Texto

Descripción generada automáticamente

Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Análisis y Visualización de Datos Masivos/ Visual Analytics and Big Data

“Biker”: un innovador sistema de análisis de datos sobre el ciclismo en el País Vasco

|  |  |
| --- | --- |
| Trabajo fin de estudio presentado por: | Iker Sebastián Pérez |
| Tipo de trabajo: | Desarrollo Software |
| Director/a: | Henry Eduardo Baquero Vega |
| Fecha: | 11/09/2024 |

Resumen

La continua evolución en las tecnologías de la información ha posicionado los datos como epicentro de numerosos ámbitos de la sociedad. Desde su recogida y limpieza hasta la aplicación de sus más novedosas técnicas de Inteligencia Artificial (IA), el dato continúa aportando innumerables ventajas a aquellos quienes lo usan en mayor o menor medida. No obstante, ninguna de estas fases del ciclo de vida de los datos ha tenido tanto impacto en los últimos años como la visualización de datos.

Es la visualización de estos la que ha aumentado su uso exponencialmente, entre otros factores, debido a la aparición de diversas herramientas dedicadas a esta finalidad. Estas herramientas destacan por su alta usabilidad y su intuitiva interfaz, lo que las hace accesibles y fáciles de integrar en los sistemas ya existentes.

En este proyecto, se lleva a la práctica el ciclo de vida completo de los datos. Dicho proceso se inicia con la recogida de datos a través de interfaces de programación de aplicaciones (APIs). Con los valores recogidos, se lleva a cabo la etapa de limpieza y procesamiento para su posteriormente almacenamiento. Seguidamente, se aplican técnicas de IA como regresiones o redes neuronales, las cuales son necesarias para la obtención de información relevante para el usuario final. Finalmente, se culmina el ciclo con la visualización de la información obtenida, permitiendo al usuario convertir la información presentada en conocimiento.

La razón principal para la realización de este proyecto es el poder aplicar en un entorno real el ciclo de vida de un conjunto de datos al completo, desde su recogida hasta su visualización. Con ello, se podrá comprender el potencial de cada fase, así como el de las herramientas utilizadas, desde un punto de vista totalmente práctico.

**Palabras clave:** Ciclo de vida del dato, recogida de datos, gestión de datos, análisis de datos, visualización de datos.

Abstract

The continuous evolution in information technologies has positioned data as the core of many areas of society. From its collection and cleaning to the application of the most innovative Artificial Intelligence (AI) techniques, data continues to provide great advantages to those who use it. However, none of these phases of data life cycle has generated as much impact in recent years as data visualization has.

It is visualization of data what has increased its use exponentially, among other factors, due to the appearance of various tools dedicated to this purpose. These tools stand out for their high usability and intuitive interface, which makes them accessible and easy to integrate into existing systems that deal with data.

In this project, the full data lifecycle is implemented. This process begins with the collection of data through application programming interfaces (APIs). Once data is collected, the cleaning and processing stage go on, to subsequently store the data. Next, AI techniques such as regressions or neural networks are applied, which are necessary to get high-relevance information for the end user. Finally, the process ends up with the visualization of the information obtained, allowing the user to turn the information presented into knowledge.

The main reason for going through this project is to be able to apply the life cycle of a complete dataset in a real environment, from its collection to its visualization. With this, it will be possible to understand the potential of each phase, as well as that of the tools used, from a totally practical point of view.

**Keywords**: Data life cycle. Data collection. Data management. Data analysis. Data visualization.

Índice de contenidos

[1. Introducción 1](#_Toc176897958)

[1.1. Motivación 2](#_Toc176897959)

[1.2. Planteamiento del trabajo 3](#_Toc176897960)

[1.3. Estructura del trabajo 4](#_Toc176897961)

[2. Contexto y estado del arte 7](#_Toc176897962)

[2.1. Contexto del problema 7](#_Toc176897963)

[2.2. Estado del arte de herramientas similares existentes 8](#_Toc176897964)

[2.2.1. MeteoRuta 8](#_Toc176897965)

[2.2.2. Comobity 9](#_Toc176897966)

[2.2.3. TuTiempo.net 10](#_Toc176897967)

[2.2.4. Flare 11](#_Toc176897968)

[2.3. Estado del arte de las tecnologías a utilizar 12](#_Toc176897969)

[2.3.1. Recogida de datos 12](#_Toc176897970)

[2.3.2. Limpieza y procesamiento de datos 14](#_Toc176897971)

[2.3.3. Almacenamiento de datos 15](#_Toc176897972)

[2.3.4. Análisis de datos 16](#_Toc176897973)

[2.3.5. Visualización de datos 19](#_Toc176897974)

[2.4. Conclusiones 23](#_Toc176897975)

[3. Objetivos concretos y alcance 25](#_Toc176897976)

[3.1. Objetivo general 25](#_Toc176897977)

[3.2. Objetivos específicos 25](#_Toc176897978)

[3.3. Alcance 26](#_Toc176897979)

[4. Metodología de trabajo y arquitectura 28](#_Toc176897980)

