GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA BIG DATA PARA LA IDENTIFICACIÓN Y PREDICCIÓN DE ZONAS POTENCIALMENTE PELIGROSAS PARA CICLISTAS

Autor: Pablo Mena Gómez de Merodio

Director: David Contreras Bárcena

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

"Sistema Big Data para la identificación y predicción de zonas potencialmente peligrosas para ciclistas"

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2018/19 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Pablo Mena Gómez de Merodio Fecha: ……/ ……/ ……

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena Fecha: ……/ ……/ ……

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D._____

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: _____, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a …….. de ……... de …….

ACEPTA

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA BIG DATA PARA LA IDENTIFICACIÓN Y PREDICCIÓN DE ZONAS POTENCIALMENTE PELIGROSAS PARA CICLISTAS

Autor: Pablo Mena Gómez de Merodio

Director: David Contreras Bárcena

Madrid

Agradecimientos

En primer lugar, tengo que agradecerles a mis padres el gran apoyo que me han dado a lo largo de la carrera y, muy especialmente, a lo largo de este último año. Nunca han dejado de ayudarme y siempre han mostrado un gran interés por mis estudios a pesar de no entender la mitad de las cosas de las que hablo.

También tengo que agradecérselo a mi hermana por ser una fuente de inspiración y por estar dispuesta a revisar mis trabajos, pero sobre todo por las charlas y las risas que hemos pasado todos estos años y que espero que no acaben nunca.

A mi director David, que me ha guiado a lo largo de este proyecto y que despertó en mí el interés por la programación. A mis compañeros de trabajo Beltran y Cayetano con los que he pasado horas revisando líneas de código y tomando cerveza a partes iguales.

Y finalmente a mis amigos, a los nuevos y a los viejos. Por ayudarme a desconectar, por sacarme de casa, por soportar mis quejas y por mil cosas más que no se pueden explicar.

SISTEMA BIG DATA PARA LA IDENTIFICACIÓN Y PREDICCIÓN DE ZONAS POTENCIALMENTE PELIGROSAS PARA CICLISTAS

Autor: Mena Gómez de Merodio, Pablo.
Director: Dr. Contreras Bárcena, David.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto se ha desarrollado el módulo de análisis de datos y predicción de comportamiento dentro de una infraestructura Big Data para el cálculo de rutas seguras a través de la ciudad de Madrid. El proyecto se centra en la implementación de un sistema de predicción capaz de identificar zonas potencialmente peligrosas para los ciclistas, basándose en históricos de trayectos y accidentes [1], así como en información en tiempo real ofrecida por el usuario.

Palabras clave: Big Data, Machine Learning, predicción de accidentes,

1. Introducción

La gran mayoría de los sistemas de navegación desarrollados hasta el momento se centran principalmente en encontrar la ruta más corta entre los puntos de origen y destino. Sin embargo, no existe ninguno que tenga en cuenta la seguridad vial como factor influyente en el cálculo de la ruta. Este proyecto pretende desarrollar un sistema de predicción que analice registros sobre desplazamientos y accidentes, así como información meteorológica y de polución, para ofrecer al usuario una ruta lo más corta posible evitando zonas con alta probabilidad de accidentes.

2. Definición del proyecto

Los puntos clave para el funcionamiento del proyecto son los siguientes:

1. Realizar un mapeado de la zona de circulación de las bicicletas BiciMad para poder desarrollar un programa de trazado de rutas que ofrezca la ruta más corta entre dos puntos [3].
2. Desarrollar un análisis estadístico básico de los datos a utilizar mediante el uso del lenguaje de programación R.

3. Programar un modelo de machine learning mediante el uso de la librería de Google TensorFlow. El modelo utiliza como datos de entrenamiento aquellos recopilados en una fase previa del proyecto. Entre los datos de entrenamiento destacan: Información sobre las características de las calles, datos sobre el usuario, valores de polución e información sobre las condiciones meteorológicas.
4. Desarrollar el *backend* de la aplicación para que, a partir de las estaciones de origen y destino, se ofrezca la ruta más corta y aquella que suponga menor riesgo para el usuario.

3. Descripción del sistema

Este proyecto utiliza como punto de partida los datos recopilados en el TFG "Ingesta de datos en proyecto Big Data". Partiendo de los resultados obtenidos en dicho proyecto, se ha diseñado un modelo de machine learning utilizando las librerías de PySpark, Keras y TensorFlow. El modelo generado aprenderá utilizando la técnica del aprendizaje supervisado y el análisis de componentes independientes y una vez que se obtengan unos resultados válidos, podrá implementarse en el programa de trazado de rutas.

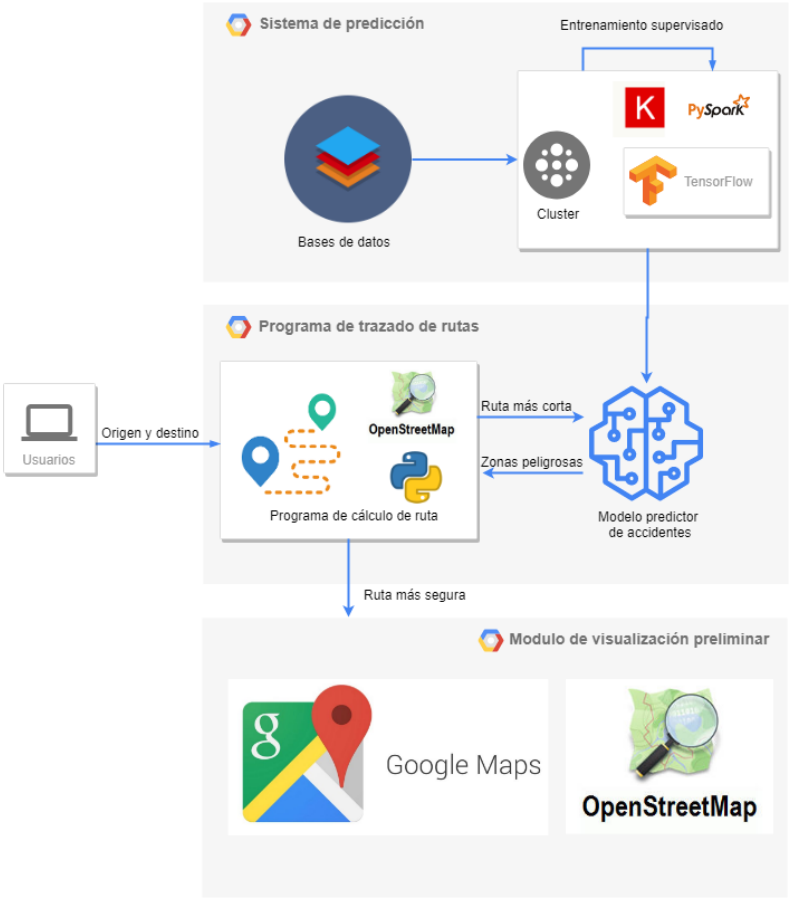


Figura 1 Diagrama de la arquitectura del sistema

Como se puede apreciar en la ilustración, el usuario introducirá las coordenadas de origen y destino y el programa trazará la ruta más corta. Dicha ruta se pasará al modelo de machine learning que devolverá las zonas a evitar y se trazará una nueva ruta. Finalmente, las rutas se visualizarán utilizando las librerías de OSMNX [2] y Google Maps.

4. Resultados

Como resultado final del proyecto, se obtienen las relaciones estadísticas de las variables empleadas, así como un modelo funcional capaz de predecir las zonas potencialmente peligrosas para los ciclistas. Dicho modelo sirve como backend para un futuro módulo de visualización implementado en otro proyecto. Sin embargo, ha sido implementado un sencillo programa que permite obtener una representación visual de las rutas y compararlas con el histórico de accidentes de la ciudad de Madrid.

En las siguientes imágenes se representa la ruta más rápida entre dos puntos (el cruce de la calle Galileo Galilei con Rodríguez San Pedro y la calle Pez en la ciudad de Madrid) y las zonas potencialmente peligrosas detectadas por el sistema. Todo ello sobre un mapa de calor que representa los accidentes de bicicletas ocurridos durante los últimos años.

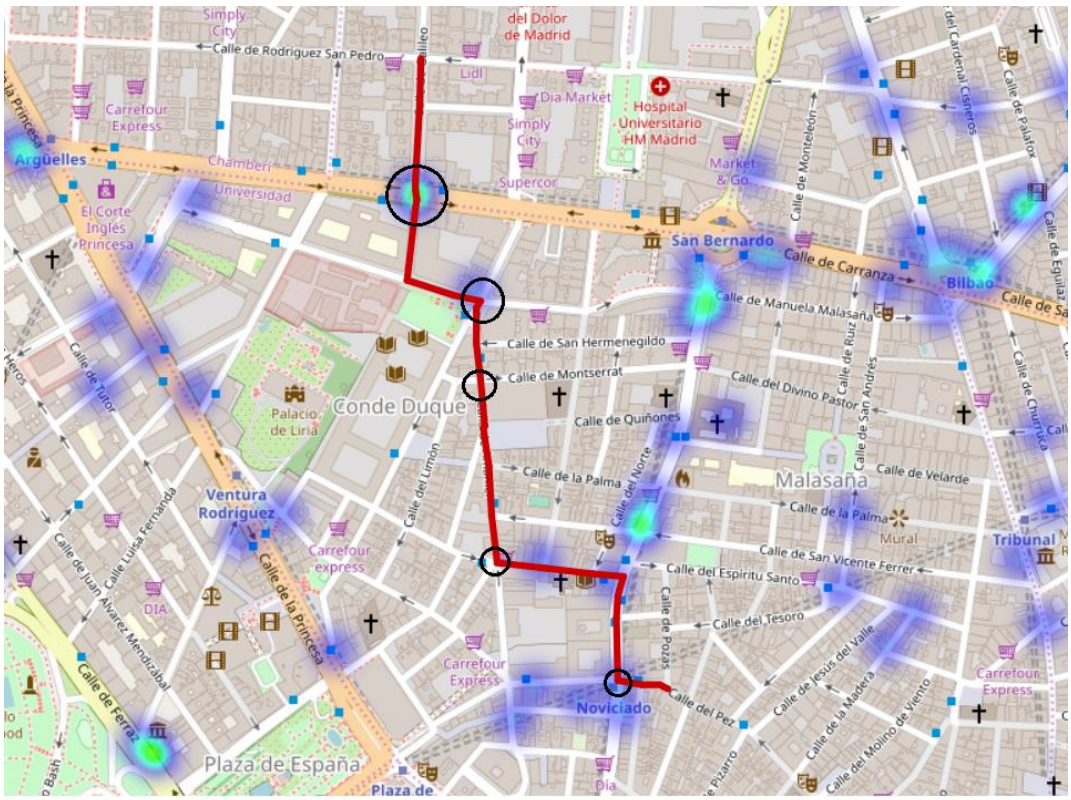


Figura 2. Ruta más corta y focos de peligro detectados por el programa

Partiendo de las zonas identificadas como peligrosas, el sistema ofrece una ruta alternativa que evita los focos y reduce la distancia. Finalmente, el programa genera una segunda iteración para evitar los posibles peligros de la nueva ruta, ofreciendo una alternativa más larga pero también más segura que la original.

Este resultado se puede apreciar en la siguiente imagen, donde la ruta roja es la más corta, la verde es la más segura y la azul ofrece un equilibrio entre ambas.

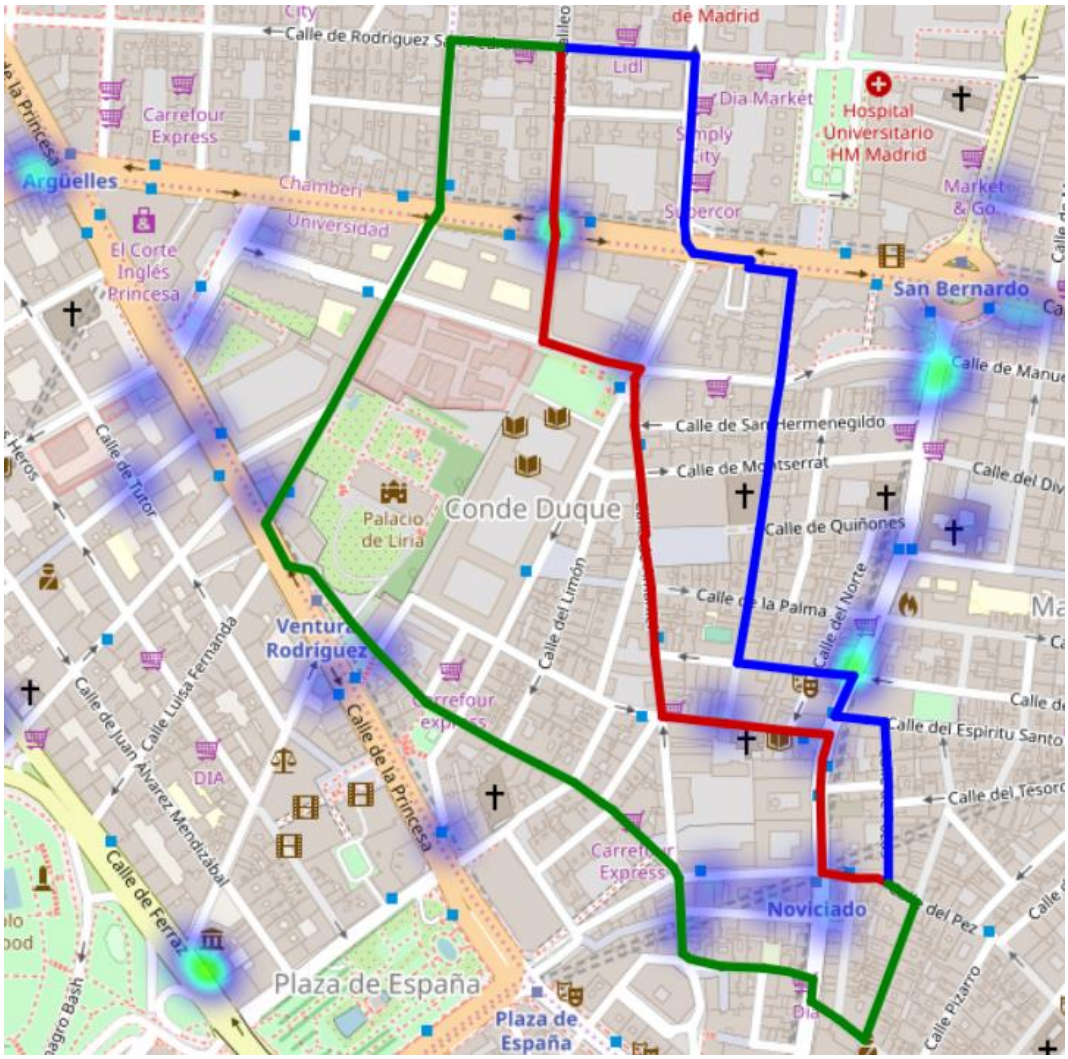


Figura 3. Comparativa de rutas calculadas

5. Conclusiones

Como conclusión, se ha logrado cumplir con todos los objetivos establecidos en el proyecto. Es decir, se ha desarrollado un algoritmo de machine learning capaz de

predecir las posibilidades de sufrir un accidente, así como de calcular la siguiente ruta más óptima; se han calculado las relaciones entre las distintas variables y se ha comprobado que existe una correlación real entre los accidentes y los datos utilizados para el proyecto.

6. Referencias

[1] A. d. Madrid, «datos.madrid,» Ayuntamiento de Madrid, 2018. [En línea]. Available: https://datos.madrid.es/portal/site/egob.

[2] «Open Street Map,» [En línea]. Available: https://osmnx.readthedocs.io/en/stable/.

[3] J. M. A.-J. a. A. Becerra-Terón, «Distance Based Queries in Open Street Map,» de *International Workshop on Database and Expert Systems Applications (DEXA)*, Valencia, 2015.

SISTEMA BIG DATA PARA LA IDENTIFICACIÓN Y PREDICCIÓN DE ZONAS POTENCIALMENTE PELIGROSAS PARA CICLISTAS

Author: Mena Gómez de Merodio, Pablo.

Director: Dr. Contreras Bárcena, David.

Colaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this project, the analysis and prediction module inside a Big Data system has been developed. The main objective of this project is to calculate the safest routes for cyclist through the city of Madrid. The system is focused on implementing a prediction method capable of identify potentially dangerous zones using the route and accident records provided by the City Hall [1] and the real time data provided by the users

Keywords: Big Data, Machine Learning, accident prediction

1. Introduction

Almost every navigation system that has been developed so far are focused on finding the shortest path between the origin and the destination. However, there is no system that takes into account the safety of the driver as a critical factor when the route is calculated. This project pretends to develop a system capable of study the recorded routes and the registry of accidents together with the weather and pollution records, in order to offer the shortest path that avoids the points with a high probability of suffer an accident

2. Objectives

The main objectives established for this project are presented below.

1. Perform a map of the zone where the BiciMad bicycles can move in order to develop a route program that can find the shortest path between two points [3].
2. Develop a basic statistical analysis of the abailable data with the programming language R
3. Programming a machine learning model using the TensorFlow API developed by Google. The model uses as training data the information recorded in a previous

phase of the project. The data used to train the model includes routes, information related with the streets, user information, pollution values and weather information

4. Develop the backend for the app in order to offer the shortest and safest route from one BiciMad station to another

3. System description

This project uses, as starting point, the information recorded in the final project "Ingesta de datos en proyecto Big Data". Using the results of that project, it has been developed a machine learning model using the PySpark, Keras and TensorFlow APIs. The model will be trained using supervised learning and the independent components analysis. Once that the results of the predictions are good enough for our standards, it will be implemented inside the tracing range path.

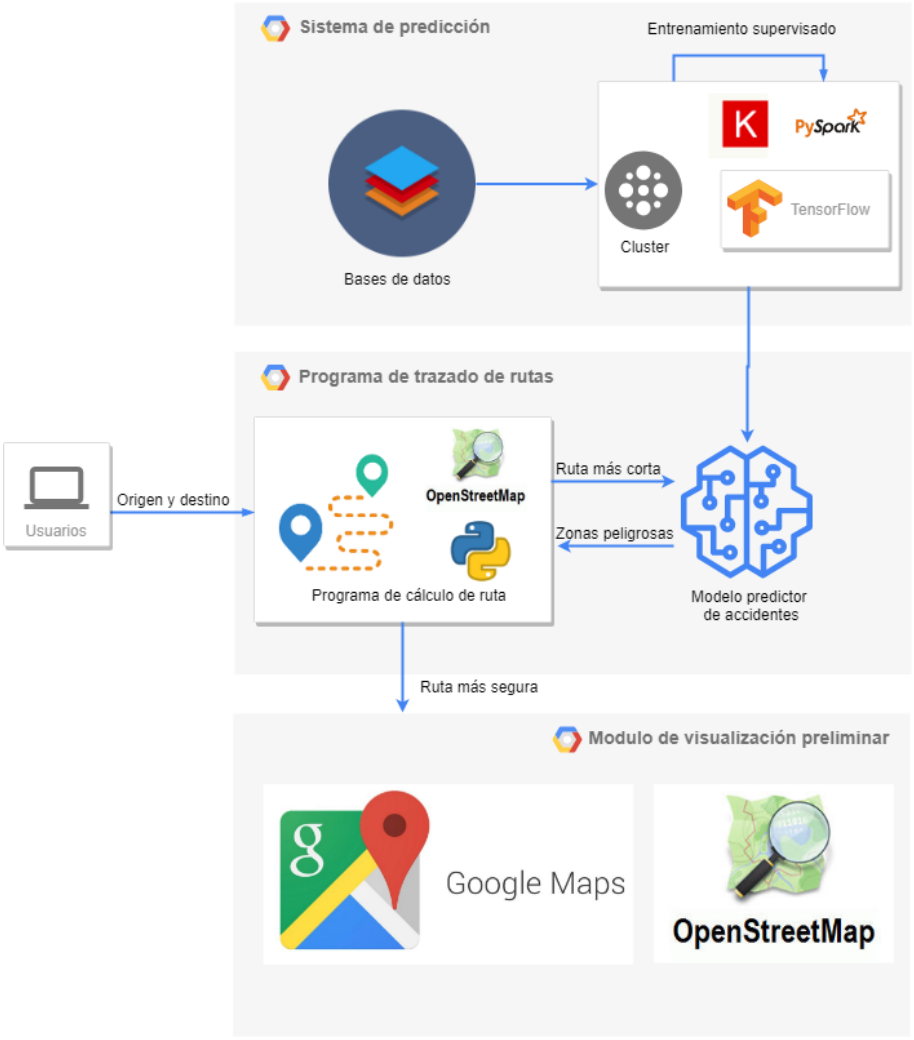


Figura 4 Diagrama de la arquitectura del sistema

As it can be seen in the previous image, the user will introduce the coordinates of the start and end point and the program will find the shortest path between them. That route will be sent to the machine learning model that will return the points that must be avoided and a new path will be calculated. At the end, all the routes that have been used in the program will be drawn by tha visualization module with the OSMNX and Google Maps APIs.

4. Results

At the end, the program will return the statistical relations between the variables and a machine learning model capable of predict the most dangerous places for bicycle riders. This model will be used as the backend for a future visualization program that will be developed in another project. Nevertheless, it has been implemented a simple program capable of draw the routes and compare them with the accident records of Madrid.

In the next images it can be seen the shortest route between two points and the most dangerous zones detected by the system. All of it draw over a heat map that represents the crashes of bicycles of the last few years.

