ambos casos, el número de validaciones realizadas es similar, a pesar de tener muchas más paradas de bajada predichas para los días 21 y 22. Esto no podemos atribuirlo a los días en sí, puesto que los datos para validar solamente incluyen los realizados en cercanías y metro ligero. Por lo que, lo único que podemos intuir es que se han realizado más viajes en estos medios de transporte los días 1 y 2, aunque tampoco es algo que se pueda afirmar rotundamente.

En cuanto al porcentaje de éxitos obtenidos, es casi el mismo en los dos casos. En base

a esto, podemos decir que ha afectado muy poco que los viajes sigan patrones más o menos habituales al algoritmo de bajadas; y que en cada caso predice de manera satisfactoria.

7.3.2 Técnicas de IA

Por último, analizamos la diferencia en precisión obtenidas mediante las distintas ténicas de Inteligencia Artificial utilizadas.

Random Forest

En el caso de random forest obtenemos las siguientes precisiones:

- Días 1 y 2 de Enero de 2019: contamos con un conjunto final de datos para pasar al algoritmo de 109.511 filas; y obtenemos una **precisión del 77,71%**.
- Días 21 y 22 de Enero de 2019: contamos con un conjunto final para pasar al algoritmo de 115.981 filas; y obtenemos una **precisión del 75,08%**.

Red Neuronal

En el caso de la red neuronal obtenemos las siguientes precisiones:

- Días 1 y 2 de Enero de 2019: contamos con un conjunto final de datos para pasar al algoritmo de 109.511 filas; y obtenemos una **precisión de 55,25%**.
- Días 21 y 22 de Enero de 2019: contamos con un conjunto final para pasar al algoritmo de 115.981 filas; y obtenemos una **precisión del 46,58%**.

En ambos algoritmos, la precisión fue mejor para los días 1 y 2 de Enero. En random forest la diferencia no fue muy notable entre ambos conjuntos, pero en la red neuronal la mejoría en la precisión fue casi de un 10%. Con esto, podemos concluir que en períodos vacacionales las personas tienden a usar más el transporte público para llegar a su destino y, es posible que el movimiento de la ciudadanía sea más uniforme porque hay lugares que suelen representar un destino común en períodos festivos, como centros urbanos y grandes superficies comerciales. Sin embargo, para poder realizar afirmaciones sobre el movimiento de la gente deberíamos realizar un estudio en detalle de los trayectos representados en cada conjunto de datos, y esto es algo que no entra dentro del alcance de este proyecto.

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo, se detallan las conclusiones finales sobre el proyecto, así como el grado de completitud del mismo en base a los objetivos y funcionalidades planteadas al principio del mismo. También se indican los conocimientos adquiridos y la destreza desarrollada en las distintas tecnologías utilizas a lo largo del trabajo.

Por otra parte, se indican posibles ampliaciones y mejorar a realizar del proyecto y se marcan como trabajo futuro.

8.1 Conclusiones

Tras la finalización del proyecto, hemos podido llegar a las siguientes conclusiones:

- Se han llevado a cabo de manera satisfactoria todos los objetivos del proyecto.
- Fijándonos en la planificación inicial, la iteración 4 correspondiente al visor GIS no se pudo realizar. Esto ocurrió debido a, principalmente, problemas de tiempo durante la obtención de datos con el servidor. A pesar de esto, todas las tareas prioritarias contenidas en las iteraciones 1,2 y 3 se llevaron a cabo de manera satisfactoria.
- El trabajo de Extract, transform and load se ha completado dejando un *backend* prácticamente listo para integrar con el futuro prototipo del sistema de análisis de transporte.
- Se ha conseguido deducir paradas de bajada con una precisión adecuada y muy prometedora. Esto se ha conseguido utilizando tanto el algoritmo de bajadas basado en reglas de decisión, como varias técnicas de Inteligencia Artificial: random forest y una red neuronal.
- Se han adquirido conocimientos en las tecnologías utilizadas para el desarrollo, así como en todos los algoritmos utilizados:

- Se ha desarrollado una gran destreza en SQL, la tecnología utilizada para realizar todas las reglas de decisión del algoritmo de bajadas.
- Se ha desarrollado una gran destreza en la creación de scripts para la obtención y gestión de grandes volúmenes de datos con *Python* y las librerías asociadas (csv y pandas).
- Se ha desarrollado una gran destreza con las librerías utilizadas para la implementación de los algoritmos de Inteligencia Artificial. Estas son: sklearn, keras y tensorflow.
- Se han afianzado y ampliado los conceptos sobre Inteligencia Artificial así como de las dos técnicas con las que se ha trabajado.

8.2 Trabajo futuro

Una vez finalizado el proyecto, a pesar de todo lo conseguido de manera satisfactoria, se plantean posibles mejoras. Algunas de ellas, son las que se definieron en el planteamiento del proyecto (capitulo 3) como es el caso de un visor cartográfico. Y otras, son simplemente mejoras que podrían añadirse a la lógica de los algoritmos a la hora de deducir las paradas de bajada candidatas.

8.2.1 Utilización de Hubs

Como idea para mejorar el algoritmo de bajadas basado en reglas de decisión se plantean los hubs. Los hubs serían los lugares que, por probabilidad, indicaríamos como parada de bajada en caso de no ser capaces de predecirla de otra manera. Entendemos por hub a un gran centro comercial, paradas muy concurridas, un hospital.... Se utilizarán, sobre todo, en casos en los que el tipo de trayecto sea *Fin de viaje* y no se le pueda aplicar el principio de simetría. Porque, tal y como explicamos antes, la única forma de predecir trayectos de fin de viaje que utilizamos actualmente es mediante este principio. Por tanto, los hubs solucionarían la predicción de muchas de las paradas etiquetadas como *Fin de viaje* que existen en nuestro conjuntos de datos.

A la hora de seleccionar uno, deberemos atender a qué lugares de interés encontramos que cumplan con determinadas condiciones, estas son:

- El tipo de trayecto de la transacción debe ser: *Fin viaje*.
- El hub debe admitir el mismo método de transporte que el utilizado en para la parada de subida de la transacción anterior.

- Deberíamos fijarnos en que sea coherente que un usuario se desplace a ese hub a una hora determinada. Por ejemplo, solo acudirá a centros comerciales si se encuentran en horario de apertura.