[4.1. Metodología de trabajo 28](#_Toc176897981)

[4.2. Arquitectura 29](#_Toc176897982)

[5. Marco normativo 31](#_Toc176897983)

[6. Desarrollo específico de la contribución 32](#_Toc176897984)

[6.1. Recolección de los datos 33](#_Toc176897985)

[6.2. Limpieza y procesamiento de datos 36](#_Toc176897986)

[6.2.1. Datos ligados al tráfico 37](#_Toc176897987)

[6.2.2. Datos ligados a la meteorología 40](#_Toc176897988)

[6.3. Almacenamiento de datos 45](#_Toc176897989)

[6.4. Análisis de datos 46](#_Toc176897990)

[6.4.1. Análisis exploratorio de datos (EDA) 47](#_Toc176897991)

[6.4.2. Análisis de los subconjuntos de datos 48](#_Toc176897992)

[6.4.3. Exportar resultados del análisis 54](#_Toc176897993)

[6.5. Visualización de datos 55](#_Toc176897994)

[6.5.1. Análisis meteorológico 55](#_Toc176897995)

[6.5.2. Análisis de la calidad del aire 56](#_Toc176897996)

[6.5.3. Análisis de los flujos de tráfico 57](#_Toc176897997)

[6.5.4. Análisis de incidencias 58](#_Toc176897998)

[7. Código fuente y datos analizados 60](#_Toc176897999)

[7.1. Repositorio 60](#_Toc176898000)

[7.2. Código fuente 60](#_Toc176898001)

[7.2.1. main.py 61](#_Toc176898002)

[7.2.2. Archivos generales 67](#_Toc176898003)

[7.2.3. analisis.py 67](#_Toc176898004)

[7.3. Datos Analizados 73](#_Toc176898005)

[8. Plan de trabajo y gestión del proyecto 74](#_Toc176898006)

[8.1. Planificación del proyecto 74](#_Toc176898007)

[8.1.1. Cronograma 74](#_Toc176898008)

[8.1.2. Análisis DAFO 75](#_Toc176898009)

[8.1.3. Análisis CAME 76](#_Toc176898010)

[8.2. Problemas encontrados, limitaciones y desvíos 77](#_Toc176898011)

[8.2.1. Recolección de los datos 77](#_Toc176898012)

[8.2.2. Limpieza y procesamiento de los datos 78](#_Toc176898013)

[8.2.3. Almacenamiento de los datos 78](#_Toc176898014)

[8.2.4. Análisis de los datos 79](#_Toc176898015)

[8.2.5. Visualización de los datos 79](#_Toc176898016)

[9. Resultados 80](#_Toc176898017)

[9.1. Cumplimiento de los objetivos 80](#_Toc176898018)

[10. Conclusiones 82](#_Toc176898019)

[10.1. Lecciones aprendidas 83](#_Toc176898020)

[11. Trabajo futuro 84](#_Toc176898021)

[Referencias bibliográficas 85](#_Toc176898022)

[Glosario 90](#_Toc176898023)

[Lista de siglas y acrónimos 92](#_Toc176898024)

Índice de figuras

[**Figura 1.** *Representación visual de las fases del ciclo de vida del dato.* 4](#_Toc176898025)

[**Figura 2.** *Captura de la web MeteoRuta.* 8](#_Toc176898026)

[**Figura 3.** *Captura de la aplicación Comobity.* 10](#_Toc176898027)

[**Figura 4.** *Captura de la web TuTiempo.net.* 11](#_Toc176898028)

[**Figura 5.** *Ejemplo de visualización con librerías de Python.* 19](#_Toc176898029)

[**Figura 6.** *Ejemplo de visualización con D3.* 20](#_Toc176898030)

[**Figura 7.** *Ejemplo de visualización con Power BI.* 21](#_Toc176898031)

[**Figura 8.** *Arquitectura de las fases del ciclo de vida del dato.* 29](#_Toc176898032)

[**Figura 9.** *Registros de la colección “incidencias” en MongoDB.* 38](#_Toc176898033)

[**Figura 10.** *Registros de la colección “flows” en MongoDB.* 40](#_Toc176898034)

[**Figura 11.** *Registros de la colección “calidad\_aire” en MongoDB.* 41](#_Toc176898035)

[**Figura 12.** *Registros de la colección “estaciones” en MongoDB.* 43](#_Toc176898036)

[**Figura 13.** *Registros de la colección “meteo” en MongoDB.* 45](#_Toc176898037)

[**Figura 14.** *Colecciones de la base de datos “TFM” en MongoDB.* 46](#_Toc176898038)

[**Figura 15.** *Ejemplo de una matriz de correlación.* 47](#_Toc176898039)

[**Figura 16.** *Clustering de los flujos de tráfico.* 48](#_Toc176898040)

[**Figura 17.** *Clustering de las incidencias.* 52](#_Toc176898041)

[**Figura 18.** *Colecciones del análisis en la base de datos “TFM” en MongoDB.* 54](#_Toc176898042)

[**Figura 19.** *Registro de las predicciones en la base de datos.* 54](#_Toc176898043)