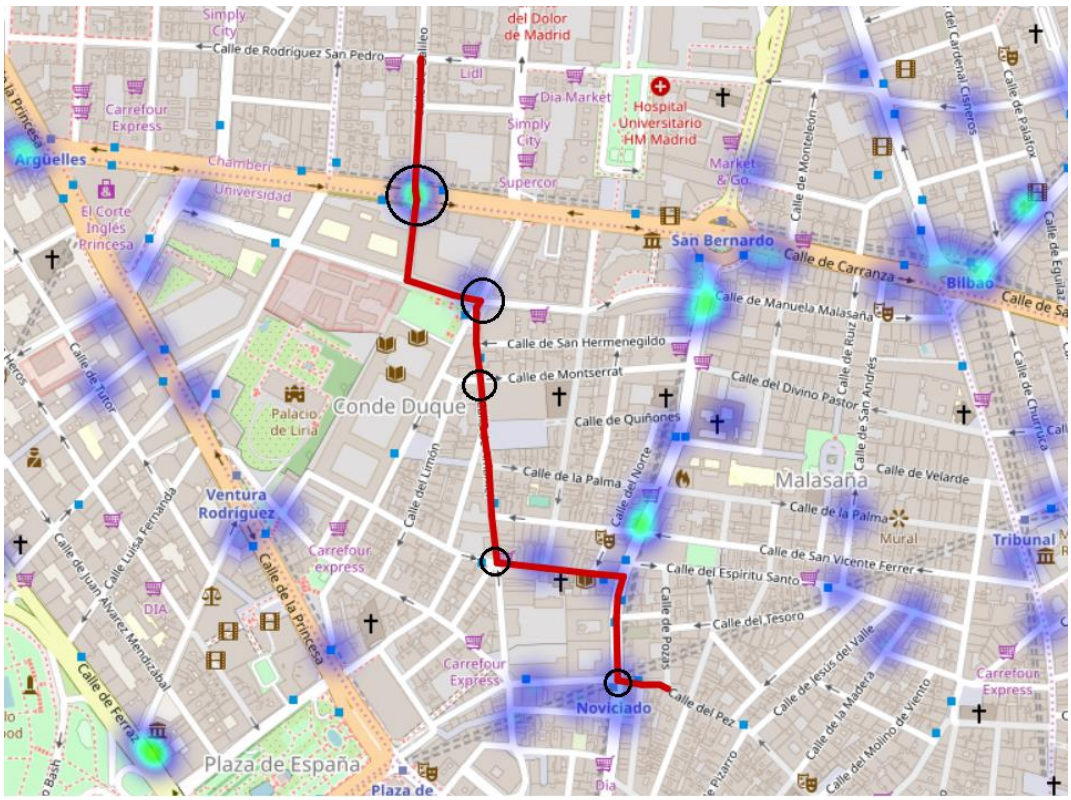


Figura 5. Ruta más corta y focos de peligro detectados por el programa

Using the places finded by the program, the system returns an alternative route that avoids the most dangerous places and reduce the distance. At the end, the program generates a second iteration to avoid the possible dangers of the new route, returning an even safer path in change of a bigger length.

This fact can be seen in the next image where the red route is the fastest, the green one is the safest and the blue one is a middle point

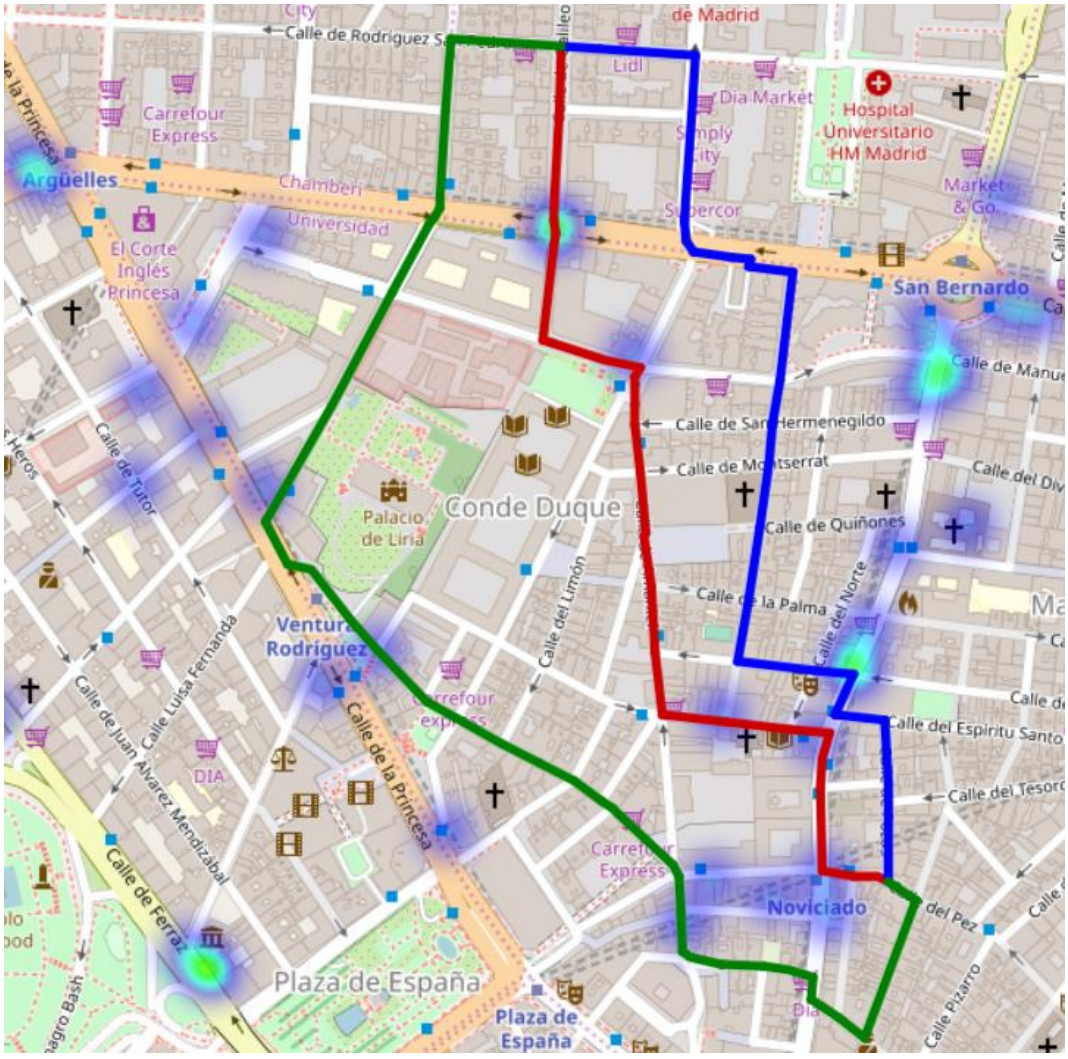


Figura 6. Comparativa de rutas calculadas

5. Results

As conclusion, it has been accomplished all the objectives of the project. It has been implemented an algorithm capable of predict the probabilities of suffer an accident and calculate the following best route; the correlation between variables has been calculated

and it has been proven that there is a real relation between the accidents and the data used in this project

6. Referencias

[1] A. d. Madrid, «datos.madrid,» Ayuntamiento de Madrid, 2018. [En línea]. Available: https://datos.madrid.es/portal/site/egob.

[2] «Open Street Map,» [En línea]. Available: https://osmnx.readthedocs.io/en/stable/.

[3] J. M. A.-J. a. A. Becerra-Terón, «Distance Based Queries in Open Street Map,» de *International Workshop on Database and Expert Systems Applications (DEXA)*, Valencia, 2015.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

13.551.837. Ese fue el número de desplazamientos en bicicletas de la iniciativa BiciMad que se registraron en el año 2018. Esto significa que, durante el último año, hubo una media de más de 37 mil desplazamientos diarios por la capital. En el 2017, hubo unos 200 mil pedidos mensuales a aplicaciones como Glovo o Deliveroo, las cuales usan bicicletas para realizar la mayoría de sus encargos. A estas cifras hay que añadirle todos los trayectos realizados por bicicletas individuales no registradas, cuyo número se puede suponer que es, como mínimo, equivalente al de las bicicletas registradas. Otro síntoma del innegable auge de las bicicletas y otros medios de transporte similares es la aparición de varias aplicaciones semejantes a BiciMad como oBike o los servicios de alquiler de patinetes eléctricos que tanta controversia generó en Madrid a principios de año.

El incremento en el uso de las bicicletas en general, y en particular de las bicicletas eléctricas, es una realidad. Todo comenzó en el año 2014, cuando el ayuntamiento de Madrid instaló el sistema de alquiler de bicicletas de BiciMad, ofreciendo a cualquiera que estuviese interesado, una alternativa de transporte limpia y eficiente. Desde entonces, el uso de bicicletas en Madrid no ha dejado de crecer. Sin embargo, como ocurre con cualquier medio de transporte, el aumento en su uso ha conllevado un inevitable incremento en el número de accidentes de tráfico [1].

De todos los vehículos que circulan por las carreteras, las bicicletas son las más vulnerables en caso de accidente, debido a que son las que cuentan con menor protección. Factores como las calles por las que se circula, el tiempo atmosférico, el estado de la carretera, la edad del usuario o el tipo de trayecto, entre otros, influyen en la probabilidad de tener un accidente al conducir una bicicleta.

En países como Holanda o Japón, donde el uso de bicicletas como medio de transporte es muy común y donde se registran gran cantidad de accidentes (206 muertos en

Ámsterdam por accidentes de bicicletas en 2017 [1] y más de 11 mil accidentes con bicicletas en 2015 en Tokio) se han desarrollado sistemas de detección de zonas de alto riesgo para ciclistas mediante el uso de técnicas basadas en machine learning.

Aprovechando la gran cantidad de información sobre trayectos y accidentes de bicicletas que ofrece el ayuntamiento de Madrid y basándonos en la idea de Shigeo Kaneda de la universidad de Doshiha [2]. El objetivo de este proyecto es diseñar y desarrollar un sistema de predicción de accidentes para trayectos en bicicleta en la ciudad de Madrid, que permita ofrecer una ruta alternativa en caso de peligro para el conductor.

1.1 MOTIVACIÓN DEL PROYECTO

Como ha quedado patente, las bicicletas son un factor a tener en cuenta como medio de transporte dentro de nuestra sociedad. Es por ello que la relevancia de los accidentes de tráfico con bicicletas implicadas no es algo que se pueda ignorar. Es previsible que en los próximos años el número de usuarios y la cantidad de medios de transporte alternativos aumente y la situación se agrave, lo que podría tener consecuencias catastróficas.

Al analizar los registros de accidentes de bicicletas de los últimos años, quedó patente como las incidencias se concentraban en zonas determinadas. En la figura 9, se puede apreciar claramente este hecho.

Figura 7. Mapa de calor de los accidentes de tráfico con implicación de bicicletas

Se puede ver como gran cantidad de accidentes se concentran en calles principales como la Gran Vía o Atocha, así como en rotondas o zonas de mayor densidad de tráfico . Analizando la información disponible sobre accidentes e información en tiempo real, se pueden realizar predicciones que ayuden al usuario a encontrar una ruta más segura en un tiempo mínimo.

Otra motivación consiste en encontrar las causas por las que las medidas actuales no son todo lo eficientes que deberían. Un ejemplo de este hecho se puede apreciar en la figura 10. En ella están representados los accidentes (puntos negros), las calles de un único sentido (rojo), de doble sentido (azul) y los carriles bici (verde). Como se puede ver y a diferencia de lo que sería lógico pensar, los accidentes en vías con carriles bici son bastante numerosos, lo que indica claramente que existe algún problema con las infraestructuras ya sea a nivel estructural o de uso.

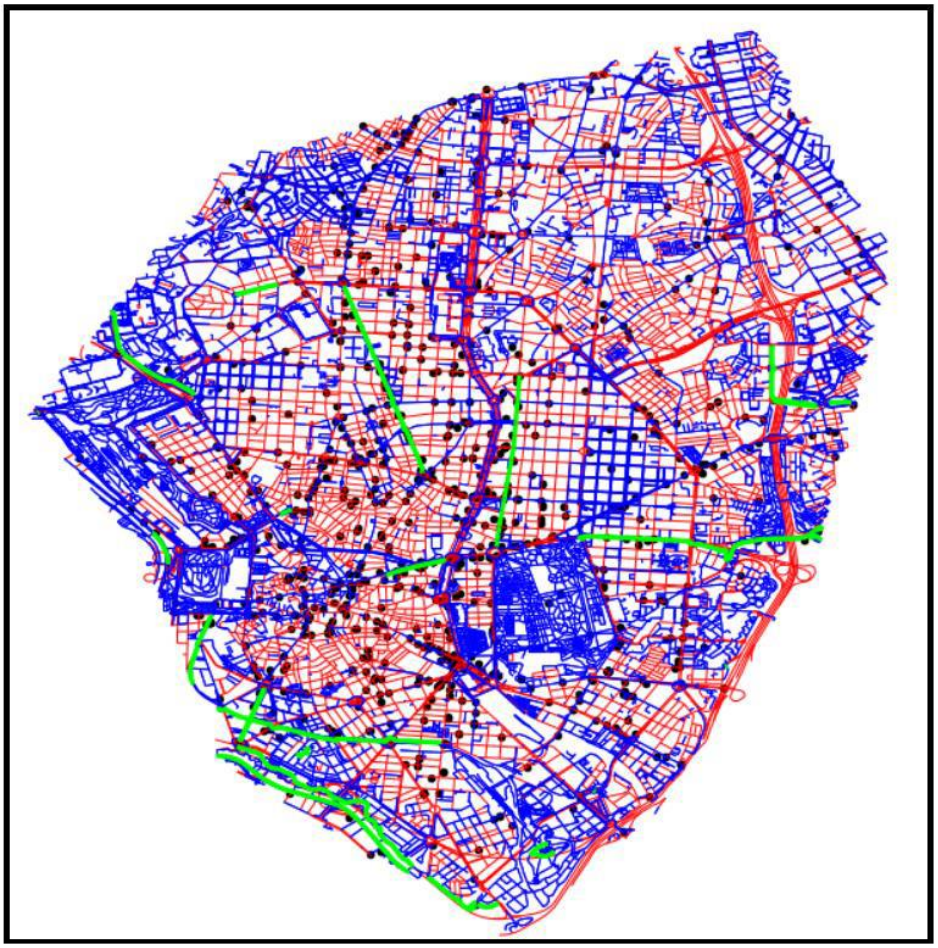


Figura 8. Mapeado de la ciudad de Madrid con accidentes y tipo de vía

El objetivo final de este proyecto consiste es obtener una aplicación que sea capaz de predecir, con un porcentaje alto de acierto, las posibilidades de tener un accidente al circular por la ciudad con una bicicleta y ofrecer a los usuarios una alternativa que afecte lo menos posible al tiempo de ruta.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

- **Big Data**

  Conjunto de datos de gran tamaño, complejidad y velocidad de crecimiento que no pueden ser gestionados mediante los medios habituales debido a su gran volumen. También es la base en la que sustenta la tecnología del machine learning

- **Machine Learning**

  Rama de la inteligencia artificial que utiliza grandes cantidades de datos para detectar patrones, realizar predicciones y tomar decisiones sin necesidad de que estas sean programadas directamente por un humano. Este concepto también implica que los sistemas se mejoran de forma automática mediante la práctica y el procesado de nuevos datos

- **Cluster**

  Conjunto de máquinas que comparten un hardware, pero que trabajan como un único ordenador. Es capaz de almacenar y procesar datos de forma distribuida y nos otorga la capacidad de procesamiento necesaria para realizar nuestro proyecto

- **Dataframe**

  Objeto utilizado para almacenar cantidades masivas de datos mediante el uso de columnas. Es equivalente a una tabla, pero permite almacenar la información de forma distribuida

- **Función map**

  Función especializada de Spark que permite paralelizar procesos y procesar grandes cantidades de datos en poco tiempo

- **Jupyter notebook**

  Plataforma *open-source* que ofrece soporte interactivo para el análisis de datos en cualquier lenguaje de programación. Permite realizar visualizaciones de los datos y facilita el análisis de resultados

- **Open Street Map (OSMNX)**

  Librería de Python gratuita que ofrece información actualizada y fiel sobre mapas y calles de cualquier lugar del mundo. Los datos son recolectados por los usuarios y aporta gran variedad de información sobre las ciudades. Gracias a ella hemos podido obtener información sobre la topografía de la ciudad, lo que ha sido clave para el desarrollo del proyecto [3].

- **TensorFlow**

  Librería desarrollada por Google con el objetivo de facilitar el desarrollo de modelos de machine learning tanto a bajo como a alto nivel.

- **Pyspark**

  Librería Python que adapta Apache Spark [3], *framework* de computación *open-sourc*e en *clusters*. Ofrece una lectura paralelizada de datos distribuidos trabajando en un entorno con tolerancia a fallos.

- **Hadoop**

  Open *framework* que permite almacenar cantidades masivas de datos en un entorno distribuido y procesarlos en paralelo mediante el uso de un *cluster*

- **HDFS**

  Sistema de almacenamiento distribuido de Hadoop. Cuenta con una estructura maestro esclavo y almacena la información en varios bloques de forma que se evita la pérdida de datos en caso de fallo.

- **YARN**

  Plataforma para la negociación de recursos en un *cluster* Hadoop, es el responsable de distribuir los procesos a través de los nodos y devolver al programa original todos los datos unidos.

- **Portal de datos**

  Plataforma del ayuntamiento de Madrid [4] en la que se cuelga información relevante para la ciudad y de la que hemos extraído los datos más críticos de nuestro sistema.

Capítulo 3. ESTADO DE LA CUESTIÓN

Desde hace ya varios años, no han parado de surgir aplicaciones y sistemas de navegación GPS para encontrar la ruta más eficiente entre origen y destino. La mayoría de estas aplicaciones, por no decir todas, se basan en el principio de obtener la ruta más corta o aquella que requiera menos tiempo. Y es entorno a esa idea que ha ido evolucionando el campo de los sistemas de navegación.

Los algoritmos de Dijkstra y Bellman-Ford para generar el camino más corto en un grafo, sentaron las bases de la que es, a día de hoy, una de las tecnologías más utilizadas en todo el mundo. Con el paso del tiempo, tanto los algoritmos de búsqueda [5], como las técnicas empleadas para obtener información sobre las rutas han ido evolucionando.

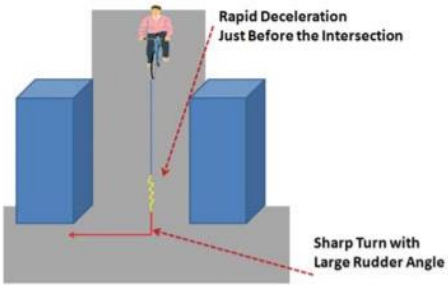
En 2005 Google presentó Google Maps después de comprar Where 2 Technologies. No fue la primera página que digitalizó mapas, pero aun así se ha convertido en el referente a nivel mundial gracias a su alcance y versatilidad. Para intentar ponerse a su altura, Nokia compró en 2007 la empresa Navteq por 8100 millones de dólares, la cual se dedicaba al desarrollo de sensores de tráfico y al mapeo de carreteras. Un año más tarde hizo su aparición la empresa israelí Waze Ltd, que cambió por completo el sector y desbanco a sus competidores, provocando que Nokia pasase de estar valorada de 140.000 millones a 8.100 millones en tan solo 4 años.

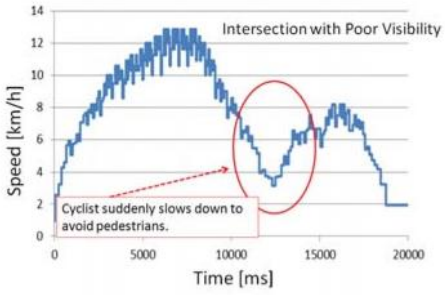
Lo que marcó la diferencia entre Waze y las demás empresas de navegación fue el cambio de paradigma en la forma de obtener la información. A diferencia de los softwares de navegación previos a su aparición, Waze era mantenido por los usuarios, eran ellos los que recolectaban información mediante el GPS de sus teléfonos móviles y eran los propios usuarios los que informaban sobre atascos, accidentes o controles. Gracias a este sistema la empresa recolectaba información y aprendía de sus clientes para ofrecer información útil. La navegación pasaba de ser estática a dinámica.

Desde entonces y hasta ahora, los datos se han vuelto más complejos y los usuarios ya no son únicamente receptores de información, si no también emisores, como se mencionó anteriormente. Ahora mismo los sistemas dependen de la información que les envíen los usuarios. Datos como la velocidad de desplazamiento o la densidad de usuarios en un área determinada, ayudan a los sistemas a calcular que ruta es la más adecuada en cada situación.

Basándose en la idea de obtener información sobre las rutas a partir de los usuarios, Shigeo Kaneda y su equipo publicaron el artículo "A Hazard Detection Method for Bicycles by Using Probe Bicycle" [2]. En él se explica cómo se han utilizado bicicletas de prueba para analizar las reacciones de los usuarios al circular por la ciudad.

Partiendo del principio de que la velocidad y el nivel de brusquedad en los giros sigue un patrón en las zonas con riesgo de accidente y gracias a las nuevas tecnologías que permiten a los sistemas analizar grandes cantidades de datos y aprender por sí mismos, fueron capaces de localizar zonas potencialmente peligrosas para las bicicletas que no habían sido detectadas hasta entonces.