8.2.2 Visor GIS

El visor GIS, se planteó inicialmente como parte del proyecto, pero debido a complicaciones durante el desarrollo y falta de tiempo, se decidió prescindir de él. Sin embargo, consideramos que mejoraría ampliamente el trabajo actual, puesto que significaría una forma de representar los datos y ver los resultados obtenidos de una forma más visual; lo cual sería realmente útil especialmente para los usuarios finales sin conocimientos previos sobre los datos.

El visor GIS consiste en una aplicación web con la que podemos interactuar y en la que podemos visualizar los datos que precisemos en un mapa.

En nuestro caso, utilizaríamos el visor cartográfico para mostrar en el mapa de Madrid las paradas de bajada deducidas. Y, a partir de ellas, mostrar las zonas del mapa con un mayor volumen de trayectos en festivos o laborables mediante un mapa de calor; y mostrar información relevante de cada parada. La información imprescindible debería incluir:

- Número de veces que una parada ha sido: *Inicio viaje*, *Fin viaje* o *Etapa*.
- Cuales son las lineas de esa parada que los viajeros utilizan con mas frecuencia.
- Volumen de trayectos totales que pasan por esa estación.

También deberíamos tener la opción de aplicar filtros en el momento de mostrar la información: fecha, linea, tipo de transporte, ….

Como idea para realizar la implementación del visor GIS, se plantea lo siguiente como supuesto básico:

- Base de datos postgresql con una extesión postgis, donde almacenaremos toda la información de las paradas de bajada, así como los datos de las topologías de la red de transporte.
- Framework Spring para obtener una api REST.
- Visor GIS que conectaremos mediante peticiones REST. Estaría creado con el framework vue.js y la librería Leaflet para la creación de los mapas.

Finalmente, se adjuntan en el Anexo B posibles mock-up del visor GIS.

Apéndices

Apéndice A

Códigos de validación

En este capítulo se muestran los códigos de validación con su literal, medios de transporte en los que aparece, suposición de su significado y clasificación.

Código: 0 **Número apariciones**: 225581

Literal:
Código de resultado inválido

Medios de transporte:
Todos

Suposición:
Código de error. La transacción es invalida.
##########
Código: 1 **Número apariciones**: 67110199

Literal:
Validación completa - con temporal con descuento de viaje

Medios de transporte:
Todos

Suposición:
Subida a un medio de transporte cualquiera.
##########
Código: 2 **Número apariciones**: 55303758

Literal:
Validación completa - aviso sobre las unidades que quedan (según el título)

Medios de transporte:
Todos

Suposición:
Variante del código 1 que avisa de que el abono está a punto de caducar.
##########
Código: 3 **Número apariciones**: 1948

Literal:
Validación completa - sin descuento de viaje (en trasbordo)

Medios de transporte:
Autobuses EMT

Suposición:
Indica una subida en el caso en que que ha existido una subida anterior en un margen de tiempo suficientemente corto para considerarse un trasbordo.
##########
Código: 4 **Número apariciones**: 4444294

Literal:
Validación completa - con descuento de viaje (con multiviaje)

Medios de transporte:
Todos

Suposición:
Subida en un medio de transporte con descuento de viaje. El viajero viaja desde o hacia una zona tarifaria que no está incluida en su abono.
##
Código: 5 **Número apariciones**: 9838

Literal:
Validación completa - con reintegro de viaje (con un multiviaje en salida)

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en dos estaciones de la línea 10. Puerta del Sur o Joaquín de Vilumbrares. Nótese que son las dos primeras estaciones de la línea 10.

Mi suposición a juzgar por el literal es que se trata de una bajada bajo ciertas circunstancias especiales que resultan en un reintegro del viaje realizado.
##
Código: 6 **Número apariciones**: 0

Literal:
Validación completa - servicio gratuito

Medios de transporte:
Desconocido

Suposición:
Subida a un medio de transporte cualquiera sin cargo asociado.
##
Código: 7 **Número apariciones**: 0

Literal:
Validación completa - paso por barrera intermedia con descuento de viaje

Medios de transporte:
Desconocido

Suposición:
Su literal es muy similar al del código 8, con la diferencia de que especifica que sí hay descuento de viaje. Puede ser una variante del código 8.
##
Código: 8 **Número apariciones**: 224

Literal:
Validación completa - paso por barrera intermedia sin descuento de viaje

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en dos estaciones: Puerta de Arganda y la Fortuna. Puerta de Arganda es fin de línea 9 operada por el Metro de Madrid y comienzo de la línea especial A9 (extensión de la línea 9 operada por Transportes Ferroviarios de Madrid).

La Fortuna es extremo de la línea 11. Es importante destacar que un usuario que quiera proseguir su viaje por la línea A9 debe cambiar de tren obligatoriamente en Puerta de Arganda. De la misma forma, la Fortuna es la única estación de Madrid que conecta con el metro de Leganés, habiendo un cambio de tren obligatorio en la Fortuna para usuarios que vengan de Leganés o vayan a Leganés.

Mi suposición en base al literal sería que este código se da en cambio de tren obligatorio en Puerta de Arganda y la Fortuna. Más concretamente, creo que lo que está especificando este código es "cambio de tren a una línea con un operador distinto".
##########
Código: 9 **Número apariciones**: 9599385

Literal:
Validación completa - con temporal sin descuento de viaje (interesa en validación en salida)

Medios de transporte:
Cercanías, metro y metro ligero

Suposición:
Bajada en cualquier medio de transporte donde exista posibilidad de detectar la bajada (es decir, excluyendo autobuses).
##########
Código: 10 **Número apariciones**: 542377

Literal:
Validación en salida sin detección de viaje

Medios de transporte:
Cercanías, metro y metro ligero

Suposición:
Variación del código 9 para casos donde no existe validación previa de bajada
##########
Código: 11 **Número apariciones**: 49657

Literal:
Validación completa - prolongación de recorrido Título Temporal

Medios de transporte:
Autobuses interurbanos

Suposición:
Indica una subida específicamente en un autobus interurbano. Notar que también existen subidas en autobuses interurbanos con código de validación 1. No tengo claro en qué se diferencian estas subidas, si bien la clave debería estar en la parte del literal que especifica "prolongación de recorrido"
##########
Código: 12 **Número apariciones**: 5555

Literal:
Validación completa - prolongación de recorrido Título Multiviaje

Medios de transporte:
Autobuses interurbanos

Suposición:
Variación del código 11 para título multiviaje.