[**Figura 20.** *Registro del clustering en la base de datos.* 55](#_Toc176898044)

[**Figura 21.** *Dashboard del análisis meteorológico.* 56](#_Toc176898045)

[**Figura 22.** *Dashboard del análisis de la calidad del aire.* 57](#_Toc176898046)

[**Figura 23.** *Dashboard del análisis de los flujos de tráfico.* 58](#_Toc176898047)

[**Figura 24.** *Dashboard del análisis de incidencias.* 59](#_Toc176898048)

[**Figura 25.** *Estructura organizativa del software.* 60](#_Toc176898049)

[**Figura 26.** *Resultados de las predicciones de la calidad del aire.* 70](#_Toc176898050)

[**Figura 27.** *Resultados de las predicciones de las mediciones meteorológicos.* 72](#_Toc176898051)

[**Figura 28.** *Estructura organizativa de la copia de los datos.* 73](#_Toc176898052)

[**Figura 29.** *Cronograma y su división por sprints.* 74](#_Toc176898053)

[**Figura 30.** *Análisis DAFO.* 75](#_Toc176898054)

Índice de tablas

[Tabla 1. *Tabla comparativa de las herramientas analizadas.* 12](#_Toc176898055)

[Tabla 2. *Tabla comparativa de herramientas de visualización.* 22](#_Toc176898056)

[Tabla 3. *Tabla de resultados* *con las predicciones de la calidad del aire.* 50](#_Toc176898057)

[Tabla 4. *Tabla de resultados* *con las predicciones de los datos meteorológicos.* 53](#_Toc176898058)

[Tabla 5. *Tabla de cumplimiento de objetivos.* 81](#_Toc176898059)

# Introducción

La evolución tecnológica de los últimos años ha propulsado la investigación de muchos ámbitos que afectan de forma directa al ser humano. Estos avances tecnológicos han tenido un considerable impacto en la sociedad, cambiado hábitos tanto a nivel individual, con el aumento de dispositivos Internet of Things (IoT), como en el ámbito industrial, donde empresas multinacionales se han visto obligadas a digitalizar.

En todos estos casos, el dato ha ganado relevancia, convirtiéndose en la unidad esencial para el ser humano en su día a día. Por ello, y sumado a que la transición a la era digital no ha hecho más que comenzar, se ha querido enfatizar la relevancia del dato, posibilitando así un proyecto donde se trabaje cada una de sus fases del ciclo de vida en mayor detalle.

En concreto, se presenta el proyecto de “Biker”, una aplicación para todo aquel aficionado al ciclismo en el País Vasco, donde se ofrece al usuario final la posibilidad de conocer cierta información de un alto interés a la hora de practicar dicho deporte. En la aplicación, se tratan conjuntos de datos que recogen las incidencias existentes en las carreteras del País Vasco, así como el número de vehículos que transitan las mismas. También se utilizan datos del clima para analizar ambos conjuntos de manera unificada y encontrar relaciones e información que aporte un valor añadido a los ciclistas.

Dicho tratamiento de los datos comienza con la extracción de éstos desde las APIs públicas de Open Data Euskadi y Euskalmet. Desde estas entidades, se facilita la recogida de valores públicamente, siendo los relacionados con el tráfico y el clima los almacenados para este proyecto. Entre todos estos, se seleccionan los que interesen especialmente para llevar a cabo el análisis. Posteriormente, se formatean los datos seleccionados y se guardan en distintas colecciones de la base de datos de MongoDB en local. Esta base de datos realiza la función de almacén de datos (Data Warehouse).

Seguidamente, se explotan los registros con tal de extraer las relaciones que se consideren más relevantes. De esta forma, se obtienen predicciones de índices tanto meteorológicos como de tráfico, para estimar los potenciales valores de los días siguientes. Estas predicciones son posibles gracias a la aplicación de varias técnicas de IA, entre las que destacan la regresión lineal, Random Forest y la red neuronal.

Finalmente, se visualiza la información extraída de todo el análisis, utilizando herramientas de Business Intelligence (BI). Como línea futura, se podría valorar la creación de una aplicación móvil, de cara a ganar en accesibilidad y manejabilidad en el uso de esta herramienta.

En definitiva, mediante este proyecto se propone realizar el desarrollo del ciclo de vida completo de un conjunto de datos referente al tráfico y el clima en el País Vasco. Dicho desarrollo, se enfoca al sector ciclista, cuyos usuarios pueden obtener información novedosa y útil, gracias a las diferentes técnicas de IA utilizadas. Asimismo, resaltar que una de las finalidades principales del proyecto consiste en facilitar la información recogida a través de las visualizaciones a los usuarios finales, siendo necesario la realización de todo el proceso previo, y permitiendo interpretar a su vez los diferentes índices e indicadores que resumen el estado del clima y tráfico en el País Vasco.