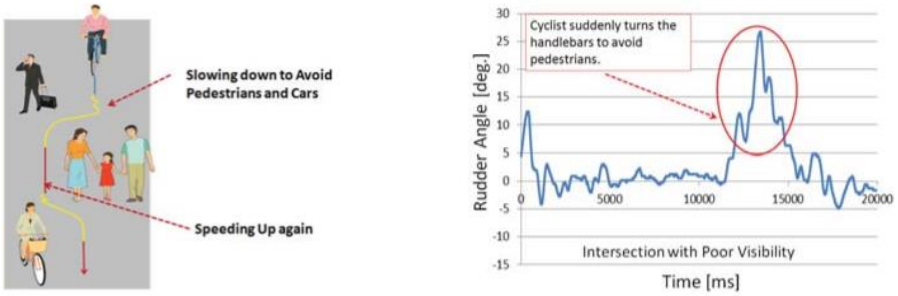


Figura 9. Imágenes y gráficas extraídas del estudio "A Hazard Detection Method for Bicycles" del profesor Shigeo Kaneda

Como demostró el trabajo del profesor Kaneda, la herramienta clave para el desarrollo de la nueva generación de software de navegación es, sin duda, el machine learning. La capacidad de analizar los datos y hacer que el sistema aprenda por sí solo abre todo un horizonte de posibilidades.

Ya existen precedentes en otros campos como la medicina, donde se utilizan tecnologías basadas en machine learning para identificar y predecir la gravedad de los linfomas [6]. O en economía, que usa sistemas de predicción para realizar una estimación de cómo variará el mercado [7]. Los mismos principios en los que se basan estas tecnologías se pueden aplicar para predecir la situación del tráfico.

En 2017 Google comenzó a trabajar en un sistema para predecir la disponibilidad de aparcamiento en 50 ciudades de todo el mundo [8]. Registros de aparcamiento, el tiempo empleado en la búsqueda de aparcamiento o los trayectos compartidos por los usuarios de Google Maps, son parte de los datos que usa el sistema para realizar sus predicciones.

Viendo todos estos logros que ha alcanzado el machine learning y a raíz del boom que ha impulsado esta tecnología en los últimos años, cabe pensar que se trata de algo nuevo que no se había desarrollado hasta ahora. Sin embargo, el machine learning es una rama de la inteligencia artificial que lleva existiendo desde hace varios años, pero que, por desgracia,

ha sufrido duros reveses a lo largo de su historia, llegando a ser un campo estancado durante casi una década.

Los primeros pasos en el machine learning empezaron en la década de los 50, pero probablemente el ejemplo más conocido surgió en 1997, cuando IBM desarrolló el ordenador Deep Blue II [9], que ganó al campeón mundial de ajedrez Gary Kasparov. El año previo al encuentro, Kasparov logró derrotar a la versión anterior de Deep Blue en 4-2, tras lo cual IBM realizo algunas modificaciones en su sistema, pero manteniendo el principio central de usar machine learning. El ordenador de IBM se basaba en arboles de decisión, que analizaban los movimientos futuros óptimos para cada posición del tablero. Para ello, se analizaron y registraron millones de partidas de ajedrez. El ordenador contaba con 30 procesadores y 16 chips especializados en el procesamiento de ajedrez, que eran capaces de analizar más de dos millones de posiciones por segundo. De hecho, en situaciones donde existían varios movimientos forzosos consecutivos, el programa era capaz de analizar 100 millones de posiciones por segundo. Durante la partida contra Kasparov se alcanzaron los 330 millones de posiciones por segundo.

Su estructura era muy similar al funcionamiento de un cluster con Hadoop. Existía un maestro que distribuía el trabajo a los esclavos y estos realizaban el procesado de forma paralela en cada nodo, igual que trabajamos hoy en día.

Desde entonces, el campo del machine learning ha crecido sin parar. En los últimos 20 años ha habido avances en campos tan variados como el reconocimiento facial, el reconocimiento de imágenes, la conducción o la traducción entre otros. Aprovechando el ejemplo inicial de Deep Blue, podemos usar otro caso del mismo campo para medir el avance de esta tecnología a lo largo de los años. Estamos hablando de AlphaGo Lee.

AlphaGo Lee es el proyecto de Google DeepMind que cogió el relevo de Deep Blue. A diferencia de su predecesor, AlphaGo no juega al ajedrez sino al Go, un juego de estrategia de origen chino más complejo y amplio que el ajedrez. Mientras que el tablero de ajedrez es

de 8x8 él de go es de 19x19, lo que significa que existen 10^{170} posibles posiciones, más del triple que en él ajedrez.

A diferencia de Deep Blue, AlphaGo no solo usaba una estructura en árbol, también explotaba varias redes neuronales que permitían una mejor selección de movimientos, pero la mayor diferencia reside en el hecho de que no solo usó partidas jugadas por profesionales como material de entrenamiento (entrenamiento supervisado), sino que también jugó contra si mismo para aprender, lo que resulto ser la clave para un mejor aprendizaje. La prueba definitiva fue la victoria contra Lee Sedol, el campeón del mundo de Go, en 2016, cuando la inteligencia artificial fue capaz de ganar al campeón 4-1 [10].

La última versión de AlphaGo es AlphaGo Zero, la cual ganó a su predecesor después de tan solo 36 horas de entrenamiento y sin usar partidas registradas, simplemente jugando contra sí mismo (Aprendizaje no supervisado).

Los casos ilustrados en los párrafos previos utilizaban arboles de decisión como principio básico. Sin embargo, existe una gran variedad de algoritmos usados en el campo del machine learning, y a la hora de desarrollar un proyecto Big Data es de vital importancia seleccionar el modelo que mejor se adapte a cada caso de uso. Al no existir ningún sistema perfecto, es necesario realizar un análisis previo de los datos de los que disponemos y de los resultados que esperamos obtener. De esta forma, es posible seleccionar el algoritmo que ofrezca los mejores resultados.

El sistema de predicción más sencillo es la regresión lineal. Este método consiste en trazar una recta que se encuentre lo más cerca posible de todos los datos. Para encontrar esta recta se tienen en cuenta todos los campos y se les asigna un peso el cual se va ajustando poco a poco con cada iteración. La fórmula matemática es la siguiente, donde *y* es el valor predicho, *b* es la ordenada en el origen, *w* son las ponderaciones y *x* son los atributos

$$y = b + w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n$$

Los pesos de las ponderaciones van cambiando con cada iteración en base a un parámetro denominado tasa de aprendizaje. El valor de la tasa es un asunto delicado, ya que si se elige un valor demasiado pequeño el sistema será incapaz de aprender a un ritmo adecuado, pero si le asigna un valor demasiado alto el proceso puede divergir y ser incapaz de encontrar los parámetros óptimos.

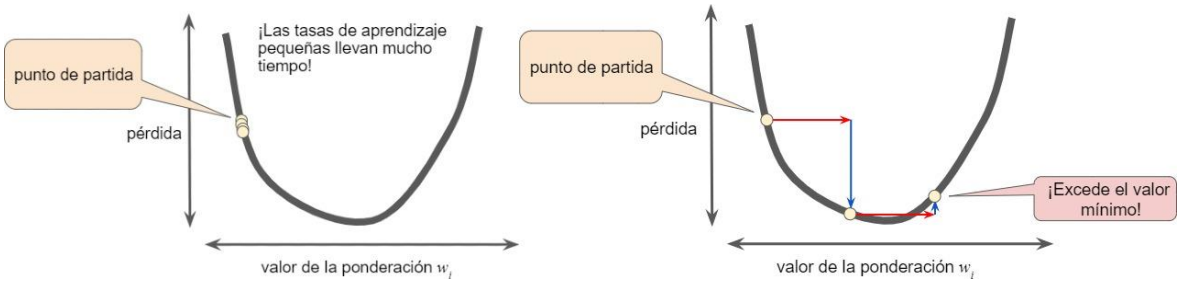


Figura 10. Gráficas comparativas de las consecuencias de escoger tasas de aprendizaje poco óptimas

La regresión lineal puede ser muy útil para realizar predicciones gracias a su sencilla implementación y eficiencia. Sin embargo, no resulta eficaz para modelos más complejos. Por otra parte, si se desea pronosticar si sucederá o no un suceso determinado, como ocurre en nuestro proyecto, se debe usar algoritmos de clasificación.

El equivalente de la regresión lineal en los algoritmos de clasificación es la regresión logística. Este tipo de regresión emula los resultados de una función sigmoidea para obtener una probabilidad entre 0 y 1 y poder así tomar una decisión en base a un umbral preestablecido. Los resultados de la regresión logística deben ser tratados con cautela. Un modelo con una tasa de error muy baja no implica necesariamente que se trate de un modelo eficaz. Por ejemplo, si un modelo está diseñado para predecir si un mensaje es spam o no spam e indica que todos los correos son no spam, es probable que obtenga una tasa de error pequeña, pero los resultados serán nefastos. Es por ello por lo que se definen tres conceptos para medir la calidad del sistema:

- Exactitud: Número de predicciones correctas entre el número total de predicciones
- Precisión: Proporción de identificaciones positivas que se realizaron correctamente
- Exhaustividad: Proporción de positivos reales se identificó correctamente

En función de qué es lo que se espera del sistema se busca potenciar unas características sobre otras, intentando mantener siempre un equilibrio. La fórmula matemática detrás de esta regresión es la siguiente, donde *y* indica la probabilidad y *z* es la función de la regresión lineal:

$$y = \frac{1}{1 + e^{-z}}$$

Otras técnicas más sofisticadas que ofrecen mejores resultados en modelos más complejos son el análisis de componentes principales y el análisis de componentes independientes.

El análisis de componentes principales (PCA) [11] aprovecha la correlación entre las variables de entrada y salida del sistema para utilizar aquellos parámetros que sean más útiles para el sistema y despreciar aquellos que ofrezcan menos información. Para ello, el sistema normaliza las variables y obtiene los autovectores y autovalores de la matriz de covarianza, quedándose con aquellos que sean mayores. Utilizando los autovectores seleccionados, se construye la matriz de proyección y se realiza la transformada con los valores iniciales obteniendo los pesos de las variables.

Por su parte, el análisis de componentes independientes (ICA) es similar al PCA, con la diferencia de que en este caso las componentes no están directamente relacionadas. Dado que los datos tienen una estructura lineal, la dificultad reside en estimar la matriz de mezcla. El caso más representativo del ICA es la separación ciega de fuentes [12]. Este problema aparece en muchos campos, como el procesado de señales de audio, antenas, geología o medicina. El problema consiste en separar varias fuentes superpuestas para poder obtener

las señales originales. Un ejemplo clásico es el denominado cocktail party [13], en el que se intenta obtener una única conversación de entre todas las conversaciones de una fiesta.

Los algoritmos explicados previamente son utilizados en problemas lineales, es decir, sirven para resolver problemas donde se puede representar la solución como una recta, pero no sirven en modelos no lineales. Para resolver este tipo de problemas se pueden utilizar redes neuronales.

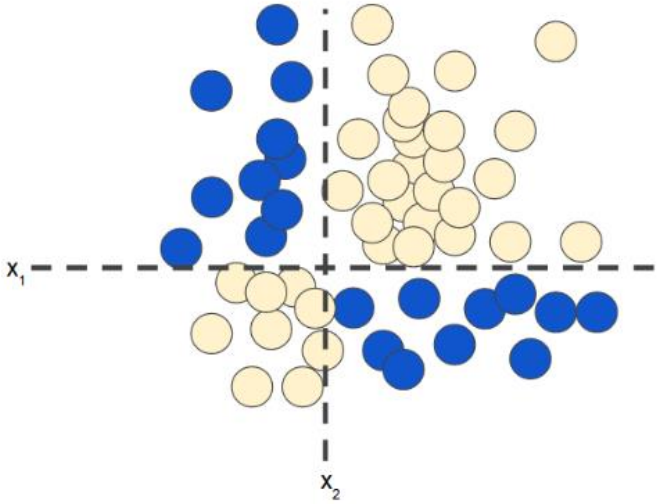


Figura 11. Ejemplo de problema no lineal

Las redes neuronales utilizan neuronas que aplican funciones no lineales a los parámetros de entrada, de forma que cada capa que le apliquemos a la red aprende una función más compleja y de nivel más alto que la anterior. Además, existen ponderaciones entre cada capa de forma que con cada iteración los pesos de cada nodo varían. Las funciones que se aplican se llaman funciones de activación y suelen variar entre una capa y otra, de forma que se aplican distintos cambios en cada capa. Existen gran variedad de funciones no lineales que se usan en la creación de redes neuronales. Sin embargo, dos de las más relevantes son la función sigmoidea, la cual ya se ha mencionado previamente, y la función de activación de unidad lineal rectificada (ReLU)

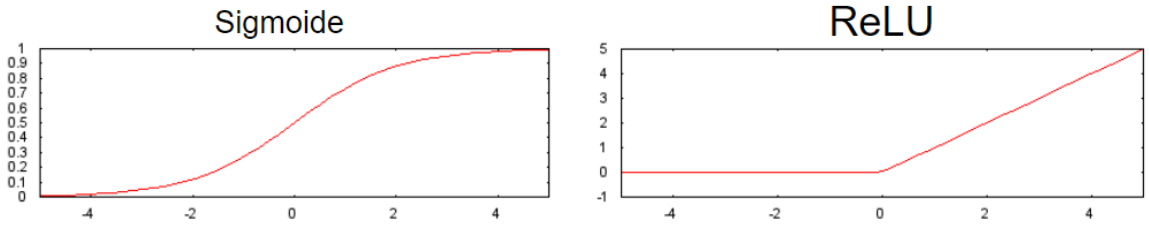


Figura 12. Funciones sigmoide y ReLU

Gracias a estas funciones se puede calcular el valor de cada nodo de la red siguiendo la siguiente formula, donde σ representa la función de activación *x* es el parámetro de entrada *w* es el peso de cada parámetro y *b* la ordenada en el origen.

$$valor = \sigma(wx + b)$$

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como ya se ha mencionado previamente, el objetivo de este proyecto consiste en realizar un análisis de los trayectos y los accidentes de tráfico con implicación de bicicletas para poder obtener un modelo de *machine learning* que sea capaz de predecir qué zonas son más peligrosas para los usuarios.

Este enfoque tan solo ha sido tratado en algunos estudios de universidades de Japón y China, pero nunca se ha orientado como una herramienta para el usuario en su día a día, sino más bien como una técnica para estudiar el comportamiento de los ciclistas [14]. Es por ello que se trata de un sector que no ha sido cubierto por ninguna de las aplicaciones o sistemas de navegación que se encuentran actualmente en el mercado.

Teniendo en cuenta las ventajas tecnológicas de las que se dispone en este momento, la gran cantidad de información con la que se cuenta tras la publicación de datos llevada a cabo por el ayuntamiento de Madrid, y todos los argumentos presentados en los párrafos anteriores, se puede afirmar que se trata de un proyecto revolucionario en el sector. Es una apuesta segura gracias a su innovación, al gran tamaño del grupo demográfico hacia el que está enfocado y al hecho de que cubre una necesidad real.

4.2 OBJETIVOS

El objetivo de este proyecto es el análisis de los datos obtenidos mediante procesos de ingesta y el desarrollo de un sistema de predicción basado en machine learning. Dicho sistema de predicción ofrecerá a los usuarios la ruta más segura entre dos puntos en base a los datos ya mencionados anteriormente y a la información proporcionada por el propio usuario.

De esta forma, los objetivos de este proyecto quedan marcados de la siguiente manera:

4.2.1 PROCESADO DE LOS DATOS

Para poder explotar al máximo los recursos de los que se dispone, es necesario en primer lugar familiarizarse con la fuente principal de datos sobre la que se basa el proyecto: Los datos de trayectos de Bicimad, los ficheros de accidentes disponibles de la comunidad de Madrid, los históricos meteorológicos y de polución.

Tras la primera aproximación se considerará qué datos son más relevantes para el proyecto y cuales se pueden obviar ya que no ofrecen información relevante. Una vez que se haya terminado el filtrado de datos, estos se deben ajustar para poder ser utilizados posteriormente en el modelo de machine learning.

4.2.2 PROGRAMA DE TRAZADO DE RUTAS

Como parámetros de entrada del modelo de predicción se utilizará información sobre diversos campos además de las coordenadas de la ruta. Para obtener dichas coordenadas se desarrollará un programa que calcule la ruta más optima basándose en la distancia entre los puntos.

4.2.3 DESARROLLO DE UN MODELO DE MACHINE LEARNING PARA PREDECIR ACCIDENTES

Se realizará un estudio sobre los diversos algoritmos y modelos de machine learning disponibles para decidir cual se adapta mejor al sistema. Una vez que se haya seleccionado la opción que más convenga al sistema, se implementará el algoritmo elegido utilizando la librería especializada TensoFlow. Tras su

implementación se medirá la calidad del sistema y se realizarán las modificaciones necesarias.

Una vez que se termine de depurar el modelo, se realizará una comparativa con los resultados obtenidos al utilizar otros tipos de algoritmos y se analizará la diferencia entre los resultados obtenidos.

4.2.4 PROGRAMA DE TRAZADO DE RUTAS ALTERNATIVAS

Partiendo de la lista de coordenadas potencialmente peligrosas que son devueltas por el modelo de machine learning, se desarrollará un nuevo programa de trazado de rutas que se apoye en el generado en el punto 4.2.2. Este nuevo programa será capaz de calcular la ruta más corta a la vez que evita las zonas indicadas por el predictor.

4.3 METODOLOGÍA

Los primeros meses del proyecto se utilizaron para familiarizarse con el trabajo en un entorno Big Data. Primero de todo se realizó un análisis estadístico en pequeña escala de las variables principales (el histórico de los trayectos de bicicletas de Bicimad) utilizando R. Este análisis se llevó posteriormente a gran escala integrando todos los datos en el *cluster* y, mediante funciones de Python, se extrajeron conclusiones sobre el conjunto total de los datos.

El siguiente paso consistió en realizar un estudio sobre el comportamiento de los datos facilitados para poder decidir qué modelo y algoritmo es el más adecuado para nuestro sistema. Para ello se realizó una comparativa entre las funciones disponibles dentro de la librería de TensorFlow considerando los pros y contras de cada opción. Es importante destacar lo crítico de este paso, ya que sirvió como punto de partida para el sistema de predicción.

A continuación, se realizó el programa de trazado de rutas capaz de ofrecer el camino más corto entre dos puntos. Para poder desarrollar este programa lo primero fue obtener un mapeado de la ciudad de Madrid usando la librería OpenStreetMap. Gracias a la información ofrecida por dicha librería, se pudo proceder a desarrollar el programa de trazado en base a la longitud de las calles. Es relevante mencionar que los datos de salida de este programa son usados más adelante como parámetros de entrada para el modelo de machine learning.

Una vez que se desarrollaron los pasos anteriores, se procedió a la programación del modelo de machine learning utilizando los datos aportados por mis compañeros en sus respectivos TFGs, así como la librería de Google TensorFlow. Tras obtener un modelo funcional, se realizaron las pruebas necesarias para pulir el sistema y mejorar los resultados.

Finalmente se generó una variante del programa de trazado de rutas para obtener caminos que eviten las zonas indicadas por el modelo.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

4.4.1 PLANIFICACIÓN

La planificación del proyecto se estableció según se representa en la siguiente tabla. A lo largo del desarrollo del sistema se fue ajustando el calendario mediante reuniones periódicas con el director del proyecto. Dichas reuniones servían para realizar correcciones y establecer unos objetivos de tiempo coherentes con la dificultad de las metas propuestas.

	OCTUBRE	NOVIEMBRE	DICIEMBRE	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO
Familiarización con el modelo de datos									
Análisis a pequeña escala de las variables principales									
Análisis a gran escala de las variables principales									
Estudio de los posibles modelos									
Programa de trazado de rutas									
Optimización de los datos									
Desarrollo del modelo de predicción									
Depuración del modelo									
Programa de trazado de rutas alternativas									

Tabla 1. Planificación del proyecto

4.4.2 ESTIMACIÓN ECONÓMICA

A la hora de calcular el coste del proyecto, se debe tener en cuenta que forma parte de un desarrollo mayor en el que se encuentran implicadas terceras personas. Es por ello, que se debe considerar, no solo el coste de los recursos y de la mano de obra de este módulo del sistema, sino también de todos los implicados.

Recursos utilizados	**€/Ud**	**Uds**	**Total**
Cluster ICAI	80000	1	80000
Xiaomi mi Air 13,3''	899	1	899
Dell Inspiron 13 7000 Series	761	1	761
Lenovo Y50-70	965	1	965
		Total	82625

Tabla 2. Coste económico de los recursos empleados

Modulos del proyecto	Horas	€/Hora	Total
Análisis de los trayectos	300	5	1500
Programación de la ingesta de datos multifuente	300	8,25	2475
Análisis de los datos y programa de predicción	350	9	3150
Visualización de datos	300	8,5	2550
		Total	9675

Tabla 3. Coste económico de la mano de obra

Teniendo en cuenta la mano de obra y los recursos utilizados (electricidad, emplazamiento, dietas, etc), podemos suponer que la estimación económica global asciende a unos 93.000€.

Capítulo 5. SISTEMA DESARROLLADO

Como ya se ha explicado anteriormente, el sistema desarrollado cuenta con dos apartados muy diferenciados, el programa de trazado de rutas y el programa de predicción de accidentes. Además, como podemos ver en la siguiente imagen, también se ha implementado un módulo de visualización preliminar que permite realizar una representación muy simple de los resultados obtenidos.