##########
Código: 13 **Número apariciones**: 124471

Literal:
Validación completa- Tarjeta titulo validado (Solución para corregir el problema con el Commit-Transaction)

Medios de transporte:
Todos

Suposición:
Código creado para denotar un problema específico, desconozco a que se puede referir exactamente.
##########
Código: 14 **Número apariciones**: 24388

Literal:
Validación completa - (Baterías intermedia) Título temporal salida descuento viaje (origen distinto desconocido).

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO, en diversas estaciones que no parecen tener nada en común. A juzgar por el literal mi suposición es que se trata de una bajada ("título temporal salida"), en una estación de metro con origen de distinta ámbito desconocido ("origen distinto desconocido"). "Baterías intermedias" podría sugerir además que se trata de un cambio de tren.
##########
Código: 15 **Número apariciones**: 67538

Literal:
Validación completa - (Baterías intermedia) Título multiviaje salida descuento viaje (origen distinto desconocido).

Medios de transporte:
Metro

Suposición:
Variante del código 14 para título multiviaje.
##########
Código: 16 **Número apariciones**: 545132

Literal:
Validación Completa - Con t. temporal en batería intermedia de salida, sin descuento de viaje (Entrada en la misma zona)

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en dos estaciones: Puerta del Sur y Tres Olivos. Puerta del Sur conecta la línea 10 y la 12 (MetroSur). Tres Olivos conecta la línea 10 del metro de Madrid con una extensión de la misma denominada MetroNorte. En otras palabras, ambas estaciones conectan la línea 10 del Metro de Madrid con MetroSur y MetroNorte. Ambas estaciones obligan a cambiar de tren para continuar el viaje.

Mi suposición es que este código denota el cambio de tren obligatorio en estas estaciones, con origen en la misma zona tarifaria (“Entrada en la misma zona”).
##########
Código: 17 **Número apariciones**: 644816

Literal:
Validación Completa - Validación t. multiviaje salida sin descuento de viaje y con origen - destino del mismo ámbito.

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en estaciones que se encuentren en MetroSur o en MetroNorte. Mi suposición es que el código señala una bajada en una de las estaciones que conforman estas líneas. Además, **destino del mismo ámbito** en este contexto nos indica que ambos origen y destino se encuentran en la misma línea MetroSur/MetroNorte.
##########
Código: 18 **Número apariciones**: 94519

Literal:
Validación Completa - Con t. multiviaje en batería intermedia de salida, sin descuento de viaje (Entrada en la misma zona)

Medios de transporte:
Metro

Suposición:
Variación del código 16 para título multiviaje.
##########
Código: 19 **Número apariciones**: 127241

Literal:
Validación Completa - Para multiviaje en batería intermedia de salida sin descuento de viaje (Entrada en la misma zona)

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en tres estaciones: Puerta del Sur, Tres Olivos y Las Tablas. El literal es exactamente el mismo que con el código de validación 18, sin embargo este código puede suceder en Las Tablas, mientras que el 18 no. Las Tablas es una estación de la línea de MetroNorte de Madrid con la particularidad de que conecta esta línea de Metro con la línea ML1 de Metro Ligero (tranvía).

En base a esto, mi suposición es que es casi un duplicado del código 18 que, a parte de incluir los casos en los que se puede dar este código, incluye el caso de cambio de metro ligero a metro (no estoy seguro si también el caso contrario). Lo que no entiendo es qué diferencia un código 19 en Puerta del Sur y Tres Olivos de un código 18 (porque se dan ambos códigos en nuestros datos).
##########
Código: 20 **Número apariciones**: 981

Literal:
Validación completa - t. multiviaje en barrera intermedia de salida sin descuento de viaje. Titulo utilizado para validar en entrada anteriormente, no más viajes, origen mismo ámbito

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en la estación Puerta del Sur. Mi suposición es que señala una bajada en Puerta del Sur. Aunque el uso de "barrera intermedia" parece señalar validación en cambio de tren, el que el literal especifique "no más viajes" me hace sospechar que de alguna manera esta transacción de esta señalando como fin de un multiviaje. "Origen del mismo ámbito" señala que el origen está en la misma línea MetroSur.
##########
Código: 21 **Número apariciones**: 0

Literal:
Validación completa - Validación t. multiviaje en barrera intermedia de salida sin descuento de viaje. título utilizado para validar en entrada anteriormente, no más viajes, origen otro ámbito

Medios de transporte:
Metro

Suposición:
Variante del código 20 para origen de otro ámbito.
##########
Código: 22 **Número apariciones**: 12452

Literal:
Validación completa - Validación t. multiviaje en barrera intermedia de salida descuento un viaje con derecho a devolución y origen en el mismo ámbito.

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en Puerta del Sur. Mi suposición es que se trata de una versión alternativa de los códigos 20 y 21. En este caso señalaría una bajada en Puerta del Sur bajo unas condiciones especiales (si bien no tengo claro cuales serían estas condiciones analizando las apariciones de este código en nuestros datos) que dan al viajero derecho a devolución del importe del viaje y origen del mismo ámbito (en este contexto mismo ámbito se refiere a la línea MetroSur).
##########
Código: 23 **Número apariciones**: 675

Literal:
Validación completa - Validación t. multiviaje en barrera intermedia de salida descuento un viaje con derecho a devolución y origen en el mismo ámbito.

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en Puerta del Sur. Mi suposición es que se trata de una versión alternativa de los códigos 20 y 21. En este caso señalaría una bajada en Puerta del Sur bajo unas condiciones especiales (si bien no tengo claro cuales serían estas condiciones analizando las apariciones de este código en nuestros datos) que dan al viajero derecho a devolución del importe del viaje y origen del mismo ámbito (en este contexto **mismo ámbito** se refiere a la línea MetroSur).
##########

Código: 24 **Número apariciones**: 157

Literal:
Validación completa - Validación t. multiviaje en barrera intermedia de salida descuento dos viajes con derecho a devolución y origen desconocido (sin validación en entrada).