## Motivación

Las principales razones para realizar este proyecto se explican a continuación:

Durante estos últimos años, el auge en la explotación de los datos ha incrementado de forma exponencial, siendo un aspecto esencial en la transformación digital que se está llevando a cabo en cada vez más empresas. Esto se debe a los tantos beneficios que una buena gestión de los datos puede aportar, siendo clave en la toma de decisiones de cualquier ámbito. Por ello, mediante este proyecto se ha decidido profundizar en la totalidad de las fases que conforman el ciclo de vida de los datos, tomando un conjunto y aplicando su ciclo de vida al completo, desde la recogida y almacenamiento hasta su análisis y visualización.

Además, el tema del que trata el conjunto de datos y el enfoque que se le proporciona tiene cierta afección e interés personal, ya que la aplicación toma como base el ciclismo, un deporte muy común en la región del País Vasco. Al ser nativo de esta zona geográfica y haber podido vivir de primera mano todo el impacto que dicho deporte genera en la población vasca, supone un aliciente más para el desarrollo de esta idea, pudiendo tal vez en un futuro generar una aplicación con un alcance mayor.

Por otra parte, la decisión de realizar este proyecto también engloba la linealidad de trabajar sobre la seguridad, no en cuanto a sistemas informáticos se refiere en este caso, si no en la seguridad física de los ciclistas. La seguridad es un tema que siempre ha sido relevante y que se ha ido adaptando a los continuos cambios que ha sufrido cualquier sector. Sin embargo, en la gran mayoría de circunstancias ha sido un aspecto infravalorado, siendo un variable más de cualquier ecuación cuando en realidad debería ser el cimiento principal sobre el cual se debería construir. En cuanto a la aplicación se refiere, se permite la consulta tanto del tráfico como de la climatología, lo que deduce el nivel de seguridad que se estima para la práctica del ciclismo bajo dichas condiciones.

Algo similar ocurre con la actividad física, ya que mediante este proyecto se quiere propulsar e incentivar sobre todo a las jóvenes generaciones a la práctica del deporte, en este caso el ciclismo. Mediante herramientas de este calibre, se facilita la planificación de las sesiones que se quieran realizar y permitir un acercamiento mayor hacia el mismo.

En líneas generales, una de las ideas principales que recogen los puntos previamente comentados hace énfasis en la relevancia de aplicar prácticamente un proceso que tiene un uso muy notorio en la sociedad actual, como es el ciclo de vida de los datos. Dicho proceso aporta un gran valor añadido en un amplio abanico de escenarios, el cual se prevé que siga ampliándose en los próximos años. Análogamente, se destaca la importancia de fomentar la práctica del deporte, así como de hacerlo en un entorno seguro.

## Planteamiento del trabajo

Ante la carencia de una aplicación dedicada al ciclismo que integre tanto información meteorológica como propiedades del tráfico, se ha identificado una considerable necesidad en la sociedad, la cual se intenta cumplimentar a través del presente proyecto.

Consecuentemente emerge “Biker”, un proyecto que busca facilitar la planificación en la práctica del ciclismo mediante la fácil obtención de información sobre el clima y el tráfico en el País Vasco. Se presenta una solución que cubre una necesidad existente en la actualidad, la cual es solventada hoy en día por aplicaciones de una índole más genérica o mediante varias aplicaciones que se complementan entre sí. Además, es destacable el hecho de que la información a compartir se pretende mostrar mayoritariamente a través de representaciones gráficas, consiguiendo una mayor sencillez en su interpretación.

Por consiguiente, a la hora de querer practicar ciclismo, el usuario es capaz de ahorrar tiempo en la planificación, permitiendo por lo tanto su práctica durante más tiempo. De manera adicional, se obtiene la información necesaria para realizar este deporte siendo conocedor de las condiciones y pudiendo evaluar el riesgo de su práctica.

## Estructura del trabajo

Mediante este proyecto se trabaja cada una de las fases del ciclo de vida de los datos. Se comienza con la investigación y el análisis de varias herramientas existentes, las cuales tienen cierto grado de similitud respecto al proyecto presentado. En esta misma sección se valoran todas las posibles tecnologías a utilizar para su desarrollo, justificando la elección de cada una de las utilizadas.

En la sección posterior, se define el objetivo principal y los objetivos secundarios del proyecto, englobando desde la exploración y recolección de datos hasta su visualización. Seguidamente, se presenta la metodología utilizada para la gestión del proyecto, con sus respectivas herramientas de gestión.

El grosso del proyecto se divide mediante subapartados, uno por cada fase del ciclo de vida del conjunto de datos: captura, limpieza y procesamiento, almacenamiento, análisis y visualización.

**Figura 1.** *Representación visual de las fases del ciclo de vida del dato.*

Diagrama

Descripción generada automáticamente

Fuente: Javier Esmerado, Medium.

Respecto a la imagen previa, la única diferencia existente es que las fases de almacenamiento y tratamiento están intercambiadas. Como se menciona en más de una ocasión a lo largo del documento. Este hecho se debe a la conformación de un Data Warehouse en vez de un Data Lake, almacenando únicamente los datos que son de interés en lugar de la totalidad de los mismos, habiendo sufrido una transformación previa a su almacenamiento.