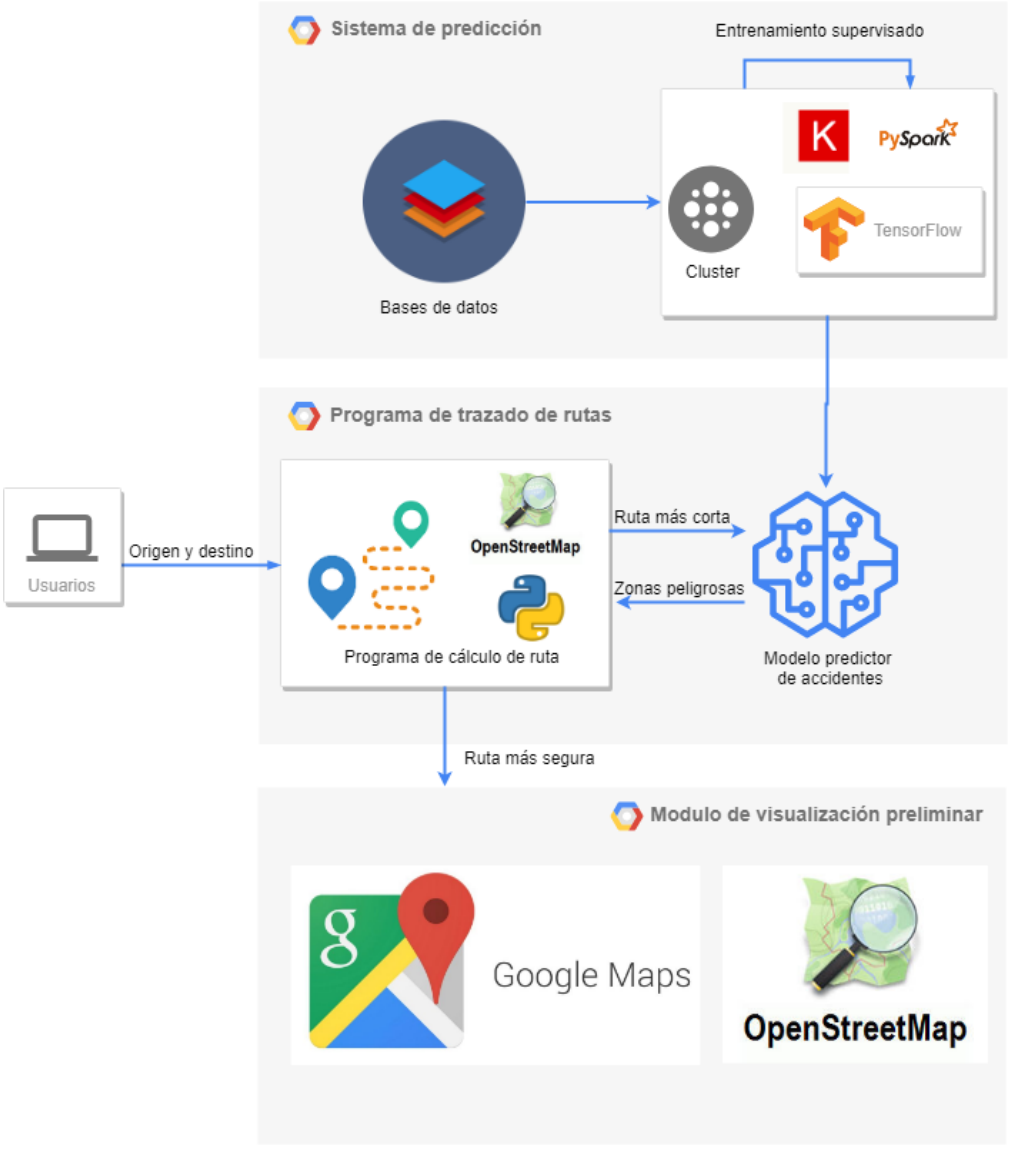


Figura 13. Diagrama de la arquitectura del sistema

El programa en su conjunto está compuesto por 6 notebooks, los cuales están implementados dentro de un último Jupyter que sigue el diagrama mostrado a continuación.

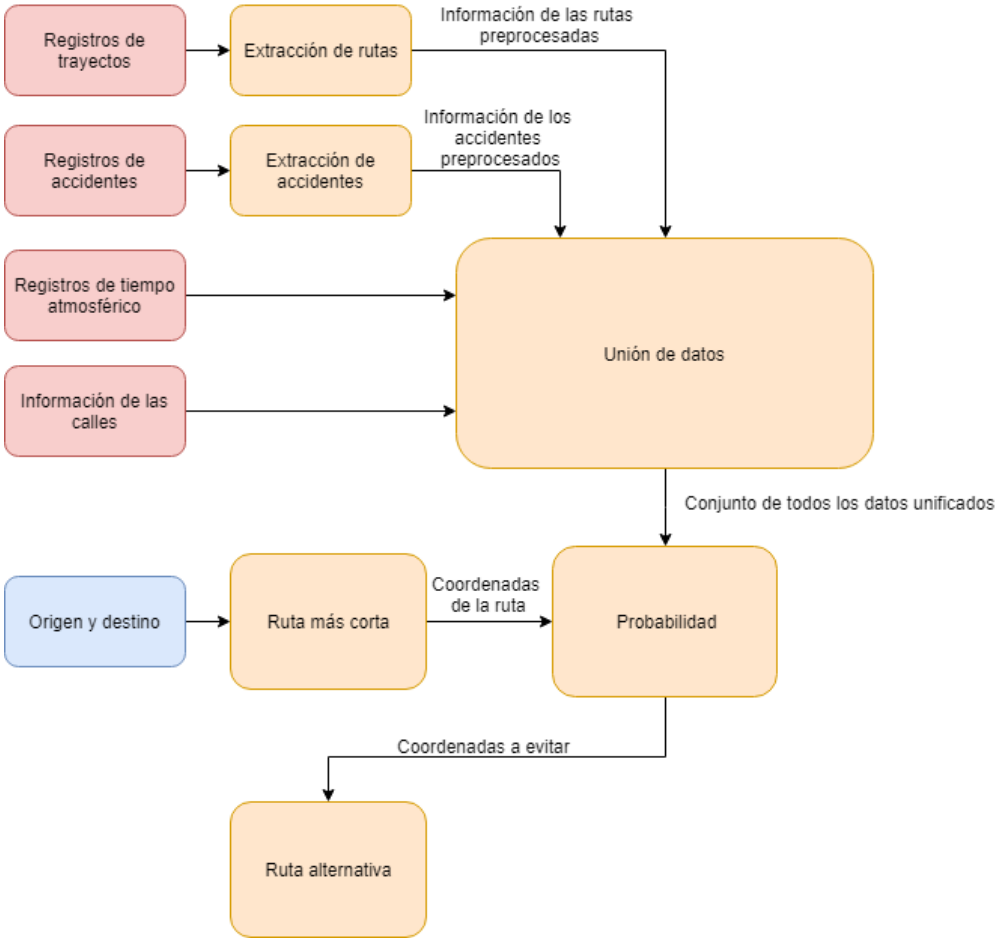


Figura 14. Diagrama de los notebooks desarrollados

Los recuadros rojos representan los datos almacenado en el *cluster* y que no han sido modelados para ser usados por el sistema, el recuadro azul son los datos a introducir por el usuario y los naranjas son los notebooks mencionados.

5.1 ESTUDIO PREVIO DE LAS VARIABLES PRINCIPALES

El eje central de este proyecto es el sistema de predicción basado en machine learning y, como se ha mencionado previamente, es necesario analizar los datos de

los que se dispone, así como los resultados que se espera obtener, para poder seleccionar el algoritmo que mejor se adapte al modelo en cuestión. Para ello, se realizó un estudio estadístico a pequeña y gran escala de los datos de los que se disponía.

Los datos de los que se dispone y sobre los que se basa el estudio, son principalmente, los aportados por el ayuntamiento de Madrid. Sin embargo, también se ha utilizado información recopilada mediante técnicas de webscrapping en un TFG previo.

La información relativa a los trayectos de las bicicletas de la iniciativa BiciMad contiene, entre otros, los siguientes campos:

- Fecha
- Hora
- Origen y destino
- Edad del usuario
- Tipo de usuario (Habitual o no)
- Tiempo de viaje

Los demás campos asociados han sido descartados debido a la imposibilidad de relacionarlos de forma funcional con el modelo o debido a que no existe un parámetro común en los registros de accidentes.

A continuación, se presentan los datos referentes a los registros de accidentes, los registros de polución y los registros de tiempo atmosférico.

	FECHA	RANGO HORARIO	CPFA Granizo	CPFA Hielo	CPFA Lluvia	CPFA Niebla	CPFA Seco	CPFA Nieve	TIPO PERSONA	SEXO	Tramo Edad	LATITUD	LONGITUD
0	07/01/2011	DE 15:00 A 15:59	NO	NO	NO	NO	SI	NO	CONDUCTOR	HOMBRE	DE 40 A 44 A�OS	40.44939722222222	-3.602327777777778
1	12/01/2011	DE 18:00 A 18:59	NO	NO	NO	NO	SI	NO	CONDUCTOR	HOMBRE	DE 25 A 29 A�OS	40.45194722222222	-3.683591666666667
2	13/01/2011	DE 9:00 A 9:59	NO	NO	NO	NO	SI	NO	CONDUCTOR	HOMBRE	DE 25 A 29 A�OS	40.44925555555555	-3.717102777777778
3	15/01/2011	DE 10:00 A 10:59	NO	NO	NO	NO	SI	NO	CONDUCTOR	HOMBRE	DE 65 A 69 A�OS	None	None
4	15/01/2011	DE 12:00 A 12:59	NO	NO	NO	NO	SI	NO	CONDUCTOR	HOMBRE	DE 30 A 34 ANOS	None	None

Tabla 4. Ejemplo de los registros de accidentes

MUNICIPIO	ESTACION	MAGNITUD	TECNICA	PERIODO-ANALISIS	ANHO	MES	DIA	00	...	14	15	16	17	18	19	20	21	22	23
079	004	01	38	02	2019	04	10	00003	...	00004	00003	00000	00000	00000	00000	00000	00000	00000	00000
079	004	06	48	02	2019	04	10	000.3	...	000.4	000.4	00000	00000	00000	00000	00000	00000	00000	00000
079	004	07	08	02	2019	04	10	00002	...	00012	00012	00000	00000	00000	00000	00000	00000	00000	00000
079	004	08	08	02	2019	04	10	00015	...	00032	00034	00000	00000	00000	00000	00000	00000	00000	00000
079	004	12	08	02	2019	04	10	00017	...	00050	00053	00000	00000	00000	00000	00000	00000	00000	00000
079	008	01	38	02	2019	04	10	00008	...	00009	00010	00000	00000	00000	00000	00000	00000	00000	00000
079	008	06	48	02	2019	04	10	000.1	...	000.2	000.1	00000	00000	00000	00000	00000	00000	00000	00000
079	008	07	08	02	2019	04	10	00004	...	00011	00008	00000	00000	00000	00000	00000	00000	00000	00000
079	008	08	08	02	2019	04	10	00039	...	00047	00035	00000	00000	00000	00000	00000	00000	00000	00000

Tabla 5. Ejemplo de los registros de polución

	FECHA	PRECIPITACIONES-ANALISIS	Tavg	VIENTOdir	VIENTOvel
0	2015/07/02	0.0	29.5	S	14.2
1	2015/07/03	0.0	28.6	SSE	12.7
2	2015/07/04	0.0	30.3	SSE	14.2
3	2015/07/05	0.0	30.5	SSW	8.4
4	2015/07/06	0.0	30.8	WSW	10.1

Tabla 6. Ejemplo de los registros de tiempo atmosférico

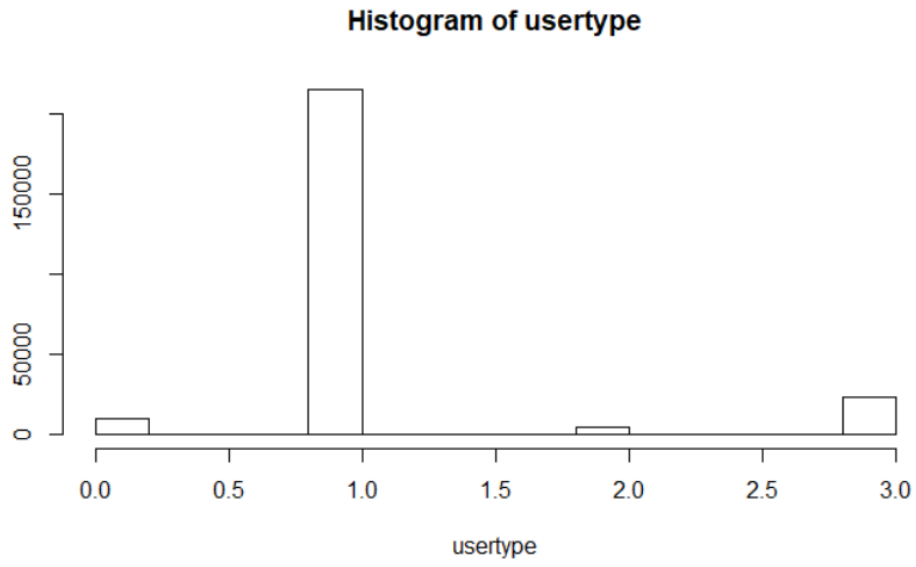
5.1.1 ESTUDIO A PEQUEÑA ESCALA

Este primer estudio se llevó a cabo utilizando el lenguaje de programación R, el cual es un software abierto para computación estadística. El objetivo del análisis era obtener pruebas que identificasen un patrón en el comportamiento de los usuarios

y que permitiesen realizar un primer filtrado de los datos para eliminar potenciales fuentes de ruido.

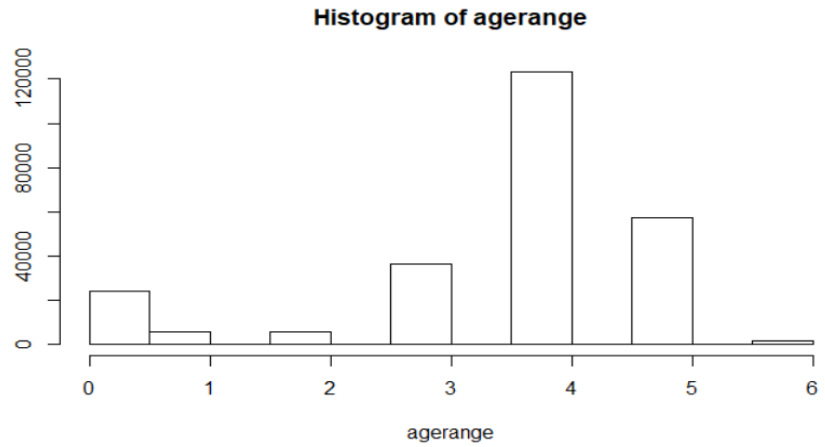
Los datos utilizados fueron los referentes al mes de abril de 2017 y los campos analizados se centraron en los tipos de usuario, la edad de los usuarios y el tiempo de viaje.

Como se puede apreciar en la siguiente gráfica, la mayoría de los usuarios cuentan con un bono anual de BiciMad (tipo 1) y son considerados como ciclistas habituales y experimentados. Sin embargo, también existen trayectos asociados a los empleados encargados del mantenimiento de las bicicletas (tipo 3) y, por tanto, es necesario filtrar estos trayectos ya que no aportan información útil sobre los viajes.



Gráfica 1. Histograma de los tipos de usuario del mes de abril de 2017

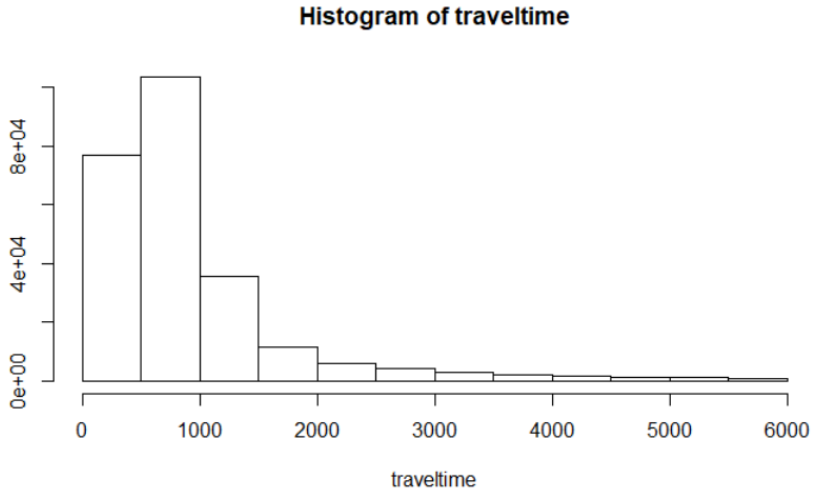
La edad también se consideró como fuente potencialmente útil para el proyecto, ya que se puede suponer que los usuarios de edad más avanzada o aquellos que sean excesivamente jóvenes, son más propensos a sufrir un accidente de tráfico.



Gráfica 2. Histograma de la edad de los usuarios en mes de abril de 2017

Se puede apreciar claramente como la edad de los usuarios sigue una función gaussiana de media 4, lo que significa que la mayoría de los usuarios son personas de edad comprendida entre 27 y los 40 años y el uso de las bicicletas se reduce según los usuarios se alejan de esta franja. Es destacable mencionar la cantidad de usuarios cuya edad no ha sido registrada (valor 0 en la gráfica 2), lo que lleva a pensar que puede existir una cantidad nada despreciable de trayectos corruptos dentro de los registros del ayuntamiento.

Para corroborar la hipótesis previamente presentada, se analizó la duración de los trayectos con el fin de obtener alguna señal que indicase la existencia real de dichos datos corruptos.



Gráfica 3. Histograma de la duración de los trayectos en el mes de abril de 2017

Los resultados representados en la gráfica 3 indican dos hechos de gran relevancia para el proyecto. En primer lugar, la existencia de trayectos de duración superior a los 3000 segundos y, en segundo lugar, el gran número de registros en la que la duración del viaje es inferior a 20 segundos. Ambos descubrimientos detectan *outlayers* que deben ser filtrados, ya que no reflejan datos reales del sistema. Es más, ambos casos pueden ser asociados a situaciones relacionadas con vandalismo. La duración excesiva de los trayectos puede estar relacionada con los hurtos de bicicletas. Esta idea esta poyada por el hecho de que se han encontrado señales GPS fuera de la zona asignada para la utilización de las bicicletas.

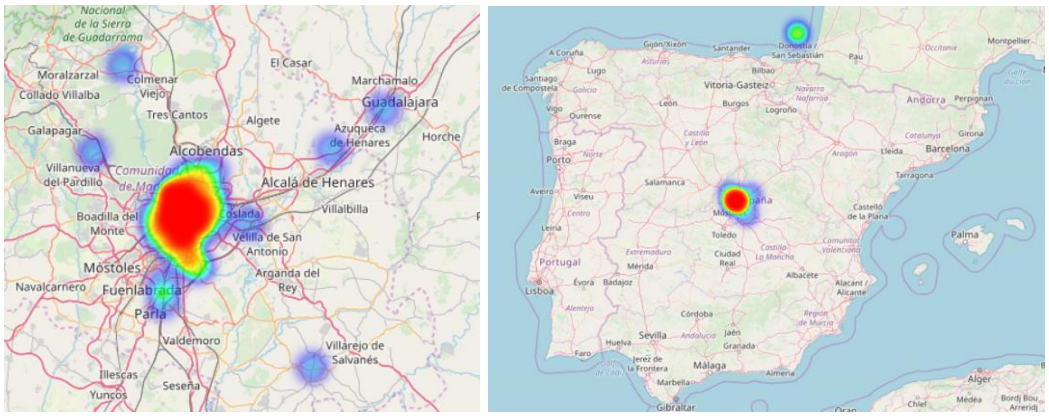


Figura 15. Mapa de calor de las señales GPS enviadas por las bicicletas

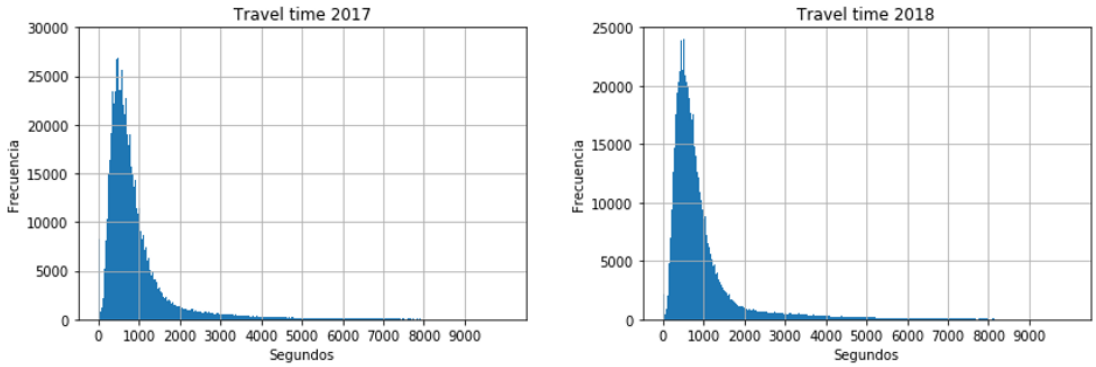
Por otro lado, los trayectos en los que la duración del viaje es prácticamente nula se han relacionado con el intento de arrancar las bicicletas de sus estaciones de manera ilegal.