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en Puerta del Sur. Parece una versión alternativa de los códigos 22 y 23 en la que el viajero sin bien presenta origen desconocido (sin validación en entrada), SÍ tiene derecho a devolución. Este derecho parece estar relacionado con el hecho de que sean "dos viajes", que no sé que significa.
##########
Código: 25 **Número apariciones**: 38307

Literal:
Validación completa - Validación t. multiviaje en entrada sin descuento de viaje. Dicho título ha sido antes cancelado en ML2-3 no excede periodo máximo permitido de un viaje (180 min)

Medios de transporte:
Metro y metro ligero

Suposición:
Solo se da en METRO y en METRO LIGERO en Colonia Jardín. Mi suposición es que se trata de un trasbordo de metro a metro ligero y viceversa.
##########
Nota: no existen códigos del 26 al 31, el siguiente código después del 25 es el 32.

Código: 32 **Número apariciones**: 79302

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen TFM - Puerta de Arganda.

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en estaciones de la línea especial A9 que empieza en Puerta de Arganda y que extiende la línea 9 bajo un operador distinto al resto del metro de Madrid. Mi suposición es que se trata de una bajada en una estación de está línea con origen en la propia Puerta de Arganda (TFM indica que el operador es Transportes Ferroviarios de Madrid, así que se refiere a origen específicamente en la estación de Puerta Arganda de la línea especial A9).
##########
Código: 33 **Número apariciones**: 10088

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen TFM - Puerta de Arganda.

Medios de transporte:
Metro

Suposición:
Variante del código 32 para título multiviaje.
##
Código: 34 **Número apariciones**: 171436

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen METRO (METROMADRID, METRONORTE, etc.) salvo Puerta de Arganda.

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en estaciones de la línea especial A9 que empieza en Puerta de Arganda y que extiende la línea 9 bajo un operador distinto al resto del metro de Madrid.

Mi suposición a juzgar por el literal es que se trata de una bajada en una estación dentro de la línea A9 con origen FUERA de la misma.
##
Código: 35 **Número apariciones**: 41826

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen METRO (METROMADRID, METRONORTE, etc.) salvo Puerta de Arganda.

Medios de transporte:
Metro

Suposición:
Variante del código 34 para título multiviaje.
##
Código: 36 **Número apariciones**: 14831

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen solo Puerta de Arganda.

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en estaciones de la línea especial A9 que empieza en Puerta de Arganda y que extiende la línea 9 bajo un operador distinto al resto del metro de Madrid. A juzgar por el literal este código indica una bajada en una estación de esta línea con origen específicamente en la estación de Puerta de Arganda operada por el Metro de Madrid (es decir, que la entrada no se produce en la propia línea A9, tratan Puerta de Arganda de la línea 9 y Puerta de Arganda de la línea A9 como estaciones distintas).
##
Código: 37 **Número apariciones**: 750

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen METRO solo Puerta de Arganda.

Medios de transporte:
Metro

Suposición:
Variante del código 36 para título multiviaje.
##########
Código: 38 **Número apariciones**: 7388

Literal:
Validación completa - Validación t. temporal salida descuento viaje y con origen desconocido.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea de metro especial A9 con origen desconocido.
##########
Código: 39 **Número apariciones**: 12512

Literal:
Validación completa - Validación t. multiviaje salida descuento viaje y con origen desconocido.

Medios de transporte:
Metro

Suposición:
Variante del código 38 para título multiviaje.
##########
Código: 40 **Número apariciones**: 239994

Literal:
Validación Completa - Con t. temporal en batería intermedia de salida sin descuento de viaje (Entrada en la misma zona)

Medios de transporte:
Metro

Suposición:
Duplicado del código 16.
##########
Código: 41 **Número apariciones**: 10643

Literal:
Validación Completa - Con t. temporal en batería intermedia de salida descuento viaje y sin validación previa.

Medios de transporte:
Metro

Suposición:
Variante del código 16 (y por extensión del 40) en la que no se ha detectado validación previa en entrada (se desconoce el origen del viajero).
##########
Código: 42 **Número apariciones**: 38030

Literal:
Validación Completa - Para multiviaje en batería intermedia de salida descuento viaje y sin validación previa.

Medios de transporte:
Metro

Suposición:
Variante del código 16 (y por extensión del 40) para título multiviaje sin validación previa.
##########
Código: 43 **Número apariciones**: 24372

Literal:
Validación completa - Validación t. temporal en entrada sin descuento de viaje. Cancelado en entrada en ML1 (batería de entradas o a bordo) no excede periodo máximo permitido (180 min)

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en una estación: Las Tablas. Las Tablas es una estación de MetroNorte que tiene la particularidad de ser también un extremo de la línea de metro ligero ML1. Mi suposición es que denota un trasbordo de metro ligero a metro.
##########
Código: 44 **Número apariciones**: 3254560

Literal:
Validación Completa - Con t. temporal salida sin descuento de viaje y con origen - destino del mismo ámbito.

Medios de transporte:
Metro

Suposición:
Variante del código 17 para título temporal.
##########
Código: 45 **Número apariciones**: 871184

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen - destino no del mismo ámbito.

Medios de transporte:
Metro

Suposición:
Variante del código 17 para destino de otro ámbito.
##########
Código: 46 **Número apariciones**: 695

Literal:
Validación completa - Validación t. temporal en barrera intermedia en entrada descuento viaje. Dicho título no se ha utilizado para validar en entrada anteriormente.

Medios de transporte:
Metro

Suposición:
Solo sucede en METRO en las siguientes estaciones: La Fortuna, Puerta de Arganda, Puerta del Sur y Joaquín de Vilumbrares. Las tres primeras son estaciones donde se da un cambio de tren obligatorio, pero no es así en Joaquín de Vilumbrares, que es simplemente una estación de MetroSur. El uso de "barrera intermedia" me hace sospechar que este código indica un trasbordo, pero no entiendo entonces que se de en Joaquín de Vilumbrares.

"Dicho título no se ha utilizado para validad en entrada anteriormente" se refiere a que no hay una subida previa detectada, por lo tanto este código NO puede estar indicando una subida. Debe referirse a una bajada o un trasbordo.
##
Código: 47 **Número apariciones**: 4536

Literal:
Validación completa - Validación t. multiviaje en barrera intermedia en entrada descuento viaje. Dicho título no se ha utilizado para validar en entrada anteriormente.