Entrando en detalle, en la primera fase se comienza explicando la recogida de datos desde unas APIs públicas de Open Data Euskadi y Euskalmet, mediante las cuales se recogen valores meteorológicos y relacionados con el tráfico.

En la segunda fase se presenta el procesamiento y la limpieza de los datos recogidos. De todos los registros devueltos por las APIs, se seleccionan los que mayor relevancia tengan para la aplicación. Una vez seleccionados y previo a la inserción en la base de datos de MongoDB en local, se realiza la limpieza y el tratamiento de estos, con el fin de almacenar únicamente datos con valor y que por consiguiente generen un Data Warehouse.

En la fase de almacenamiento, se muestra el guardado de los elementos seleccionados y su estructuración en distintas colecciones, lo que al ser una base de datos no relacional simula las diferentes tablas de lo que sería una base de datos estructurada (SQL).

Para la cuarta fase, fase de análisis, se comienza realizando un análisis exploratorio de datos (EDA), con el fin de entender en mayor profundidad los valores que dichos datos pueden tomar. Tras realizar el análisis, se plantean diversas técnicas de IA para poder obtener predicciones de los diferentes indicadores meteorológicos y referentes al tráfico que podrían existir durante los próximos días. Para ello, se utilizan varios tipos de regresiones y redes neuronales. Se tienen en cuenta todos los históricos registrados en la base de datos, así como diferentes atributos que ponderan según su correlación y permiten realizar el entrenamiento óptimo para el modelo en cuestión.

Una vez obtenidas las predicciones, se inicia la fase de visualización. El desarrollo de esta fase se lleva a cabo utilizando herramientas de BI como Power BI. De esta forma, se pueden visualizar los datos que mayor valor aporten al proyecto. Los datos son ubicados en varios dashboards, de acuerdo con el área al que pertenezca cada dato: meteorología, calidad del aire, tráfico e incidencias.

Tras el desarrollo del proyecto, se detallan aspectos de una índole más genérica como son el plan de trabajo y gestión del proyecto. En este apartado, se desglosa la planificación realizada para poder cumplir con el proyecto además de presentar las diferentes limitaciones y problemáticas que han podido existir durante el transcurso del mismo.

Finalmente, se exponen los resultados y conclusiones obtenidas, permitiendo conocer el resultado final del proyecto. Se concluye con posibles líneas futuras de trabajo que tratar sobre el proyecto.

# Contexto y estado del arte

## Contexto del problema

Cada año, decenas de personas pierden la vida o sufren graves lesiones practicando ciclismo en las carreteras del País Vasco. Estas tragedias no solo representan pérdidas humanas, sino que evidencian la preocupante situación existente en cuanto a la seguridad sobre la bicicleta. Como varias noticias avalan, el número de accidentes de ciclistas en el País Vasco ha aumentado considerablemente en los últimos años (Press, 2023). Esto lo convierte en una situación insostenible, donde los datos registran que casi la mitad de los fallecidos en accidentes de carreteras vascas son ciclistas, motoristas o viandantes («45 personas murieron en las carreteras en Euskadi en 2023, casi la mitad motoristas, viandantes o ciclistas», 2024). Con estos datos, queda claro que se deben tomar varias medidas de concienciación y seguridad, para posibilitar la práctica de este deporte de forma segura, lo que en la actualidad no es un hecho.

El caso más reciente ocurrió apenas un año vista, con el fallecimiento de un joven ciclista de 19 años cerca de la capital vasca al ser arroyado por un vehículo (Eurosport, 2023). Aunque no todos los incidentes que ocurren se deben a esta causa, la inmensa mayoría de los accidentes se pueden agrupar en que han sido causados por el tráfico (Michael T. Gibson et al., 2020). No obstante, hay un segundo grupo que también recoge gran cantidad de accidentes ciclistas como son las circunstancias meteorológicas (Cesvimap, 2021).

Ante esta situación, una mayor prevención es innegociable, así como una mejor planificación de las rutas a realizar o del momento en el que llevarlas a cabo. Para ayudar en el objetivo de disminuir y quizás algún día erradicar estos accidentes, se propone este proyecto. Este, se puede entender como una herramienta adicional para aquellos ciclistas que quieran tomar un mayor control sobre las variables altamente complicadas de controlar que afectan durante el transcurso de esta actividad.

## Estado del arte de herramientas similares existentes

Al realizar un estudio de variantes similares al proyecto presentado, se han encontrado varias herramientas ya existentes que comparten algunas de las funcionalidades que se buscan obtener. No obstante, cabe mencionar que ninguna de las alternativas encontradas ofrece la visión completa que se pretende obtener con “Biker”. A continuación, se presentan las alternativas con mayor similitud y relevancia a la propuesta realizada.

### MeteoRuta

Trata de una colaboración entre la Dirección General de Tráfico (DGT) y la Agencia Estatal de Meteorología (AEMET). Dicha colaboración surgió a inicios del año 2014 y puso a disposición de los ciudadanos españoles una web en la que poder visualizar el estado de las carreteras en tiempo real (Agencia Estatal de Meteorología, s. f.). Esta web, salió a la luz como consecuencia del fuerte temporal ocurrido durante esas fechas («Mapa interactivo de la DGT y AEMET, esencial para planificar tu ruta cuando hace mal tiempo», 2023). Actualmente, esta web sigue en activo.