5.1.2 ESTUDIO A GRAN ESCALA

Tras considerar los resultados obtenidos en el estudio anterior, se precedió a realizar un análisis en el que se incluía la información relativa a todos los trayectos registrados desde el año 2017 hasta agosto del 2018. Para poder estudiar semejante cantidad de datos, fue necesario trasladar los registros al *cluster* de la universidad,

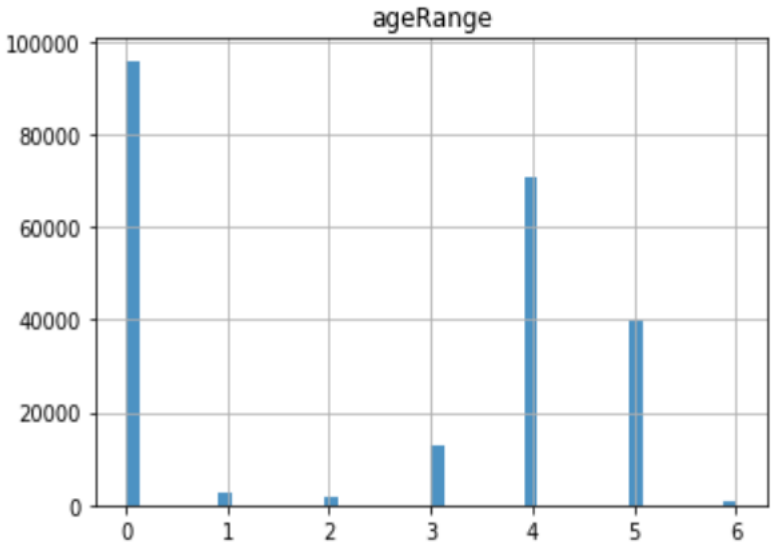
donde se utilizó el lenguaje de programación Python con su librería de Spark. Aprovechando la capacidad de procesamiento que aporta el *cluster*, se comprobó que las conclusiones extraídas durante el estudio a pequeña escala seguían siendo válidas.

Teniendo en cuenta que la conclusión más relevante se encontraba asociada con el tiempo de trayecto, este fue el primer campo que se analizó. Como demuestran las siguientes gráficas, se mantiene el mismo comportamiento y existe una gran cantidad de datos a filtrar.



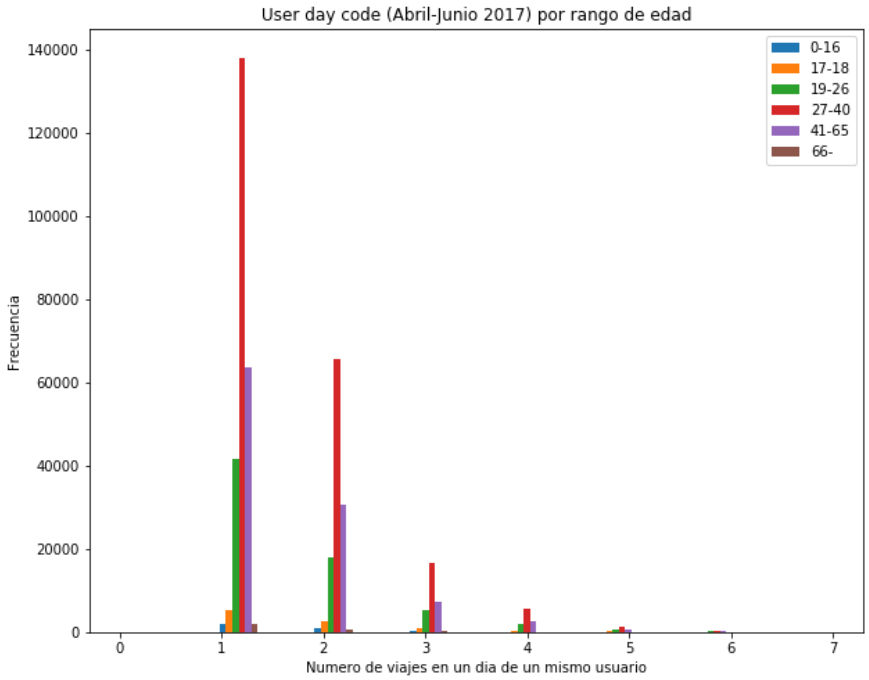
Gráfica 4. Duración de los trayectos a lo largo de los años 2017 y 2018

En cuanto a la edad de los usuarios, se mantiene la misma estructura que en el análisis a pequeña escala. Sin embargo, se puede reconocer un auge en el número de trayectos en el que no se identifica el rango de edad de los ciclistas. Este hecho es un síntoma de que uno o varios de los registros del ayuntamiento se encuentran corruptos y, por tanto, deben ser tratados con cuidado a la hora de procesarlos. Otro posible motivo de que no haya definido la edad de los usuarios es que se hayan arrancado las bicicletas de sus estaciones. Sin embargo, este caso es menos plausible, ya que es difícil de creer que exista tal número de hurtos.



Gráfica 5. Histograma de la edad de los usuarios entre los años 2017 y 2018

Otra forma de comprobar la habilidad de los ciclistas es comprobar cuantas veces al día cogen la bicicleta. Para obtener este valor se analizó el *user day code*, el cual es un valor que se le asigna cada día a cada usuario. Al comprobar las veces que se repite este código a lo largo de los registros se puede obtener el número de trayectos que se realizaron cada día, y al compararlo con la edad de los usuarios podemos establecer una conexión entre la habilidad y la edad media de los ciclistas.



Gráfica 6. Histograma del número de viajes por día en relación a la edad de los usuarios

Como era de esperar, la mayoría de los usuarios realizan únicamente un trayecto a lo largo del día y la mayoría de ellos tienen edades que se encuentran comprendidas entre los 27 y los 40 años. Sin embargo, es destacable como el siguiente sector que más trayectos por día realiza es el comprendido entre los 41 y los 65 años, cuando cabría pensar que los más jóvenes serían los más inclinados a realizar mayor número de viajes. Este hecho indica que el que los usuarios estén en mejor forma física no implica necesariamente que tengan una mayor destreza y, por tanto, se le debe dar más importancia a la experiencia de los ciclistas que a la fuerza o resistencia de estos.

Al contar con la capacidad de procesamiento del *cluster*, también es posible realizar un estudio sobre la información relativa a los accidentes de tráfico con implicación de bicicletas, lo cual no era posible durante el estudio a pequeña escala debido a que habría requerido un tiempo de computación excesivo en un ordenador normal.

La información relativa a los accidentes es reducida debido a que no se cuenta con los mismos datos que en el caso de los trayectos. Campos como la velocidad del ciclista en el momento del accidente o a donde se dirigía son datos de los que se dispone en los registros de los trayectos, pero no en los de los accidentes. Es por ello, que es necesario realizar un análisis más detallado sobre la ubicación del accidente con el fin de obtener información adicional. Para lograr este objetivo es necesario utilizar la librería de Open Street Map [15] para realizar un mapeado de la ciudad de Madrid. En base a dicho mapa, se puede obtener información sobre el tipo de vía, el número de calles que convergen en un cruce o la velocidad máxima permitida en cada zona.

A continuación, se muestran algunos de los campos que nos ofrece la librería OSMNX sobre las calles de la ciudad:

```
edges_polygon.columns
```

```
Index(['access', 'area', 'bridge', 'est_width', 'geometry', 'highway',
       'junction', 'key', 'landuse', 'lanes', 'length', 'maxspeed', 'name',
       'oneway', 'osmid', 'ref', 'service', 'tunnel', 'u', 'v', 'width'],
      dtype='object')
```

Figura 16. Campos ofrecidos por la librería OSMNX sobre las calles de Madrid

De entre los atributos devueltos por la librería es necesario destacar aquellos que son más útiles para el modelo: La anchura de la calle, el tipo de vía, longitud de la vía, velocidad máxima y conexión con otras calles.

Partiendo de la información mencionada en el párrafo anterior se ha diseñado la siguiente figura, la cual representa la zona central de la ciudad de Madrid. Las calles en color azul reflejan las calles de doble sentido, las rojas las de sentido único y las verdes son aquellas que cuentan con carriles bici oficiales. Por su parte, los puntos negros señalan ubicaciones donde se ha producido por lo menos un accidente.

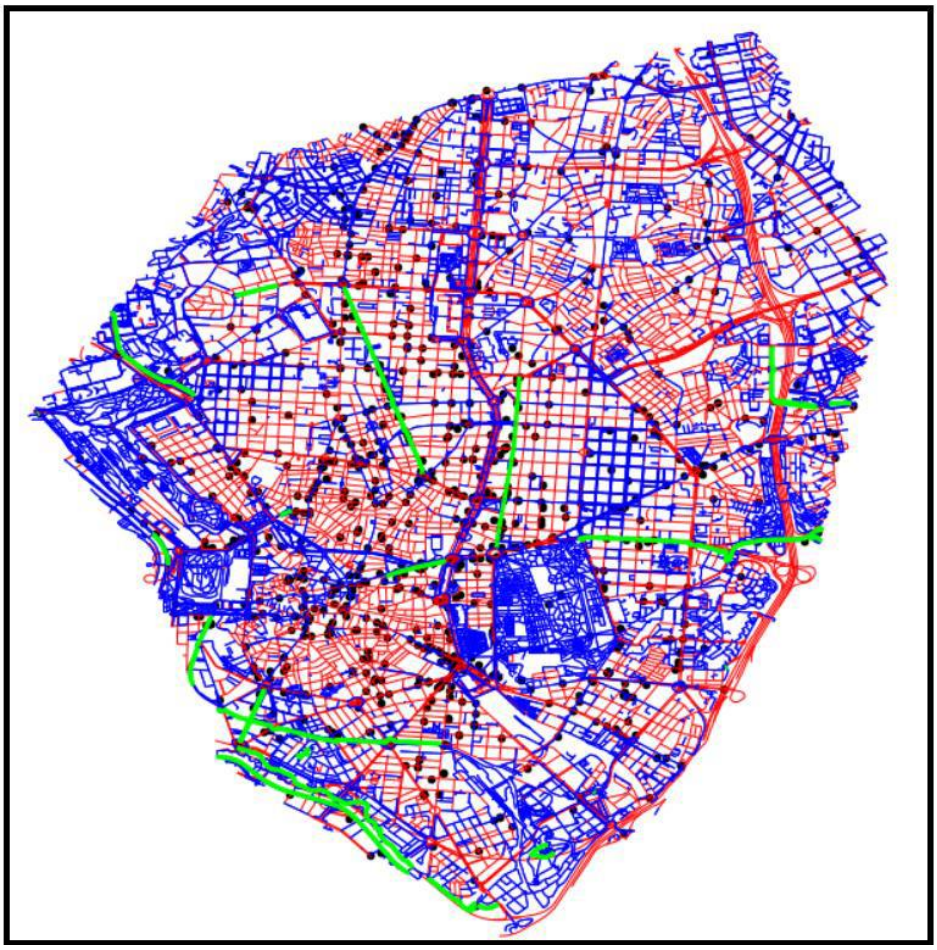


Figura 17. Mapeado de la ciudad de Madrid con accidentes y tipo de vía

Es destacable mencionar el hecho de que el número de accidentes no parece reducirse en las zonas donde hay carriles bici y tampoco existe una diferencia significativa entre la concentración en las rutas de sentido único y doble sentido.

5.1.3 CONCLUSIONES DEL ESTUDIO

Al finalizar el análisis de la fuente principal de datos de este proyecto se pueden extraer las siguientes conclusiones:

- Es necesario realizar un filtrado exhaustivo de los datos para eliminar los datos corruptos y aquellos *outlayers* que pueda perjudicar al modelo.

- Es necesario delimitar la zona geográfica sobre la que se va a trabajar para poder eliminar la información asociada a fallo del GPS o hurtos.

- La unión de varios campos ofrece información más relevante que los datos aportados por cada variable de forma independiente.

- La variable de la edad de los usuarios debe compararse con el número de trayectos por día para poder obtener un parámetro objetivo sobre las capacidades del usuario.

- Los accidentes se concentran en zonas con mayor convergencia de calles

- No existe una diferencia apreciable entre la concentración de accidentes en vías de sentido único con respecto a aquellas calles de doble sentido.

5.2 PROGRAMA DE TRAZADO DE RUTAS

5.2.1 CÁLCULO DE LA RUTA MÁS CORTA

El programa de trazado de la ruta más corta sirve para ofrecer al usuario el camino más rápido entre dos puntos, pero también se utiliza para obtener información adicional sobre los trayectos registrados. Las bicicletas de BiciMad envían un pulso GPS indicando su posición cada 60 segundos, por lo que los registros del ayuntamiento cuentan únicamente con una coordenada por minuto de trayecto, este hecho implica que existe una gran cantidad de datos que no son registrados.

Para solventar este problema se establece la hipótesis de que los usuarios utilizan siempre el camino más corto y, por tanto, se puede emplear el programa de trazado de rutas para rellenar los huecos que faltan.

Con el fin de desarrollar el programa de navegación se ha utilizado el mapa representado en el apartado 5.1.2 y la librería Open Street Map. Utilizando la información relativa a las calles, se ha implementado el algoritmo de Dijkstra usando la distancia de las calles como pesos. De esta forma, el programa utiliza las calles como si fuesen los ejes de un grafo y los cruces como nodos.

A la hora de diseñar el programa se programó de forma que la ruta calculada siguiese el sentido de las calles. Sin embargo, como ya se ha mencionado previamente, el programa también sirve para ofrecer información sobre los trayectos registrados y al comparar la distancia de los caminos que devolvía el programa con el tiempo de los trayectos, se detectaron gran número de incongruencias. Existía un número alarmantemente elevado de trayectos en los que, según el programa desarrollado, se había recorrido una distancia de varios cientos de metros en tan solo unos pocos segundos. Este hecho se debía a que gran cantidad de ciclistas no siguen las reglas de circulación o directamente circulan por la acera. Por este motivo, fue necesario modificar el sistema de forma que calculase la ruta más corta para los peatones.

Como se puede apreciar en la siguiente imagen, la longitud de las rutas se redujo considerablemente y la complejidad del grafo aumentó en gran medida. Este hecho se debe al aumento en el número de caminos por los que se puede circular (parques, pasos subterráneos, etc) lo que se traduce en un aumento en el número de ejes del grafo.

Figura 18. Ruta más corta entre dos puntos respetando el sentido de las calles

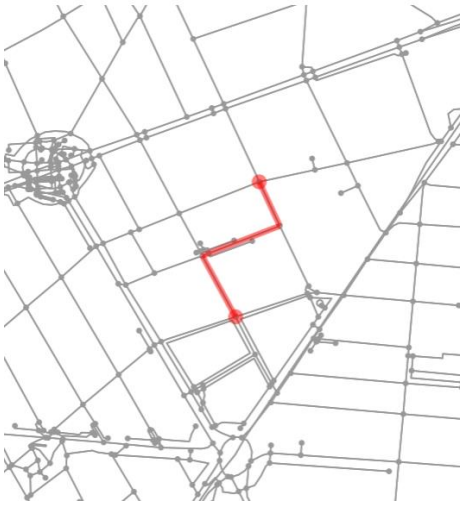


Figura 19. Ruta más corta entre dos puntos sin respetar el sentido de las calles

La versión final del programa recibe las coordenadas de origen y destino y, en base a ellas, devuelve el camino más corto para los peatones. En la siguiente figura se puede ver el grafo de la zona con los nodos representados como círculos y la ruta calculada en rojo.

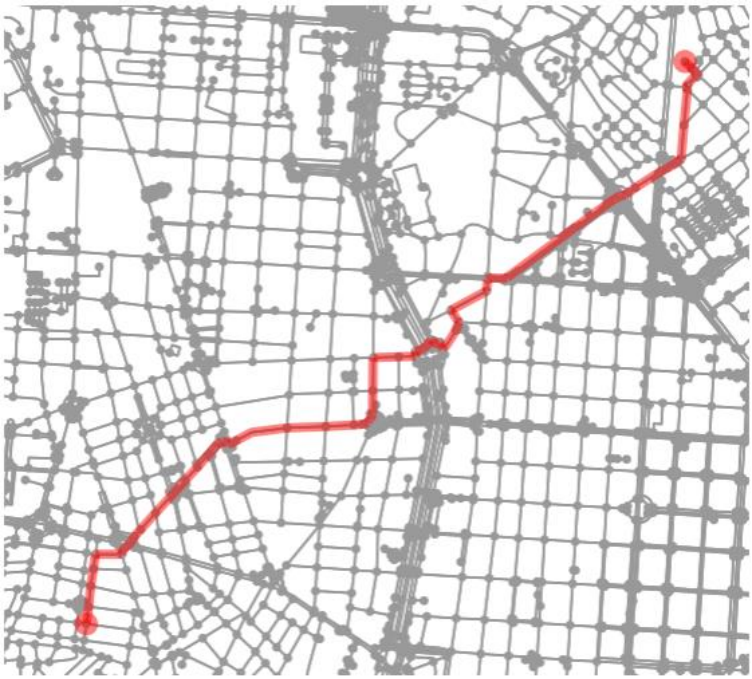


Figura 20. Ruta calculada por el programa de trazado de rutas. Del Auditorio Nacional de Música a Calle San Andres 10

5.2.2 CÁLCULO DE RUTAS ALTERNATIVAS

Partiendo del programa descrito en el apartado anterior es necesario desarrollar una variante que sea capaz de evitar zonas concretas indicándolas como parámetros de entrada. Las funciones implementadas para lograr alcanzar este objetivo son las siguientes:

- **get_nodes_in_area(G, N, d= x):** Detecta y devuelve los identificadores de los nodos del grafo G que se encuentren a una distancia d del nodo N
- **ruta_alternativa(G, o, d, fn):** Toma el grafo G y calcula la ruta desde o hasta d evitando las zonas indicadas en la lista fn.

La lógica existente dentro de la función **ruta_alternativa** consiste en identificar los nodos cercanos a las zonas a evitar mediante el uso de **get_nodes_in_area**. Una vez que han sido localizados todos los nodos implicados, estos se eliminan del grafo para después aplicar la versión original del programa.

En la Figura 23 se ha calculado la misma ruta que en la Figura 22, con la excepción de que en este caso se ha indicado que los nodos marcados en color verde, y sus alrededores, deben ser evitados.

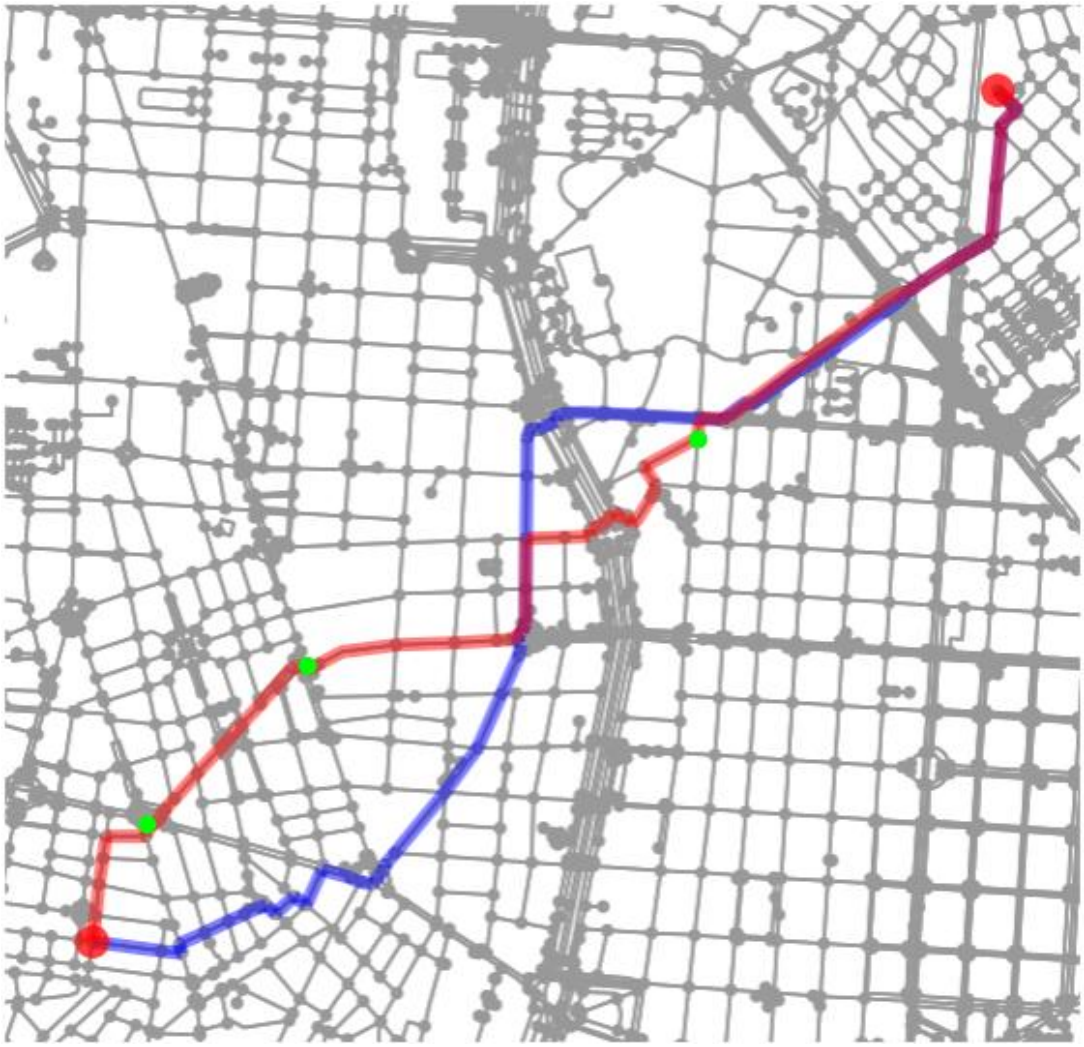


Figura 21. Comparativa de una ruta rápida con una alternativa

Durante el desarrollo del programa se identificaron dos potenciales fuentes de error. La primera de ellas se da cuando se intenta seleccionar una zona en la que no existe ningún nodo aparte del originario ya que el programa devuelve un error y se pierde todo el progreso alcanzado. Para solventar este fallo se implementó un control de excepciones que solucionó el problema y no tuvo mayores consecuencias. Por otro lado, el programa se diseñó de forma que en ningún caso los nodos de origen y destino puedan quedarse aislados, es decir, siempre existe, como mínimo, un nodo conectado al punto inicial y final. Sin embargo, en el caso de que existan demasiadas zonas peligrosas al principio o al final del trayecto, el grafo se puede dividir dejando

aislados un grupo de nodos e imposibilitando que se encuentre una ruta entre origen y destino.