Medios de transporte:
Metro

Suposición:
Variante del código 46 para título multiviaje.
##
Código: 48 **Número apariciones**: 4788

Literal:
Validación completa - Validación t. multiviaje en entrada sin descuento de viaje. Cancelado antes en entrada en ML1 (batería de entradas o a bordo)no excede periodo máximo permitido 180 min

Medios de transporte:
Metro

Suposición:
Variante del código 43 para título multiviaje.
##
Código: 49 **Número apariciones**: 3189

Literal:
Validación completa - Validación t. temporal en entrada sin descuento de viaje. Cancelado antes salida en Pinar de Chamartín, en bat. inter. METROMADRID Y ML1 en ese sentido)no excede per.máx.

Medios de transporte:
Metro

Suposición:
El literal es muy similar al del código de validación 43, con la diferencia de que especifica que la "cancelación" se realizó en Pinar de Chamartín. Por lo tanto mi suposición es que se trata de un trasbordo de metro ligero a metro.
##
Código: 50 **Número apariciones**: 401

Literal:
Validación completa - Validación t. multiviaje en entrada sin descuento de viaje. Cancelado antes salida en Pinar de Chamartín, en bat. inter. METROMADRID Y ML1 en ese sentido)no excede per.máx.

Medios de transporte:
Metro

Suposición:
Variante del código 49 para título multiviaje.
##########
Código: 51 **Número apariciones**: 104756

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen el mismo ámbito. Cancelado en entrada no excede periodo máximo permitido (180 min)

Medios de transporte:
Metro

Suposición:
El código parece equivalente al 43, pero para el otro extremo de la línea (Pinar de Chamartín). Por lo tanto mantengo que se trata de un trasbordo de metro ligero a metro.
##########
Código: 52 **Número apariciones**: 37993

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen el mismo ámbito. Cancelado en entrada no excede periodo máximo permitido (180 min)

Medios de transporte:
Metro

Suposición:
Variante del código 51 para título multiviaje.
##########
Código: 53 **Número apariciones**: 1116

Literal:
Validación completa - Validación t. temporal salida descuento viaje y con origen desconocido.

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en la estación Pinar de Chamartín. Parece indicar una salida en la que el origen es desconocido. Posiblemente variante del código 51 para origen desconocido (lo pongo por separado porque no estoy seguro).
##########
Código: 54 **Número apariciones**: 404

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen el distinto ámbito. Cancelado en entrada no excede periodo máximo permitido (180 min)

Medios de transporte:
Metro

Suposición:
Variante del código 51 para título multiviaje y destino de distinto ámbito.

##########
Código: 55 **Número apariciones**: 599

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen distinto ámbito. Cancelado en entrada no excede periodo máximo permitido (180 min)

Medios de transporte:
Metro

Suposición:
Variante del código 51 para destino de distinto ámbito.
##########
Código: 56 **Número apariciones**: 548

Literal:
Validación completa - Validación t. multiviaje salida descuento viaje y con origen desconocido.

Medios de transporte:
Metro

Suposición:
Variante del código 53 para título multiviaje.
##########
Código: 57 **Número apariciones**: 109278

Literal:
Validación completa - Validación t. temporal en entrada sin descuento de viaje y con origen ML1. Cancelado en alguna entrada de ML1 o en las canceladoras embarcadas no excede periodo máximo permitido

Medios de transporte:
Metro

Suposición:
Solo se da en METRO en la estación Pinar de Chamartín. Similar a los códigos anteriores, parece indicar un trasbordo de metro ligero a metro.
##########
Código: 58 **Número apariciones**: 5672

Literal:
Validación completa - Validación t. temporal en entrada descuento viaje y con origen desconocido.

Medios de transporte:
Metro

Suposición:
Variante del código 57 para origen desconocido.
##########
Código: 59 **Número apariciones**: 38696

Literal:
Validación completa - Validación t. multiviaje en entrada sin descuento de viaje y con origen ML1. Cancelado en entrada de ML1 o en las canceladoras embarcadas, no excede periodo máximo permitido

Medios de transporte:
Metro

Suposición:
Variante del código 57 para título multiviaje.
##########
Código: 60 **Número apariciones**: 2270

Literal:
Validación completa - Validación t. multiviaje en entrada descuento viaje y con origen desconocido.

Medios de transporte:
Metro

Suposición:
Variante del código 57 para título multiviaje y origen desconocido.
##########
Código: 61 **Número apariciones**: 91039

Literal:
Validación completa - Duplicado del código "39" para ML1. Validación t. temporal salida sin descuento de viaje y con origen - destino del mismo operador, es decir en este caso, ML1.

Medios de transporte:
Metro ligero

Suposición:
No tengo claro de qué código 39 está hablando. El único código 39 del que tenemos constancia se refiere a una bajada en la línea especial de METRO A9 con origen desconocido y para título multiviaje. Para empezar, este código claramente NO se refiere a título multiviaje ("t. temporal"). Seguramente sea un duplicado de un código 39 de una definición anterior de los códigos de validación y que ya no existe.

En todo caso, este código claramente hace referencia a una bajada en la línea de metro ligero ML1 con origen en la misma línea.
##########
Código: 62 **Número apariciones**: 17524

Literal:
Validación completa - Duplicado del código "3B" para ML1. Validación t. multiviaje salida sin descuento de viaje y con origen - destino del mismo operador, es decir en este caso, ML1.

Medios de transporte:
Metro ligero

Suposición:
Variante del código 61 para título multiviaje.
##########
Código: 63 **Número apariciones**: 0

Literal:
Validación completa - Duplicado del código "3C" (ML1). Validación t. multiviaje en entrada descuento viaje y con origen desconocido.

Medios de transporte:
Metro ligero

Suposición:
Parece referirse a una subida (“en entrada”) en una estación de metro ligero de la línea ML1 como parte de un multiviaje en el que el origen del viaje anterior es “desconocido”. ¿Cómo puede asegurar que es un multiviaje si el origen anterior es desconocido? Ni idea, porque este código no se da JAMÁS en nuestros datos.

Podría ser que con origen desconocido se refiera realmente a origen en otro medio de transporte.
##########
Código: 64 **Número apariciones**: 231055

Literal:
Validación completa - Duplicado del código "3A" para ML1. Validación t. temporal en entrada descuento viaje y origen desconocido.