Esta herramienta se puede utilizar de diferentes formas, pudiendo analizar una ruta concreta o bien ir explorando zonas del mapa manualmente, conociendo información más específica a medida que se recorren las diferentes vías.

Una captura de pantalla de una computadora

Descripción generada automáticamente**Figura 2.** *Captura de la web MeteoRuta.*

Fuente: Elaboración propia.

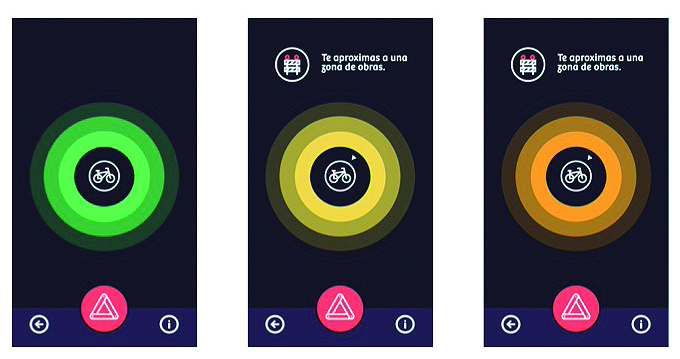
En lo que a la comparativa con “Biker” respecta, se puede considerar como la alternativa con mayor similitud encontrada, dado que engloba tanto un análisis meteorológico como datos de tráfico en una misma aplicación. Sin embargo, existen diferencias considerables como que el enfoque de MeteoRuta no está principalmente ligado con las bicicletas, que se engloba un territorio mucho mayor o que no se realiza un análisis tan detallado del tráfico como pretende realizar “Biker”. Además, otro aspecto a considerar es que MeteoRuta únicamente trabaja sobre una visualización de un mapa donde poder interactuar, mientras con “Biker” se compone de varios grafismos que aporten información variada.

### Comobity

Otra aplicación que puede tener relación en lo que al ámbito de movilidad se refiere, es la aplicación de Comobity («Comobity, la App “segura” para ciclistas», 2015). Comobity es una aplicación tanto para peatones, ciclistas y vehículos donde priorizan a las bicicletas y los peatones por delante de los vehículos de motor.

La finalidad principal de esta aplicación va dirigida hacia la búsqueda de una mayor seguridad para cualquiera de los individuos que transiten en estos tres medios de transporte. Para ello, el principio a seguir se basa en que cualquier individuo debe llevar la aplicación de Comobity abierta, pudiendo conformar así una red de usuarios con sus respectivas ubicaciones. De esta forma, ante potenciales incidentes como acercamientos imprudentes o zonas conflictivas, Comobity puede avisar al usuario de dicho riesgo. De igual forma, en caso de detectar alguna alteración en la ruta, como puede ser una zona de obras, se notificará al usuario a medida que se aproxime a la zona conflictiva.

En comparación con “Biker”, el análisis que Comobity realiza sobre el tráfico es bastante similar, a pesar de que es una herramienta pensada para ser utilizada en tiempo real. Sin embargo, la propuesta que se realiza en este documento tiene como finalidad ayudar en la planificación y que el usuario sea conocedor de las condiciones existentes previo al inicio de la práctica del deporte, por lo que no requiere la característica de actualización de datos inminente. Se ha de mencionar también, que Comobity no hace ninguna referencia al análisis meteorológico, lo que en “Biker” toma un importante peso.

**Figura 3.** *Captura de la aplicación Comobity.*

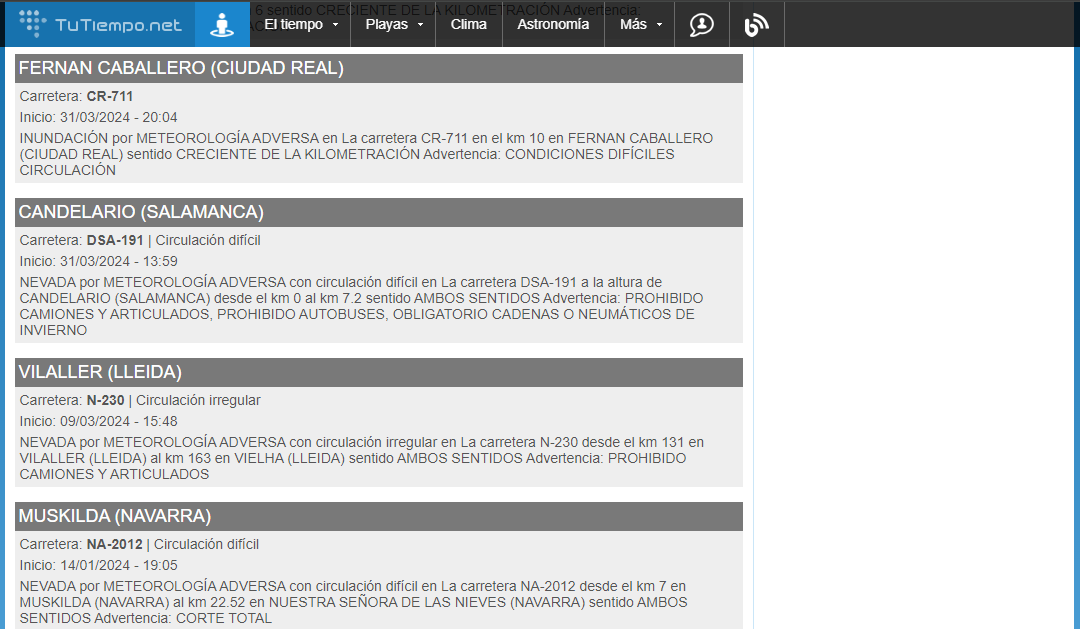
Fuente: Elaboración propia.