Figura 22. Ejemplo de grafo fragmentado

Debido a la falta de tiempo y a la complejidad de modificar y generar programas de control para este fallo, se decidió que lo mejor para el sistema era solucionar este problema en versiones futuras del modelo.

5.3 PROGRAMA DE PREDICCIÓN DE ACCIDENTES

5.3.1 MODELADO DEL SISTEMA

A la hora de empezar a diseñar el modelo de machine learning se tuvo que hacer frente al hecho de que no se contaba con la misma información sobre los trayectos que sobre los accidentes. Mientras que la información de los trayectos de BiciMad es muy detallada, los registros del ayuntamiento sobre accidentes son bastante escuetos.

Este hecho provocó que se buscase un modelo que unificase los dos tipos de datos en uno solo, lo que conllevó una inevitable pérdida de información. El enfoque que finalmente se utilizó fue utilizar los nodos del grafo como receptores de eventos. Cada vez que se detecta un accidente o el paso de un ciclista cerca de la ubicación de un nodo, se registran en un *dataframe las* condiciones del usuario y de la vía, así como el tipo de evento (1 para accidentes 0 para ciclistas normales) en el momento determinado.

El modelo de datos final contiene los siguientes campos:

- Latitud
- Longitud
- Identificador del nodo
- Edad del usuario
- Tipo de usuario
- Nivel de precipitaciones
- Temperatura
- Velocidad del viento
- Número de ejes conectados al nodo
- Tipo de evento

En un primer momento se decidió usar las funciones de OSMNX para identificar que nodos se encontraban más cerca de cada ubicación. Sin embargo, esta estrategia tenía un elevado coste de tiempo y recursos que imposibilitaba su implementación, por lo que se definieron funciones propias que realizaban este trabajo explotando toda la potencia del *cluster* y ahorrando mucho tiempo de procesamiento.

El motivo por el cual las funciones definidas en la librería OSMNX consumían tantos recursos era que utilizaban Dataframes de Pandas, lo que ralentizaba el proceso y no utilizaba las funciones *map* de Spark. En cambio, el programa desarrollado para calcular los nodos utilizaba únicamente Dataframes de Spark y la función *map* para paralelizar los procesos. La lógica del programa consistía en comparar cada coordenada de cada ruta con las coordenadas de cada nodo del grafo. A continuación, se calculaba las distancias entre todas las rutas y se escogía aquella que era menor, utilizando en todo momento técnicas de procesamiento distribuido.

Es relevante mencionar que, más adelante, se añadirán nuevas variables al modelo las cuales surgen de la unión de campos. Algunos ejemplos de estas variables son la unión de la latitud y la longitud para identificar localizaciones como cuadrantes o el uso de la edad y el tipo de usuario para analizar la destreza de los ciclistas.

5.3.2 SELECCIÓN DEL ALGORITMO

Como ya se mencionó en apartados anteriores, es necesario decidir que algoritmo se adapta mejor al modelo de trabajo. Con este fin, se realizó una comparativa entre distintas técnicas y algoritmos para escoger aquel capaz de ofrecer mejores resultados. Como es lógico, los primeros algoritmos analizados fueron más simples y según se iba avanzando en el estudio también fue aumentando la complejidad de los mismos.

En primer lugar, se investigó la regresión lineal ya explicada en el punto 3. Esta técnica es bastante útil pese a su simpleza y puede ofrecer buenos resultados para contextos

poco complejos. Por desgracia, el caso de uso de los accidentes de tráfico es excesivamente intrincado para este modelo y tuvo que ser descartado rápidamente.

El siguiente algoritmo a analizar fue el del análisis de las componentes principales. Esta técnica utiliza las n primeras variables que más información aportan al modelo y desprecia aquellas que sobran. La mayor ventaja que ofrece este sistema es que elimina aquellas variables que pueden introducir ruido en el modelo y solo deja la información útil, por otro lado, se corre el riesgo de eliminar fuentes de información y obtener peores resultados. Por ello, es necesario analizar con cuidado las variables que se van a utilizar.

La última técnica que se analizó y la que se escogió como sistema a implementar fue el análisis de las componentes independientes, también explicada en el punto 3. El motivo que propició la selección de esta técnica fue el hecho de analizar la matriz de correlación de las variables empleadas.

	LATITUD	LONGITUD	nodo	tipo de evento	ageRange	PRECIPITACIONES-ANALISIS	Tavg	VIENTOvel	degree
LATITUD	1.000000	0.071435	-0.021917	0.001139	-0.012754	-0.002111	-0.002202	-0.003016	0.065041
LONGITUD	0.071435	1.000000	0.063222	0.001380	-0.001679	-0.002534	-0.002414	-0.003461	0.056532
nodo	-0.021917	0.063222	1.000000	0.000161	-0.009055	0.000745	-0.001166	-0.000052	-0.178488
tipo de evento	0.001139	0.001380	0.000161	1.000000	0.002022	0.000382	-0.005030	0.000425	-0.001738
ageRange	-0.012754	-0.001679	-0.009055	0.002022	1.000000	0.008965	-0.048490	0.029559	-0.001607
PRECIPITACIONES-ANALISIS	-0.002111	-0.002534	0.000745	0.000382	0.008965	1.000000	-0.165154	0.048004	-0.000471
Tavg	-0.002202	-0.002414	-0.001166	-0.005030	-0.048490	-0.165154	1.000000	-0.022254	0.000716
VIENTOvel	-0.003016	-0.003461	-0.000052	0.000425	0.029559	0.048004	-0.022254	1.000000	-0.000198
degree	0.065041	0.056532	-0.178488	-0.001738	-0.001607	-0.000471	0.000716	-0.000198	1.000000

Tabla 7. Matriz de correlación de variables

Como se puede apreciar, no existe ninguna correlación demasiado fuerte. Es más, ninguna de las variables alcanza una correlación de nivel medio con la variable “tipo de evento”.

Una vez definida la técnica a emplear, se procedió a estudiar los distintos algoritmos disponibles para la optimización del modelo.

5.3.2.1 Descenso del gradiente

Este algoritmo está diseñado para encontrar el valor mínimo dentro de una función. Suponiendo que la función en cuestión es la pérdida del modelo, esta técnica permite encontrar los valores que ofrecen el mejor resultado posible.

En cada iteración del sistema se calcula el gradiente de la curva de la función y a partir del valor obtenido, se establece cual es el mejor camino a seguir para encontrar el mínimo. En este algoritmo la magnitud de cada salto se establece al inicio del proceso mediante la tasa de aprendizaje del modelo. Sin embargo, existen otros algoritmos donde el valor de los saltos va variando para evitar que se produzcan bucles infinitos cuando la tasa de aprendizaje sea excesiva.

El algoritmo del descenso de gradiente es lento pero versátil, particularmente útil en casos de funciones multidimensionales como el caso que se trata en este proyecto. Sin embargo, es complicado dar con el valor adecuado para la tasa de aprendizaje ya que se corre el riesgo de caer en una divergencia infinita o de que se tarde demasiado en entrenar al modelo.

5.3.2.2 Algoritmo de Adagrad

A diferencia del algoritmo especificado en el punto 5.3.2.1 en el que la tasa de aprendizaje es estática, el algoritmo del gradiente adaptativo (*adaptative gradient*) utiliza distintos valores para cada iteración basándose en el valor del gradiente acumulado en cada dimensión [16]. Por desgracia, es posible que el valor de alguna variable descienda demasiado precipitadamente debido a la acumulación de altos valores del gradiente al comienzo del entrenamiento, lo que generaría que el sistema sea incapaz de utilizar dicha variable.

Dado que en este proyecto la correlación entre las variables con el valor de salida es muy pequeña, se puede suponer que los gradientes iniciales serán bastante

elevados, lo que llevaría a la correspondiente pérdida de estas variables como parámetros de entrenamiento. Por este motivo, podemos afirmar que este algoritmo no es válido para este proyecto.

5.3.2.3 Algoritmo Adadelta

Adadelta es una extensión del algoritmo Adagrad que busca reducir su agresividad a la hora de reducir el gradiente de las variables [17]. En lugar de almacenar el acumulado de todas las iteraciones previas, este algoritmo establece un número máximo de ciclos de almacenamiento. Al alcanzar este número máximo se elimina el registro más antiguo y se sobrescribe con el nuevo valor del gradiente. De esta forma se elimina el riesgo de que el algoritmo elimine variables que podrían ser útiles para el modelo más adelante. Por otro lado, el aprendizaje será mas lento debido a que la curva de aprendizaje será mucho más suave que en la versión anterior del algoritmo.

5.3.2.4 Algoritmo Adam

El algoritmo de Adam (*Adaptative Moment Estimation*) [18]es otro modelo que, al igual que Adagrad o Adadelta, utiliza tasas de aprendizaje adaptativas. Al igual que Adadelta, Adam también almacena parte de los acumulados de los gradientes, pero este además guarda los valores del incremento inmediatamente anterior a cada iteración. De esta forma, se puede realizar una estimación de cual es el sentido y la magnitud más adecuada para el salto en cada iteración.

Adam utiliza pasos globales en lugar de utilizar un paso distinto para cada variable, lo que lo convierte en un algoritmo muy útil cuando el modelo a entrenar es una red neuronal (*Deep Learning*).

5.3.2.5 Follow the Regularized Leader (FTRL)

En optimización convexa es ampliamente conocido el algoritmo Follow The Leader (FTL) [19] que se caracteriza por la facilidad con que resuelve problemas de predicción de la manera más intuitiva. El sistema guarda registro del desempeño de todos los elementos involucrados a lo largo del tiempo en el proceso estudiado, selecciona aquellos que hayan tenido un mejor rendimiento y los emplea en la siguiente ronda, actualizando estos resultados en cada iteración. Sin embargo, este algoritmo presenta dos problemas principales:

- El sistema a resolver puede ser un problema de optimización muy complejo, que suponga mucha capacidad de procesamiento o tiempo.
- Siempre hay una cantidad de pérdidas mínimas que no se pueden reducir, especialmente al emplear funciones lineares.

Por estos motivos se diseñó el algoritmo *Follow The Regularized Leader* (FTRL) [20], cuyo funcionamiento es igual que en el caso anterior, pero añadiendo un término regulador que estabiliza la solución del problema. En cada iteración, el sistema intenta reducir la cantidad de pérdidas, pero en esta ocasión elegirá el movimiento cuyas pérdidas sean mínimas tras ser evaluado en la función reguladora.

Esta función reguladora, cuyas características principales son que ha de ser doblemente diferenciable, fuertemente convexa y suave que se irá actualizando, igual que los movimientos de otros elementos, a medida que avance el sistema, fijando para cada posible movimiento un límite máximo de pérdidas.

El sistema a la hora de elegir su siguiente movimiento tendrá en cuenta no solo las pérdidas estimadas para la siguiente ronda, sino también las pérdidas máximas que puede llegar a producir dicho elemento. Sumará ambos valores para todos los posibles movimientos y se quedará con aquel cuyo valor total sea menor.

Efectuará ese movimiento y con los resultados de dicha ronda actualizará los valores almacenados para calcular la siguiente.

5.3.3 IMPLEMENTACIÓN DEL ALGORITMO

Una vez establecido el modelo del sistema y el algoritmo que se va a emplear para el entrenamiento, se procedió a la implementación del programa en el *cluster* de la universidad. Las funciones desarrolladas con tal fin son las siguientes:

- **train_linear_classifier_model**: A partir de los valores de la tasa de regulación, la tasa de aprendizaje, el número de ciclos y el tamaño de la muestra, realiza el entrenamiento y devuelve un modelo capaz de realzar predicciones. Además, representa el error cometido por el modelo en cada iteración.

- **data_extraction**: A partir del *dataframe* original devuelve dos *dataframes*, el de los valores de salida y el de los valores de entrada.

- **get_quantile_based_boundaries**: A partir de los datos de entrada y del número de divisiones que se quiera hacer por variable, convierte las variables numéricas del sistema en variables categóricas.

- **construct_feature_columns**: A partir de los datos de entrada devuelve la información como variables categóricas o numéricas

- **my_input_fn**: Recibe los parámetros de salida de la función construct_feature_*columns* y devuelve los datos procesados y listos para ser usados por el modelo de TensorFlow.

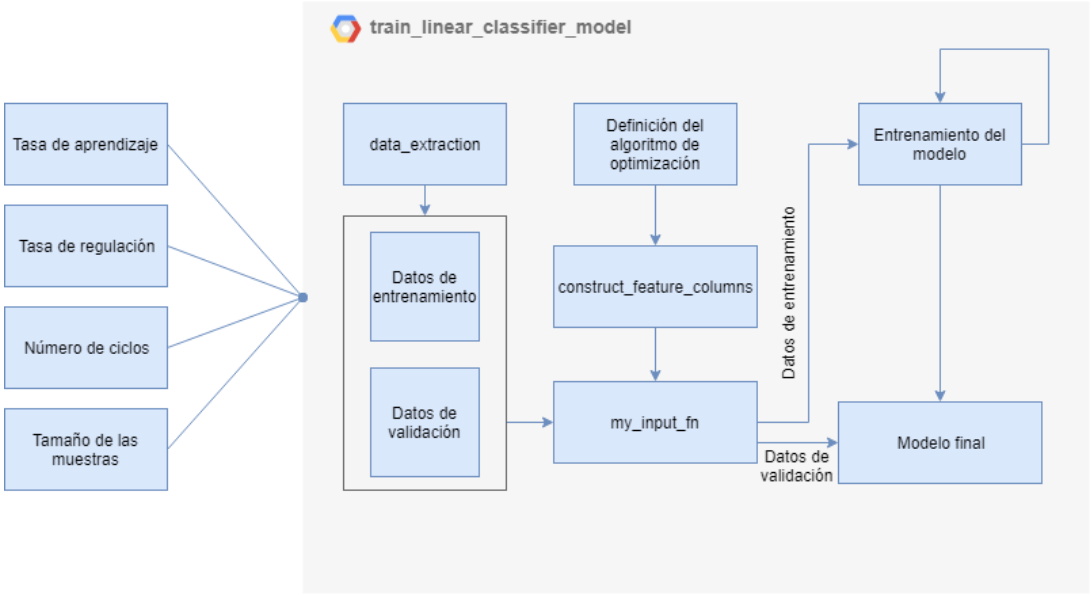


Figura 23. Diagrama de la función train_linear_classifier_model

Para el entrenamiento del sistema se utilizaron unos 13000 registros de eventos, los cuales se dividieron en una proporción de 58-42 entre los conjuntos de entrenamiento y validación. La tasa de aprendizaje elegida fue de 0.8, la tasa de regulación de 0.6 y el tamaño de las muestras de 500 eventos.

Tras preparar los datos para su posterior tratado, se introdujo la información almacenada en el modelo y se generaron 20 iteraciones para entrenar el sistema. Es necesario destacar que este hecho fue posible gracias al *cluster* de la universidad, sin el cual se habrían necesitado gran cantidad de horas para el entrenamiento que, en comparación, tan solo tardó unos pocos minutos. Al finalizar el entrenamiento se devuelve el modelo desarrollado y se imprime una gráfica con la que se puede comparar el error cometido a lo largo de todas las iteraciones valorando de este modo la eficiencia del modelo.

A pesar de que se puede suponer que el algoritmo que fue elegido era el más eficiente para el modelo en cuestión, se consideró que dada la inexperiencia en el campo del Big Data y al ser la primera vez que se escogía un algoritmo de este tipo,

era recomendable realizar una comparativa entre los errores registrados al utilizar distintos tipos de algoritmos. Gracias a los paquetes instalados en la librería TensorFlow fue posible implementar con relativamente poco esfuerzo todos los algoritmos descritos en la sección 5.3.2 en el modelo desarrollado. Sin embargo, para agilizar la recopilación de la información se decidió realizar únicamente 10 iteraciones por algoritmo.

En primer lugar, se puede ver los errores cometidos mediante el uso del algoritmo FTRL. Como se puede apreciar, la diferencia entre los errores de entrenamiento y validación es bastante elevado. Cuenta con una tasa de error final aceptable de 0.27. Sin embargo, no hay una gran mejoría entre los resultados de la primera iteración y de la última.

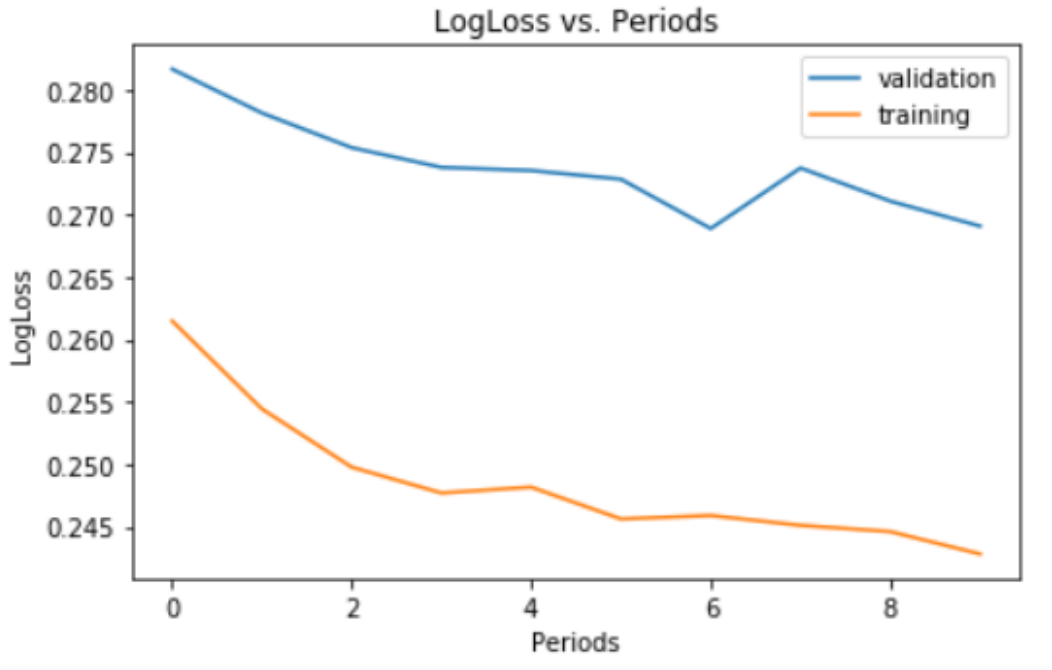


Figura 24. Tasa de error de validación y entrenamiento utilizando FTRL durante 10 iteraciones

Figura 25. Tasa de error de validación y entrenamiento utilizando descenso del gradiente

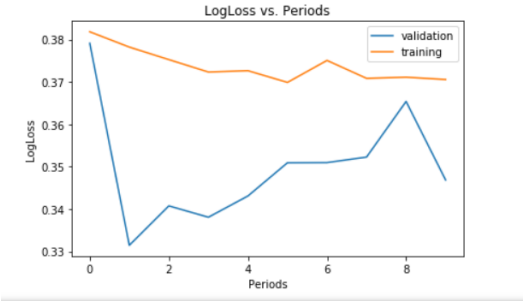


Figura 26. Tasa de error de validación y entrenamiento utilizando Adagrad

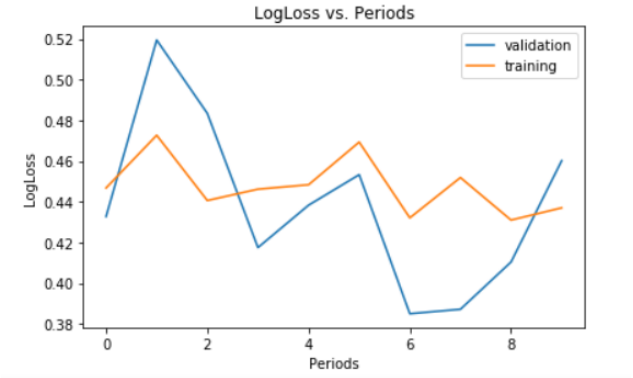


Figura 27. Tasa de error de validación y entrenamiento utilizando Adam

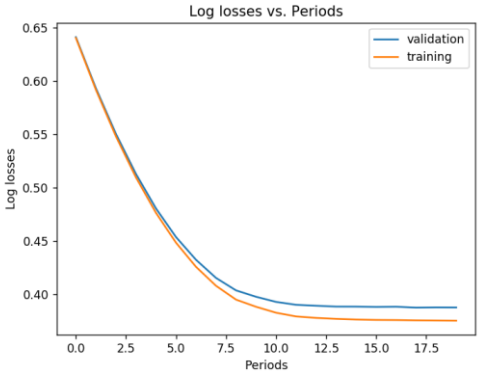


Figura 28. Tasa de error de validación y entrenamiento utilizando Adadelta

Como se puede apreciar en las imágenes anteriores, los demás algoritmos ofrecen tasas de error bastante elevadas. Además, en la mayoría de los casos las gráficas sufren variaciones muy bruscas lo que indica que el modelo no es muy estable. Como conclusión, se puede suponer que la elección del FTRL como algoritmo de optimización fue acertada.