Medios de transporte:
Metro ligero

Suposición:
Parece una variante del código 63 para título temporal.
##########
Código: 65 **Número apariciones**: 25105

Literal:
Validación completa - (ML1) Validación t. temporal en entrada descuento viaje, avisando sobre los días que quedan hasta la caducidad del título

Medios de transporte:
Metro ligero

Suposición:
Parece una variante del código 63 con el añadido de avisar que el abono utilizado está cerca de caducar.
##########
Código: 66 **Número apariciones**: 70448

Literal:
Validación completa - Validación t. multiviaje en entrada descuento viaje.

Medios de transporte:
Metro ligero

Suposición:
Se da en cualquier estación de METRO LIGERO solo en la línea ML1. De acuerdo al literal mi suposición es que se trata de una subida en una estación de dicha línea.
##########
Código: 67 **Número apariciones**: 22376

Literal:
Validación completa - Validación t. temporal en entrada sin descuento de viaje. Cancelado estación Las Tablas METRO (salida) no excede periodo máximo permitido (40 min)

Medios de transporte:
Metro ligero

Suposición:
Se da en METRO LIGERO solo en la estación Las Tablas de la línea ML1. Es claramente un código análogo al código de validación 43 de metro. Supongo que se trata por tanto de un trasbordo de metro a metro ligero.
##########
Código: 68 **Número apariciones**: 3686

Literal:
Validación completa - Validación t. multiviaje en entrada sin descuento de viaje. Cancelado estación Las Tablas de METRO (salida) no excede periodo máximo permitido (40 min)

Medios de transporte:
Metro ligero

Suposición:
Variante del código 67 para título multiviaje.
##########
Código: 69 **Número apariciones**: 68268

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen-destino solamente en la barrera intermedia de Pinar de Chamartín no excede periodo viaje máximo permitido

Medios de transporte:
Metro ligero

Suposición:
Se da en cualquier estación de METRO LIGERO de la línea ML1. Parece indiciar una bajada en metro ligero con origen en "barrera intermedia de Pinar de Chamartín". El uso de barrera intermedia se debería referir a un cambio de tren, por lo que mi suposición es que se trata de una bajada en una estación de la línea ML1 donde la subida en la misma línea se realizó mediante un trasbordo de metro a metro ligero en Pinar de Chamartín (recordemos que Pinar de Chamartín es tanto una estación de metro como un extremo de línea de ML1 de metro ligero).
##########
Código: 70 **Número apariciones**: 25818

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen-destino solamente barrera intermedia Pinar de Chamartín no excede periodo viaje máximo permitido

Medios de transporte:
Metro ligero

Suposición:
Variante del código 69 para título multiviaje.
##########
Código: 71 **Número apariciones**: 7665

Literal:
Validación completa - Validación t. temporal salida descuento viaje y con origen - destino desconocido.

Medios de transporte:
Metro ligero

Suposición:
Variante del código 69 para origen desconocido.

##########
Código: 72 **Número apariciones**: 1769

Literal:
Validación completa - Validación t. multiviaje salida descuento viaje y con origen desconocido.

Medios de transporte:
Metro ligero

Suposición:
Variante del código 69 para título multiviaje y origen desconocido.
##########
Código: 73 **Número apariciones**: 47717

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen TFM - Rivas Urbanizaciones.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea especial de metro A9 con origen en Rivas Urbanizaciones.
##########
Código: 74 **Número apariciones**: 6006

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen TFM - Rivas Urbanizaciones.

Medios de transporte:
Metro

Suposición:
Variante del código 73 para título multiviaje.
##########
Código: 75 **Número apariciones**: 49969

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen TFM - Rivas Futuras.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea especial de metro A9 con origen en Rivas Futuras.
##########
Código: 76 **Número apariciones**: 8587

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen TFM - Rivas Futuras.

Medios de transporte:
Metro

Suposición:
Variante del código 75 para título multiviaje.
##########
Código: 77 **Número apariciones**: 15991

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen TFM - Rivas Vaciamadrid.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea especial de metro A9 con origen en Rivas Vaciamadrid.
##########
Código: 78 **Número apariciones**: 2258

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen TFM - Rivas Vaciamadrid.

Medios de transporte:
Metro

Suposición:
Variante del código 77 para título multiviaje.
##########
Código: 79 **Número apariciones**: 10950

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen TFM – La Poveda.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea especial de metro A9 con origen en Rivas Urbanizaciones.
##########
Código: 80 **Número apariciones**: 2343

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen TFM – La Poveda.

Medios de transporte:
Metro

Suposición:
Variante del código 79 para título multiviaje.
##########
Código: 81 **Número apariciones**: 33347

Literal:
Validación completa - Validación t. temporal salida sin descuento de viaje y con origen TFM – Arganda del Rey.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea especial de metro A9 con origen en Arganda del Rey.
##########
Código: 82 **Número apariciones**: 9824

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen TFM – Arganda del Rey.

Medios de transporte:
Metro

Suposición:
Variante del código 82 para título multiviaje.
##########
Código: 83 **Número apariciones**: 134992

Literal:
Validación completa - Validación t. multiviaje salida sin descuento de viaje y con origen - destino fuera del mismo ámbito.

Medios de transporte:
Metro

Suposición:
Bajada en una estación de la línea especial de metro A9 con origen fuera de dicha línea.
##########
Nota: no existen códigos del 84 al 95, el siguiente código después del 83 es el 96.

Código: 96 **Número apariciones**: 179909

Literal:
Validación de suplemento aeropuerto. Respecto del título suplementado, la validación ha sido previa en otro equipo y no ha superado el periodo de estancia en la red.

Medios de transporte:
Metro

Suposición:
Bajada en una de las estaciones de METRO en el aeropuerto con validación (es decir, subida) previa detectada. Además, "no se ha superado el periodo de estancia en la red", es decir, que el intervalo de tiempo de tiempo desde la subida hasta esta bajada se encuentra dentro de los márgenes aceptados por el sistema.
##########
Código: 97 **Número apariciones**: 6778

Literal:
Validación de suplemento aeropuerto. Respecto del título suplementado, la validación se hace en el mismo equipo, porque no hay una anterior o porque se ha superado el tiempo de estancia en la red.