### TuTiempo.net

La web de TuTiempo.net (*Información meteorológica tráfico - Incidencias en carretera*, s. f.) es una herramienta que permite conocer tanto la información meteorológica como los incidentes ocurridos en la carretera en tiempo real. Por ello, es considerado una alternativa muy competitiva de no ser porque no dispone de una representación gráfica de dichos datos. El registro de incidencias en las carreteras, por ejemplo, se presenta mediante entradas de texto, lo que dificulta considerablemente el poder detectar la localización de una incidencia en una cierta zona geográfica de un rápido vistazo. Es por ello que la gran diferencia entre ambas herramientas viene en la representación utilizada para mostrar los datos.

Por otro lado, como ha ocurrido con otras herramientas previamente analizadas, mencionar que en “Biker” se espera obtener un análisis sobre el tráfico con mayor detalle, pudiendo ofrecer información clave para el ámbito del ciclismo como es la densidad del tráfico.

**Figura 4.** *Captura de la web TuTiempo.net.*



Fuente: Elaboración propia.

### Flare

Por último, se presenta la aplicación de Flare («Lanzan una app para detectar automáticamente los accident...», 2022), desarrollada por la Federación Internacional del Automóvil (FIA). Esta aplicación notifica en tiempo real los accidentes sufridos por los ciclistas, permitiendo así una asistencia sanitaria en menor tiempo y compartiendo automáticamente la geolocalización del dispositivo.

En comparación con “Biker”, uno de los objetivos finales que ambas soluciones comparten es que pretenden mejorar la seguridad de los ciclistas. No obstante, más allá del ámbito que tratan, no tienen similitud en ninguna de las funcionalidades que ofrecen, ya que una de ellas está destinada al rescate en caso de sufrir un accidente mientras que la otra pretende mejorar la planificación previa a la práctica del deporte.

A continuación, se puede apreciar una tabla comparativa entre las herramientas presentadas a lo largo de la sección.

Tabla 1. *Tabla comparativa de las herramientas analizadas.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Herramienta | MeteoRuta | Comobity | TuTiempo.net | Flare |
| Finalidad principal | Informar | Proteger | Informar | Socorrer |
| Año de lanzamiento | 2014 | 2015 | 2005 | 2022 |
| Alcance | España | España | España | - |
| Información meteorológica | Sí | No | Sí | No |
| Información del tráfico | Sí | Sí | Sí | No |
| Visualización gráfica | Sí | Sí | No | Sí |
| Aplicación móvil | App de AEMET | Sí | Sí | Sí |
| Activa | Sí | Sí | Sí | Sí |

Fuente: Elaboración propia.

## Estado del arte de las tecnologías a utilizar

Una vez realizado el análisis sobre las herramientas existentes, se investiga y justifica el uso de cada tecnología utilizadas en este proyecto. En dicho análisis, se realiza una comparativa entre las principales opciones a utilizar en cada fase del ciclo de vida del dato y se detalla la elección de la herramienta a utilizar.

### Recogida de datos

La recogida de los datos es una fase crítica en el ciclo de vida de los datos. Esto se debe a que, si no existen unos registros adecuados para su análisis, difícilmente se obtendrán unos resultados útiles. Por ello, se presentan diferentes opciones para realizar la recogida de datos, con las ventajas e inconvenientes de cada posibilidad.

#### APIs

Permite recoger los valores más recientes de forma automática sin tener la necesidad de reemplazar el origen de los datos (Castro, 2022). Esto es posible porque se trabaja sobre unos datos dinámicos, permitiendo conformar un flujo de datos actualizado a través de una recogida continua.

Aditivamente, es destacable el hecho de que recogiendo los datos mediante APIs, se obtiene una mayor libertad para elegir los elementos con los que se desea trabajar, seleccionando únicamente los campos de interés. De esta forma se puede permutar el orden en la que se lleve a cabo la fase de almacenamiento y la fase de la limpieza y procesamiento, lo que permite conformar un Data Warehouse en lugar de un Data Lake.

Un punto a tener en cuenta es que existen infinidad de APIs, teniendo cada una sus propias parametrizaciones y llegando a requerir claves para su uso en algunos casos.