A raíz de la implementación de los distintos algoritmos surgió la idea de desarrollar una sencilla red neuronal para comparar los resultados de la imagen 26 con los de la red.

A diferencia de lo que ocurrió durante la implementación de los algoritmos, el desarrollo de la red no fue una tarea sencilla e implicó la realización de cambios en varios aspectos del código, pese a ello, se mantuvo la misma estructura.

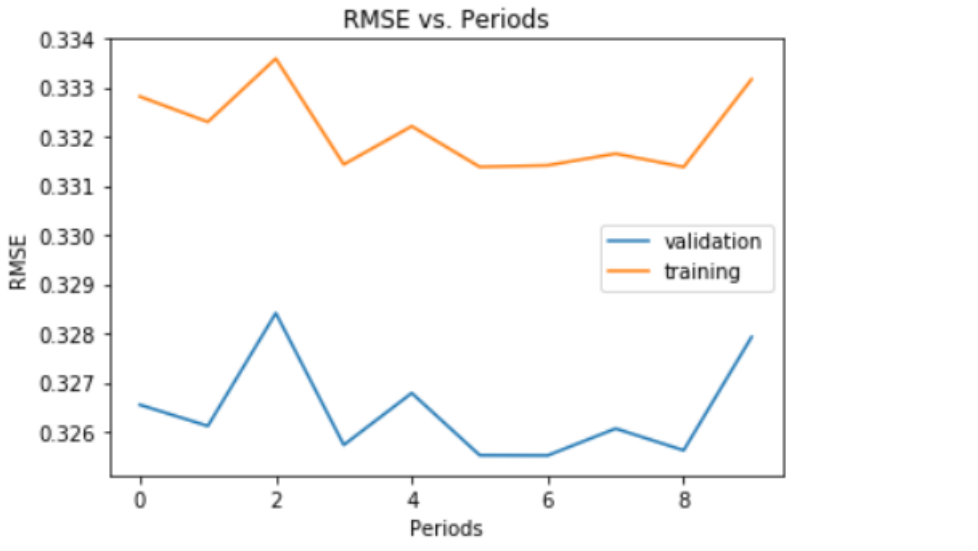


Figura 29. Tasa de error de entrenamiento y de validación usando FTRL en una red neuronal

La red diseñada cuenta con 3 capas de 10, 5 y 10 neuronas. Utiliza un optimizador FTRL, una función de activación ReLU y las mismas tasas que se usa durante el entrenamiento de la figura 25.

Como se puede apreciar, los resultados obtenidos son bastante buenos. El utilizar capas ocultas permite ofrecer una solución no lineal al problema y por tanto la tasa de error es reducida. Sin embargo, se puede observar que existe cierto sobre aprendizaje y la tasa de error sigue siendo superior a la obtenida al utilizar el análisis de las componentes independientes. Es razonable pensar que con otra configuración neuronal se podrían mejorar los resultados obtenidos. Sin embargo, la investigación asociada al desarrollo de dicha red llevaría un ingente coste de recursos y tiempo. Debido a estos motivos y al hecho de que actualmente se valora el poder justificar las decisiones tomadas por los modelos de *machine learning*, se decidió prescindir de la red neuronal y mantener el enfoque original.

Finalmente, se implementó un filtro que establece si existe probabilidad de accidente o no en caso de que se supere un valor umbral. Después de analizar los resultados obtenidos en las pruebas, se decidió establecer dicho umbral en el 95%.

5.4 *PROGRAMA DE VISUALIZACIÓN PRELIMINAR*

El desarrollo de una plataforma de visualización excede los limites definidos para el alcance de este proyecto. Sin embargo, se consideró necesario implementar un programa que permitiese representar las rutas calculadas para medir el grado de eficacia del modelo.

En un primer momento se utilizó el módulo de visualización de OSMNX que permite dibujar el grafo y las rutas. Sin embargo, quedó patente que, aunque representaba las rutas de forma eficiente, era necesario mejorar el sistema para poder identificar las calles por su nombre y ubicarlas así en la ciudad.

Figura 30. Visualización mediante grafo de las rutas calculadas

La plataforma desarrollada es una sencilla ventana lanzada desde el Jupyter Notebook de la universidad. Dicha ventana representa la ciudad de Madrid sobre un

mapa de calor que refleja los accidentes de tráfico con implicación de bicicletas que han ocurrido durante los últimos años. Las rutas calculadas son dibujadas en color rojo, azul y verde representando las rutas rápidas, seguras y muy seguras respectivamente.

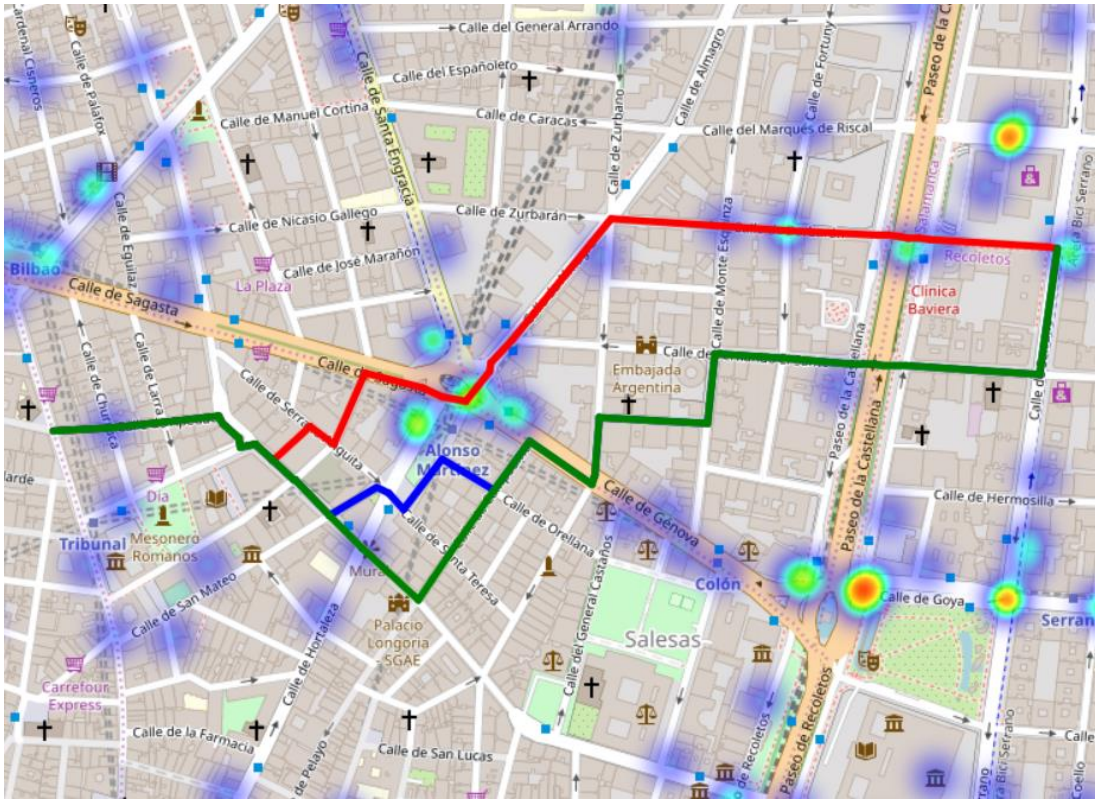


Figura 31. Visualización mediante ventana de las rutas calculadas

Capítulo 6. ANÁLISIS DE RESULTADOS

Existen dos agentes cuyos resultados deben analizarse. Por un lado, se encuentran los resultados obtenidos durante el desarrollo del modelo de machine learning, y por otro lado, el fruto del programa en su conjunto; incluyendo el modelo, el programa de trazado de rutas y la visualización preliminar.

6.1 RESULTADOS DEL MODELO DE MACHINE LEARNING

Como ya se mencionó en el apartado 5.3.3, se ha implementado el análisis de las componentes independientes como base para el modelo y se ha utilizado el algoritmo FTRL para la optimización del sistema. En apartados anteriores se probó la eficiencia del algoritmo tras 10 iteraciones. Sin embargo, la versión final del modelo utiliza 20 iteraciones, lo que cambia el aspecto de la gráfica.

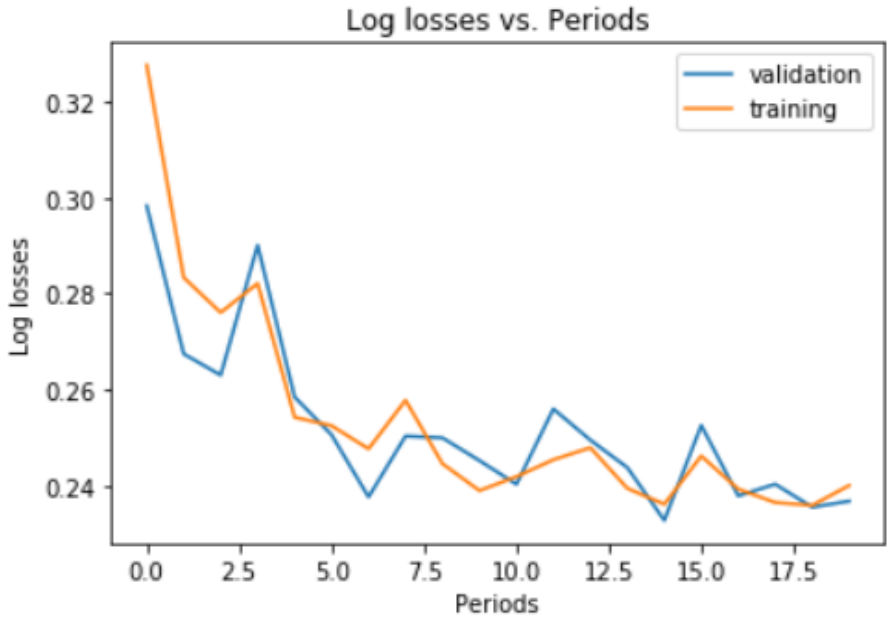


Figura 32. Tasa de error de entrenamiento y de validación utilizando el algoritmo FTRL durante 20 iteraciones

Se puede comprobar como la validación del modelo sigue la estela del entrenamiento sin caer en un sobre aprendizaje. Además, la precisión del sistema es bastante alta, este factor no se debe únicamente a la eficiencia del sistema, sino también al hecho de que existen muchos mas registros de trayectos que de accidentes, por lo que el modelo predice con mucha precisión la falta de incidencias. Para obtener una imagen más clara de lo eficaz que es el sistema es necesario calcular el *recall*, es decir, la proporción de accidentes detectados cuando realmente ha ocurrido un accidente.

Los resultados devueltos por un modelo de machine learning se pueden dividir en 4 casos según la siguiente tabla:

Verdaderos positivos (VP):	Falsos positivos (FP):
Falsos negativos (FN):	Verdaderos negativos (VN):

Tabla 8. Posibles salidas de un modelo de machine learning

Utilizando la nomenclatura explicada en la tabla anterior, se puede definir el *recall* según esta fórmula:

$$Recall = \frac{VP}{VP + FP}$$

Para mayor facilidad, el modelo desarrollado cuenta con registros que almacenan el valor de los campos relacionados con la eficacia del modelo. De esta forma se puede mostrar por pantalla el resultado obtenido.

```
INFO:tensorflow:Saving dict for global step 500: accuracy = 0.91838825, accuracy_baseline = 0.87973, auc = 0.88404876, auc_pre
cision_recall = 0.6483698, average_loss = 0.2367234, global_step = 500, label/mean = 0.12026999, loss = 0.2367234, precision =
0.7589041, prediction/mean = 0.10585631, recall = 0.62108844
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 500: /tmp/tmpha8t994o/model.ckpt-500
{'accuracy': 0.91838825, 'accuracy_baseline': 0.87973, 'auc': 0.88404876, 'auc_precision_recall': 0.6483698, 'average_loss':
0.2367234, 'label/mean': 0.12026999, 'loss': 0.2367234, 'precision': 0.7589041, 'prediction/mean': 0.10585631, 'recall': 0.621
08844, 'global_step': 500}
```

Figura 33. Estadísticas del modelo

La precisión del modelo es superior al 91% y el valor del *recall* es del 62%. Teniendo en cuenta la escasez de datos sobre accidentes y el hecho de que no se contaba con datos que podrían haber tenido gran relevancia para el modelo (velocidad del ciclista, tasa de alcohol en sangre, uso del teléfono móvil, tráfico en el momento del accidente, etc). Se puede considerar que una tasa de *recall* superior al 60% es bastante elevada y por tanto se puede calificar de éxito.

Otro parámetro usado para medir la calidad de un modelo es la curva ROC. Esta curva sirve para comparar la tasa de verdaderos positivos con la de falsos positivos aplicando distintos umbrales de clasificación. Se usa principalmente para medir el rendimiento de un modelo y compararlo con un modelo aleatorio. Cuanto menor sea el umbral mayor será la tasa de falsos positivos y viceversa. Por tanto, cuanto mayor sea la circunferencia de la curva mejor será el resultado obtenido.

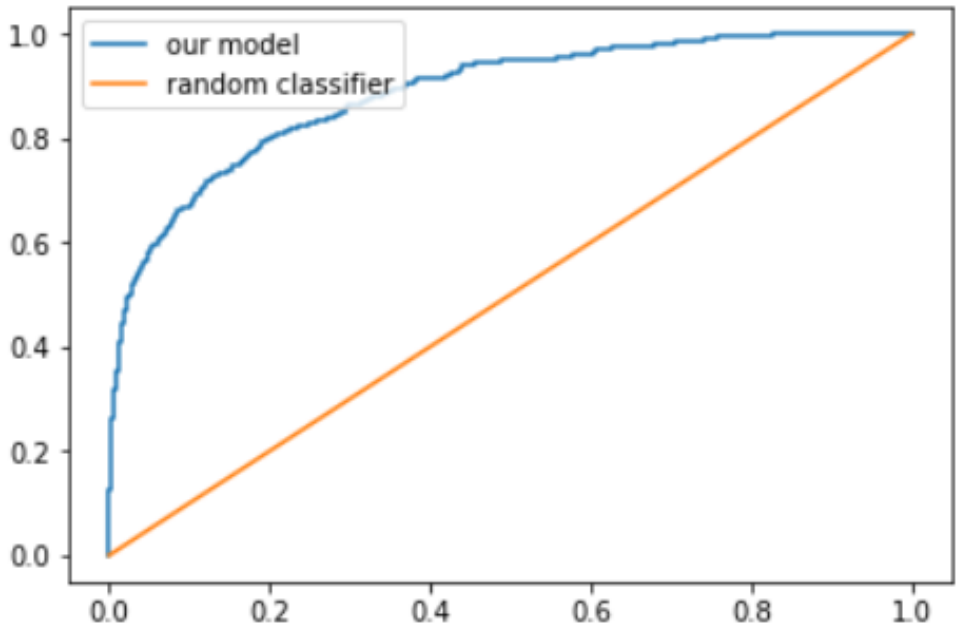


Figura 34. Curva AUC del modelo

6.2 *RESULTADOS DEL SISTEMA*

Partiendo de las coordenadas de origen y destino, el programa calcula en una primera iteración la ruta más corta posible basándose en la distancia. A continuación, se calculan los nodos del grafo por los que pasa la ruta y se introducen en un diccionario junto con la información del usuario, el estado de las calles y la situación meteorológica. Dicho diccionario se pasa como parámetro de entrada al modelo de *machine learning* que devuelve una lista con las probabilidades de sufrir un accidente en cada nodo del trayecto.

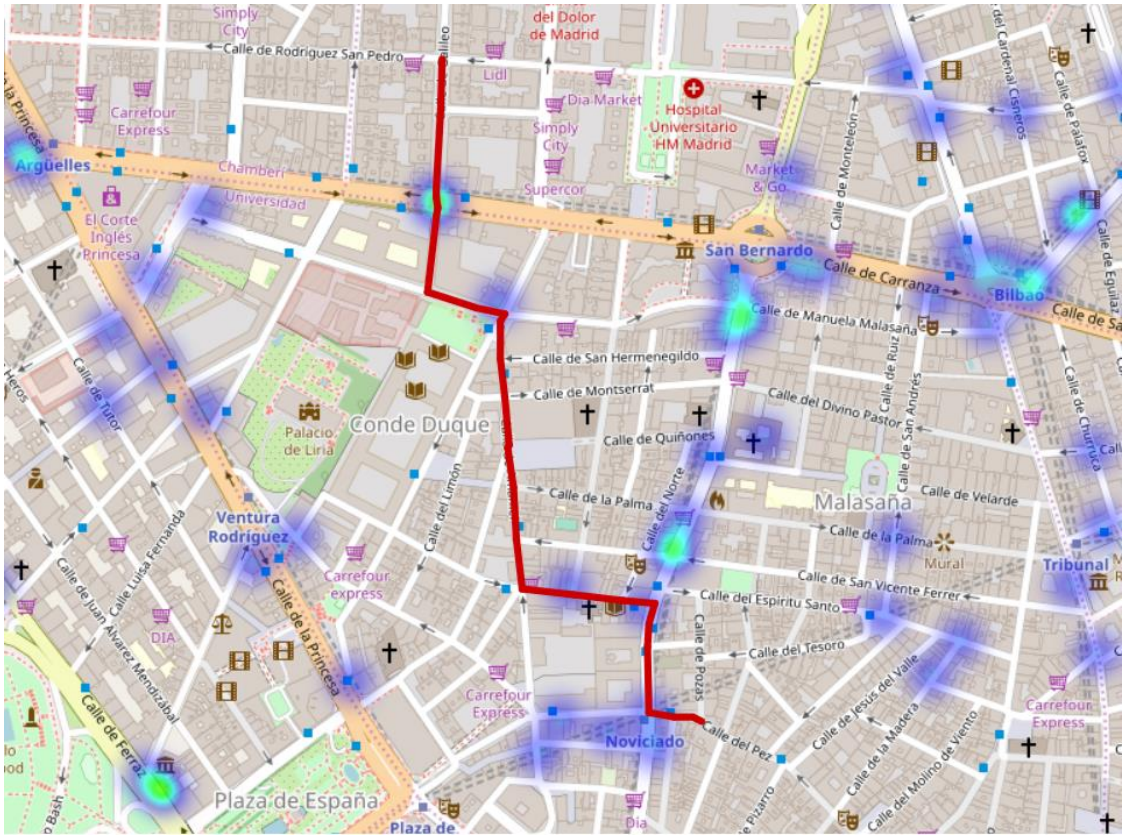


Figura 35. Ejemplo de la ruta más corta

En el caso de que las probabilidades superen el umbral preestablecido, se guardan los identificadores de los nodos dentro de una lista y se ejecuta el programa de trazado de rutas

alternativas. En las siguientes imágenes podemos ver ejemplos de las zonas detectadas por el sistema y de la nueva ruta calculada.