Medios de transporte:

Metro

Suposición:
Versión alternativa del código 96 en la que, o bien no se ha detectado una validación (subida) anterior, o bien se ha superado el tiempo de estancia permitido en la red. Es de suponer que esto implica un coste diferente que en el caso del código 96.
##########
Código: 98 **Número apariciones**: 8799

Literal:
AENA multiviaje sin descuento de viaje ya que esta en el periodo de red no aplica suplemento.

Medios de transporte:
Metro

Suposición:
Variante para título multiviaje del código 96.
##########
Código: 99 **Número apariciones**: 64

Literal:
AENA multiviaje con descuento de viaje fuera periodo. Red no aplica suplemento.

Medios de transporte:
Metro

Suposición:
Variante para título multiviaje del código 97.
##########

Apéndice B

Mock-Up visor GIS

Se muestrán posibles mock-up para el visor GIS, cumpliendo con las especificaciones básicas descritas en el proyecto.

Figura B.1: Pantalla de inicio del visor GIS.

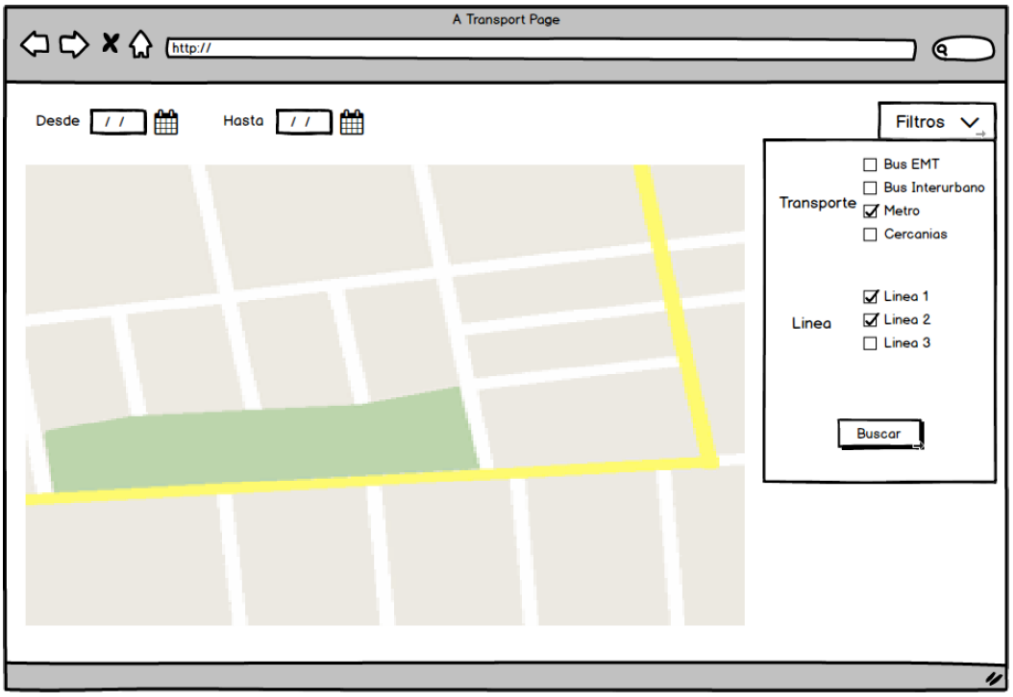

Figura B.2: Pantalla del visor donde se muestran todas las opciones de filtros disponibles.

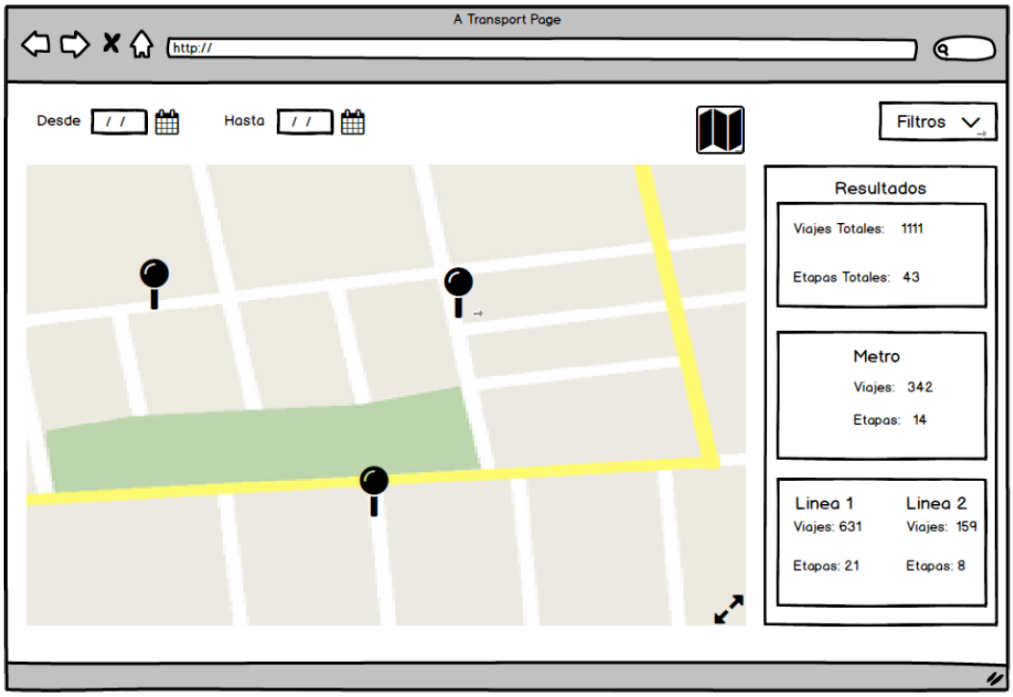

Figura B.3: Pantalla en la que aparecen: las estadísticas generales de una búsqueda a la izquierda y en el mapa la ubicación de las paradas a las que afecta esta búsqueda.

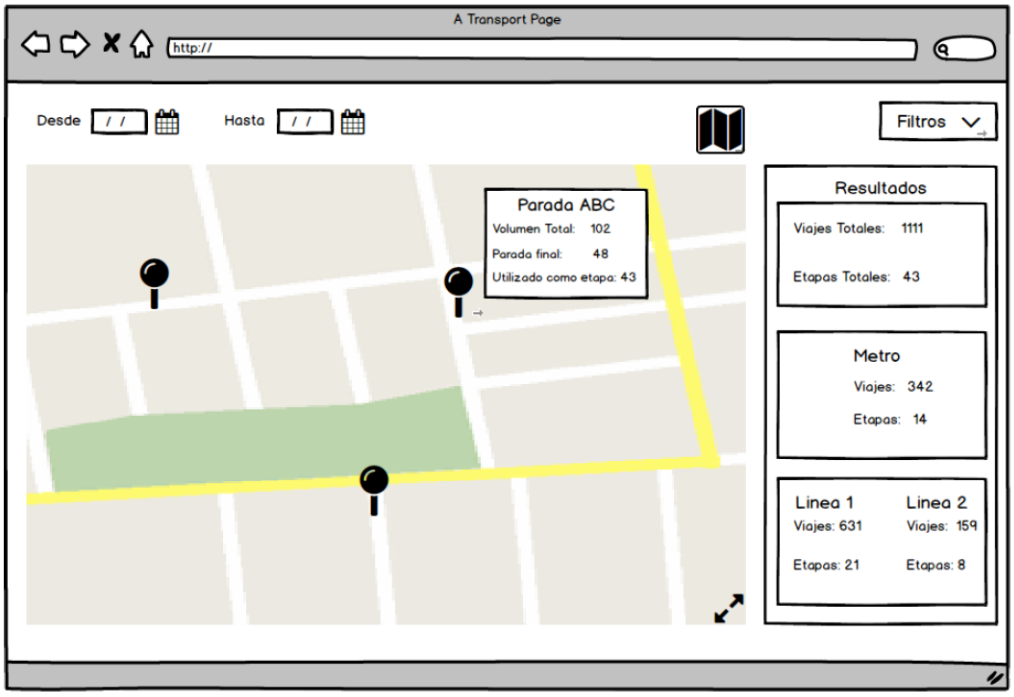

Figura B.4: Pantalla en la que se muestran los detalles específicos de una parada tras haber sido aplicados unos filtros.

Figura B.5: Pantalla que muestra el mapa de calor tras haber aplicado unos filtros concretos. Este indica el volumen de viajeros en las distintas zonas del mapa.

Lista de acrónimos

CRTM Consorcio Regional de Transportes de Madrid. 1, 2, 7, 10, 11, 17, 20–23, 36–38, 48

ETL Extract, transform and load. 36, 61

GTFS General Transit Feed Specification. 2, 4, 7, 17, 21–23, 37, 38, 48

IA Inteligencia Artificial. 1, 2, 5, 6, 10–13, 18, 23–26, 30, 31, 33, 37, 56, 57, 60–62

OTP OpenTripPlanner. 3, 7, 36, 38

PBF Protocolbuffer Binary Format. 38

ReLU Rectified Linear Unit. 51

SQL Structured Query Language. 1, 6, 62

Glosario

API Una API es una interfaz de programación de apliciones, y consiste en un conjunto de funciones y protocolos que permiten la comunicación entre varias aplicaciones y puede ser utilizada por un software externo. 38

deep learning El deep learning o aprendizaje profundo, consiste en un grupo de algoritmos de aprendizaje automático que tras procesar un gran volumen de datos, es capaz de aprender de manera autónoma y realizar tareas complejas como la identificación de patrones o la clasificación de elementos. 6

osmosis Apliación de Java por linea de comandos para procesar datos OSM. 38

query Consulta a una base de datos. 44, 46

red neuronal Ténica de inteligencia artifical inspirada en el funcionamiento del cerebro humano, y formada por un conjunto de neuronas artificales. Permite que aprenda por si mismo, y que en cada iteración corrija su errores para obtener cada vez mejores resultados. 6, 7, 12, 31

script Conjunto de comandos o instrucciones cuyo propósito es el de ejecutar un programa sencillo que realice diferentes funciones. 6, 12, 13, 26, 28, 30, 38, 62

Bibliografía

[1] A. Alsger, B. Assemi, M. Mesbah, and L. Ferreira, "Validating and improving public transport origin–destination estimation algorithm using smart card fare data," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 490–506, 2016.

[2] B. Assemi, A. Alsger, M. Moghaddam, M. D. Hickman, and M. Mesbah, "Improving alighting stop inference accuracy in the trip chaining method using neural networks," *Public Transp.*, vol. 12, no. 1, pp. 89–121, 2020. [En línea]. Disponible en: https://doi.org/10.1007/s12469-019-00218-9

[3] "Documentación sql server." [En línea]. Disponible en: https://docs.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver16

[4] "Documentación de la versión 3.8.10 de python." [En línea]. Disponible en: https://docs.python.org/release/3.8.10/

[5] "Documentación sublime." [En línea]. Disponible en: https://www.sublimetext.com/docs/

[6] "Documentación de csv." [En línea]. Disponible en: https://docs.python.org/3/library/csv.html

[7] "Documentacion de pandas." [En línea]. Disponible en: https://pandas.pydata.org/docs/

[8] "Documentacion de sklearn." [En línea]. Disponible en: https://scikit-learn.org/stable/

[9] "Documentación tensorflow." [En línea]. Disponible en: https://www.tensorflow.org/?hl=es-419

[10] "Documentación keras." [En línea]. Disponible en: https://keras.io/

[11] "Documentación matplotlib." [En línea]. Disponible en: https://matplotlib.org/

[12] "Documentación draw io."

[13] "Documentación latex." [En línea]. Disponible en: https://www.latex-project.org/help/documentation/

[14] "Documentación overleaf." [En línea]. Disponible en: https://es.overleaf.com/learn

[15] "Metodología iterativa incremental." [En línea]. Disponible en: https://proyectosagiles.org/desarrollo-iterativo-incremental/

[16] A. Géron, *Hands-on Machine Learning with Scikit-Learn, keras TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'REILLY, 2019.

[17] A. C. M. . S. Guido, *Introduction to Machine Learning with Python.* O'REILLY, 2016.

[18] "Implementación de osmosis." [En línea]. Disponible en: https://github.com/openstreetmap/osmosis

[19] "Documentación api de otp." [En línea]. Disponible en: http://dev.opentripplanner.org/apidoc/2.1.0/index.html