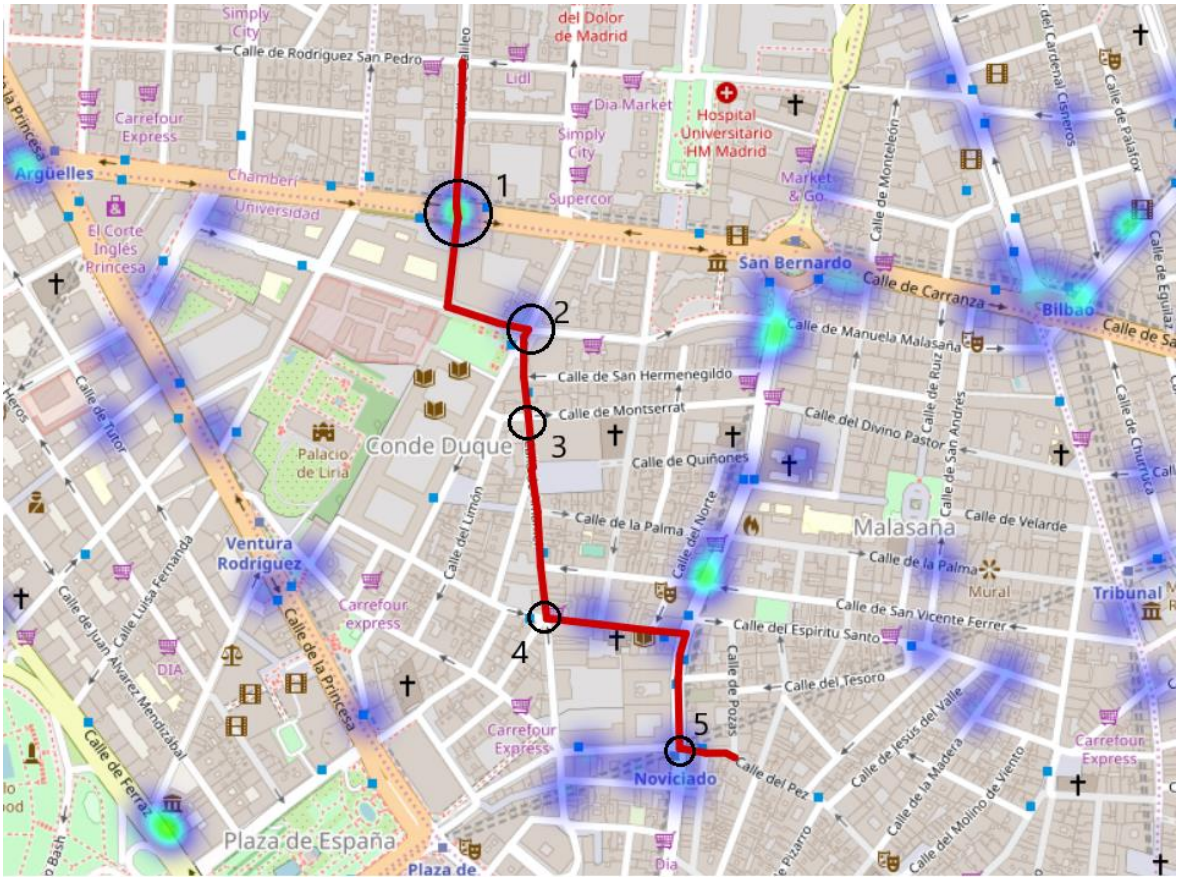


Figura 36. Identificación de las zonas detectadas como peligrosas en la ruta más corta

Se puede comprobar como las zonas detectadas se corresponden con lugares donde es lógico que pueda suceder un accidente. La zona 1 señala un cruce de calles anchas con gran concentración de accidentes, la zona 2 también ha registrado incidencias, aunque en menor grado que la zona 1, en la zona 3 no se ha sufrido ningún accidente en los últimos años pero, aun así, el sistema la detecta como peligrosa (debido a que no es posible realizar un experimento real que lo corrobore, no se puede afirmar con certeza que este caso sea correcto o un fallo del sistema). Por su parte, las zonas 4 y 5 corresponden a cruces cercanos

a zonas donde se han producido varios accidentes dispersos y, por tanto, se han calificado como peligrosas.

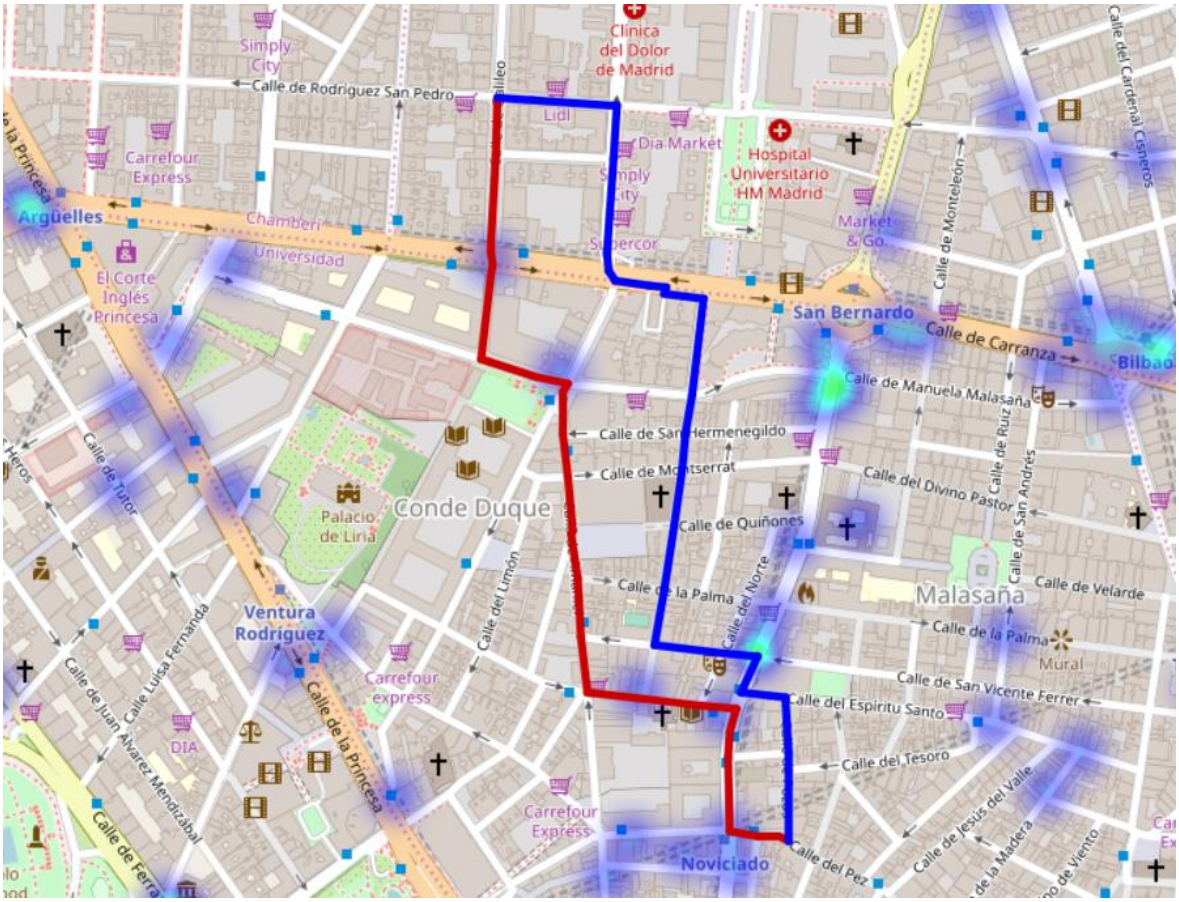


Figura 37. Ruta original vs Ruta segura

Para calcular la última ruta se vuelve a ejecutar el programa, pero utilizando como ruta inicial el camino calculado en el ciclo anterior y almacenando la información sobre las zonas peligrosas previamente identificadas.

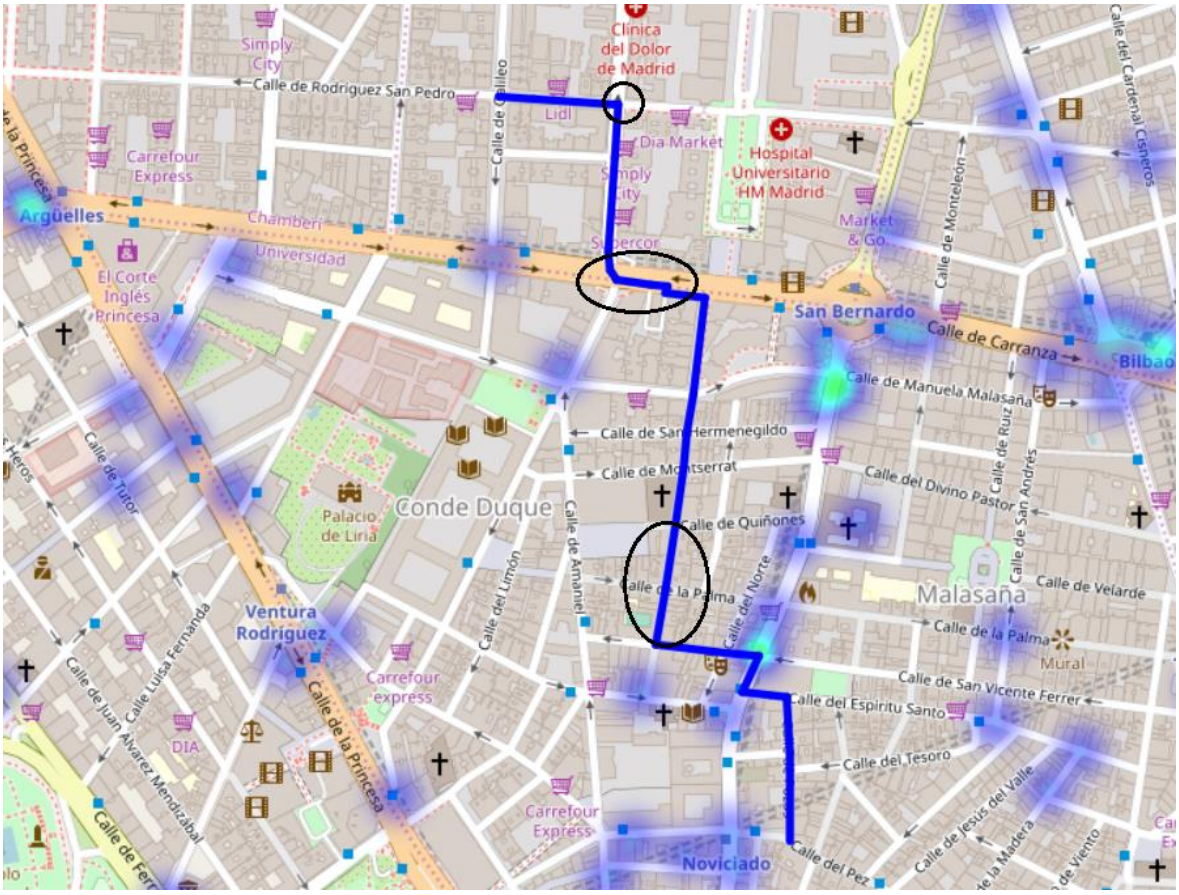


Figura 38. Identificación de las zonas detectadas como peligrosas en la primera ruta segura

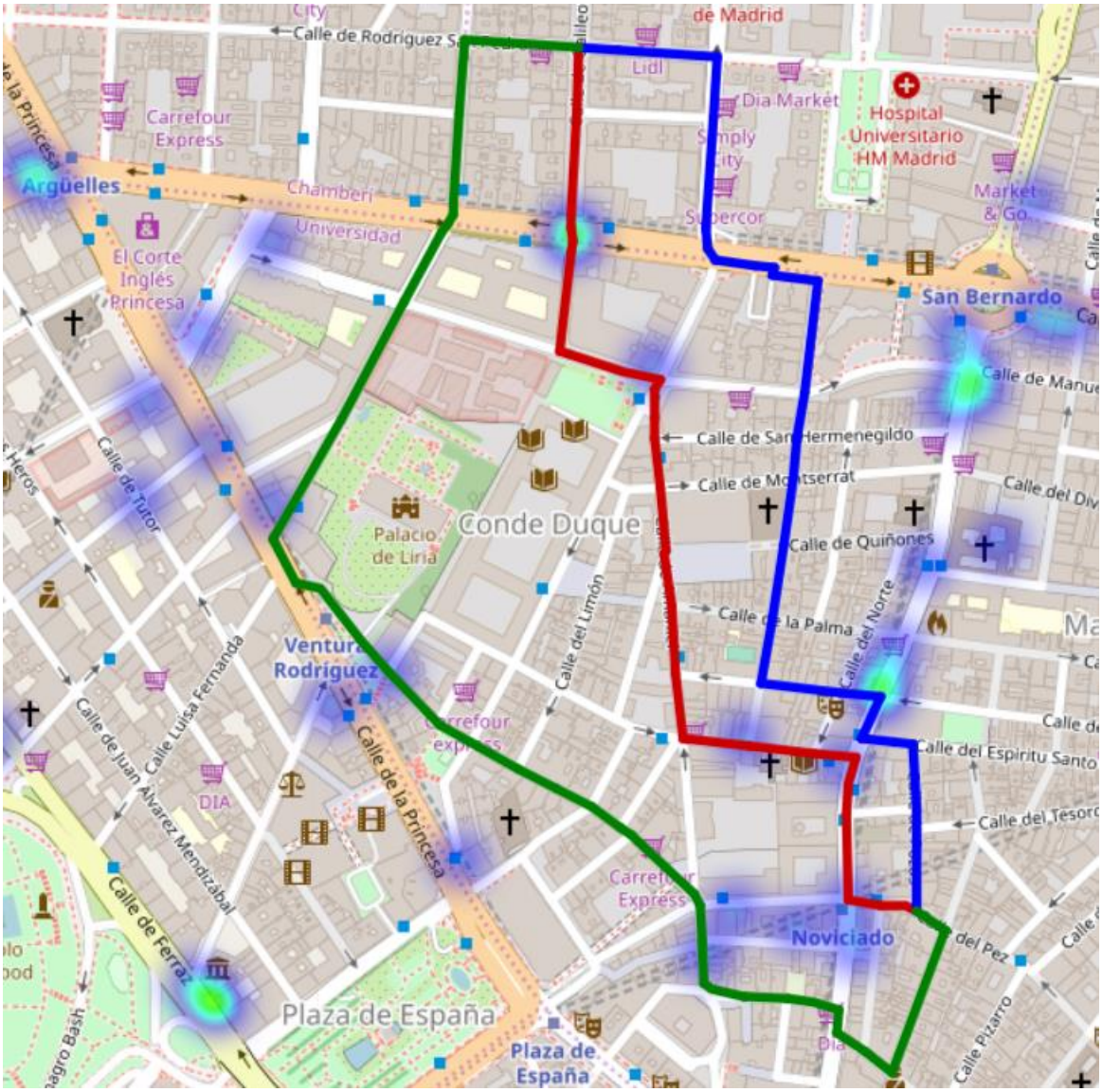


Figura 39. Ruta corta, ruta segura y ruta muy segura

En la última imagen se puede ver el resultado del módulo de visualización preliminar. Sin embargo, los datos reales que se enviarán al sistema de visualización que será implementado en otro Trabajo de Fin de Grado, son simplemente tres listas con las coordenadas de los trayectos.

Trayecto	Trayecto	Trayecto
[40.4292659, -3.6871448]	[40.4292659, -3.6871448]	[40.4292659, -3.6871448]
[40.4292739, -3.6872761]	[40.4278688, -3.6873875]	[40.4278688, -3.6873875]
[40.4293631, -3.6889778]	[40.4279673, -3.6893291]	[40.4279673, -3.6893291]
[40.429367, -3.6891342]	[40.4279723, -3.6894917]	[40.4279723, -3.6894917]
[40.4293782, -3.6892956]	[40.4279775, -3.6896608]	[40.4279775, -3.6896608]
[40.4293864, -3.6894134]	[40.4279813, -3.6897847]	[40.4279813, -3.6897847]
[40.4294006, -3.6895832]	[40.4279865, -3.6899512]	[40.4279865, -3.6899512]
[40.4294102, -3.6897311]	[40.4279911, -3.6901007]	[40.4279911, -3.6901007]
[40.4294624, -3.6910498]	[40.4280447, -3.6911057]	[40.4280447, -3.6911057]
[40.4295059, -3.6919569]	[40.428097, -3.6920876]	[40.428097, -3.6920876]
[40.4295871, -3.6935978]	[40.4273322, -3.6922023]	[40.4273322, -3.6922023]
[40.4282543, -3.6950398]	[40.4274035, -3.6937766]	[40.4274035, -3.6937766]
[40.4279981, -3.6953247]	[40.4267604, -3.6938292]	[40.4267604, -3.6938292]
[40.4279098, -3.6953497]	[40.4266661, -3.6938711]	[40.4266661, -3.6938711]
[40.4277374, -3.6955312]	[40.4271361, -3.694768]	[40.4271361, -3.694768]
[40.4275732, -3.6956967]	[40.4265916, -3.695292]	[40.4265916, -3.695292]
[40.4276324, -3.6960384]	[40.4269353, -3.6960447]	[40.4258923, -3.6959231]
[40.4277166, -3.6963288]	[40.4263838, -3.6966036]	[40.4253839, -3.6963927]
[40.427886, -3.6971498]	[40.4265715, -3.6968128]	[40.4259592, -3.697206]
[40.4271084, -3.6976019]	[40.4263178, -3.6976693]	[40.4263178, -3.6976693]
[40.4273042, -3.6979548]	[40.4267068, -3.6981718]	[40.4267068, -3.6981718]
[40.4269178, -3.6984819]	[40.4267141, -3.6981874]	[40.4267141, -3.6981874]
[40.4271013, -3.6987379]	[40.4269178, -3.6984819]	[40.4269178, -3.6984819]
[40.4270639, -3.6989487]	[40.4271013, -3.6987379]	[40.4271013, -3.6987379]
[40.427162, -3.6989761]	[40.4270639, -3.6989487]	[40.4270639, -3.6989487]
[40.4274109, -3.6992856]	[40.427162, -3.6989761]	[40.427162, -3.6989761]
[40.4273481, -3.7000637]	[40.4274109, -3.6992856]	[40.4274109, -3.6992856]
[40.4272902, -3.7008402]	[40.4273481, -3.7000637]	[40.4273481, -3.7000637]
[40.4272493, -3.7016691]	[40.4272902, -3.7008402]	[40.4272902, -3.7008402]
	[40.4272493, -3.7016691]	[40.4272493, -3.7016691]

Figura 40. Lista de coordenadas para el módulo de visualización

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este proyecto se ha desarrollado un modelo de *machine learning* que implementa la técnica del análisis de las componentes independientes junto con el algoritmo de optimización FTRL. El sistema en cuestión se ha demostrado capaz de calcular, con una precisión teórica elevada, las probabilidades de sufrir un accidente al circular con una bicicleta en función de la ruta, el usuario y el tiempo atmosférico.

Todos los objetivos planteados en el punto 4.2 se han alcanzado, con la excepción de que el sistema genera errores en caso de que los puntos de origen y destino se encuentren rodeados de zonas con una densidad de peligro elevada. Esto se debe a que en dichas situaciones el grafo se fragmenta impidiendo que se calcule la ruta más corta.

Como trabajos futuros quedan los siguientes puntos:

1. Encontrar nuevas fuentes de información para entrenar al sistema y obtener más registros sobre los campos con los que ya se cuenta, en particular sobre accidentes.
2. Mejorar la velocidad de procesamiento para calcular la ruta más corta de forma que el sistema sea capaz de ofrecer rutas con un tiempo de respuesta comparable al de otras plataformas de navegación como Google Maps.
3. Reducir la tasa de error a la vez que se eleva el *recall* para mejorar la eficiencia del sistema y poder ofrecer mejores resultados.
4. Solucionar el problema del grafo fragmentado de forma que el sistema no aísle ningún nodo que sea crítico para el funcionamiento del sistema.

Capítulo 8. BIBLIOGRAFÍA

[1] J. Q. a. L. Chun, «The Measures Study of Electric Bicycle's Hidden Safety Danger in Traffic in Moderate City,» de *International Conference on Industrial Control and Electronics Engineering*, Xi'an, 2012.

[2] I. Ferrer, «Un año nefasto para los ciclistas,» *El País*, p. Available: https://elpais.com/elpais/2018/05/03/opinion/1525356592_500341.html, 03 05 2018.

[3] S. A. A. Y. Y. K. a. Y. T. S. Kaneda, «A Hazard Detection Method for Bicycles by Using Probe Bicycle,» de *IEEE 38th International Computer Software and Applications Conference Workshops*, Vasteras, 2014 .

[4] J. M. A.-J. a. A. Becerra-Terón, «Distance Based Queries in Open Street Map,» de *International Workshop on Database and Expert Systems Applications (DEXA)*, Valencia, 2015.

[5] «spark.apache,» [En línea]. Available: https://spark.apache.org/.

[6] A. d. Madrid, «datos.madrid,» Ayuntamiento de Madrid, 2018. [En línea]. Available: https://datos.madrid.es/portal/site/egob.

[7] H. H. J. L. a. S. C. J. Jiang, «Extending Dijkstra's shortest path algorithm for software defined networking,» de *The 16th Asia-Pacific Network Operations and Management Symposium*, Hsinchu, 2014.

[8] M. A.Shipp, «Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning,» *Nature*, 2002.

[9] K. D. C. U. a. P. Y. I. Kumar, «International Conference on Inventive Communication and Computational Technologies,» de *A Comparative Study of Supervised Machine Learning Algorithms for Stock Market Trend Prediction*, Coimbatore, 2018.

[10] «Google Maps ya informa de la dificultad para aparcar en algunas ciudades de España,» *Xataka*, 2017.

[11] A. H. F.-h. H. Murray Campbell, «Deep Blue,,» *Artificial Intelligence*, vol. 134, pp. 57-83, 2002.

[12] «Mastering the game of Go with deep neural networks and tree search,» *Nature*, 2016.

[13] C. J. Jolliffe IT, «Principal component analysis: a review and recent developments».

[14] I. Aldecoa Bilbao, N. Carné Serrano y E. Monte Moreno, «Separación ciega de fuentes,» 2003.

[15] S. &. C. Z. Haykin, «The Cocktail Party Problem. Neural computation,» 2005.

[16] H. Kato, Y. Sakajyo y S. Kaneda, «Visualization Method for Bicycle Rider Behavior Analysis Using a Smartphone,» de *Computer Software and Applications Conference (COMPSAC), IEEE Annual International*, Turin, 2017.

[17] «Open Street Map,» [En línea]. Available: https://osmnx.readthedocs.io/en/stable/.

[18] J. &. H. E. &. S. Y. C. Duchi, «Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,» *Journal of Machine Learning Research.*

[19] M. D. Zeiler, «ADADELTA: An adaptive learning rate method».

[20] D. &. B. J. Kingma, «Adam: A Method for Stochastic Optimization,» de *International Conference on Learning Representations*, 2014.

[21] A. Bykov, «Follow-The-Regularized-Leader,» 2014.

[22] A. Blum, «Foundations of Machine Learning and Data Science,» 2015.