#### Repositorios

Son conjuntos de datos estáticos sobre los cuales trabajar (Tehreem Naeem, 2020). Generalmente suelen encontrarse en archivos de texto con diferentes formatos, entre los que pueden ser comunes los archivos Comma Separated Value (CSV), Extensible Markup Language (XML) o JavaScript Object Notation (JSON).

#### Web Scraping

Técnica utilizada para obtener datos de las páginas web de forma indirecta (Datademia, 2021). El funcionamiento de esta técnica consiste en recoger todo el contenido de la web en la que se hallan los datos, y posteriormente distinguir la localización de los mismos dentro de toda la estructura HyperText Markup Language (HTML) de dicha web.

#### Formularios y encuestas

Los formularios y las encuestas son técnicas mediante las cuales se obtiene información de una muestra de la población que interesa conocer (Ortega, 2019). Esta herramienta posibilita su uso tanto de forma tradicional como online. Con la digitalización cada vez más presente, la versión online aumenta su índice de uso, ya no solo por la facilidad de recoger los datos de manera digital si no por la mayor publicidad y participación que se consigue obtener.

De cara a analizar los datos que se recojan, destacar la importancia del tipo de pregunta que se realice en la encuesta o formulario en cuestión. Si se presenta una pregunta con respuesta abierta, a la hora de tratar los datos puede existir pérdida de información. Por el contrario, si se proponen preguntas con respuesta cerrada, los datos son fácilmente analizables sin tener ninguna perdida en la información recogida. Por ello, este último es el principal tipo de preguntas que se recomienda realizar en formularios y encuestas.

### Limpieza y procesamiento de datos

Tanto la fase de limpieza y procesamiento de datos, así como la de almacenamiento, se encuentran estrechamente ligadas. Esto se debe a que, en el ciclo de vida de un conjunto de datos, en muchas de ocasiones se alterna el orden de estas fases, dependiendo de si se aplica un ETL (Extract Transform Load) o ELT (Extract Load Trasnform). En el primer caso, se conforma un Data Warehouse, almacenando únicamente la información relevante mientras que en la segunda opción se crea un Data Lake.

En el caso de “Biker”, se plantean las dos opciones para valorar sus puntos positivos y negativos, teniendo siempre en mente el objetivo general de completar el ciclo de vida de los datos previamente recogidos. Seguidamente, se exponen sus diferencias y se valoran ambas alternativas.

#### Data Lake

Consiste en una estructura donde se permiten almacenar grandes cantidades de datos en su formato de origen, sin ningún procesamiento previo y para almacenar durante una gran cantidad de tiempo (*¿Qué son los lagos de datos?*, s. f.). Estos valores pueden ser estructurados, semiestructurados o no estructurados, admitiendo todo tipo de formatos dentro de dicho repositorio (*Soluciones de data lake | IBM*, s. f.).

Esta es justamente una de las principales funcionalidades de los Data Lake, ya que no es fácil encontrar configuraciones que permitan el almacenar distintos formatos de datos en una misma ubicación, y, además, durante un intervalo tan amplio en el tiempo (Navarro, 2022).

#### Data Warehouse

Consiste en una estructura donde se permiten almacenar grandes cantidades de datos tras ser analizados, transformados y modificados (*¿Qué son los almacenes de datos?*, s. f.). Por ello, únicamente se pueden almacenar elementos estructurados o semiestructurados.

La finalidad principal de estos repositorios es el almacenar información para su posterior explotación por parte de la entidad propietaria, generalmente empresas. Consecuentemente, tienen una gran velocidad en el acceso a los datos, siendo dicho acceso fácilmente realizable (Daniel, 2022).

### Almacenamiento de datos

Existen diferentes posibilidades para almacenar los datos, siendo las bases de datos la opción más habitual. No obstante, existen varias alternativas dentro de esta solución, las cuales son comentadas a continuación.

#### Base de datos relacionales

Las bases de datos relacionales son un tipo de bases de datos que toma como base las relaciones entre los distintos grupos de datos que se conforman. Estos grupos se conocen como tablas, las cuales se relacionan entre sí y conforman un modelo relacional, el pilar de la base de datos (*¿Qué es una base de datos relacional?*, s. f.).

El principal beneficio de este modelo es que el fundamento sobre el cual se construye es muy básico, pudiendo complicarse a medida que se escale la dimensión de los datos. Complementariamente, la alta usabilidad existente con este tipo de bases de datos y sus propiedades ACID la hacen ser una opción perfecta para almacenar conjuntos de datos estructurados (*¿Qué es una base de datos relacional?*, s. f.).

Algunas de las bases de datos relacionales más comunes son:

* MySQL (*MySQL*, s. f.).
* PostgreSQL (PostgreSQL Global Development Group, 2024).
* Microsot SQL Server (*SQL Server 2022 | Microsoft*, s. f.).
* Oracle Database (*Base de datos rentable y de alto rendimiento*, s. f.).

#### Base de datos no relacionales

Las bases de datos no relacionales son un tipo de bases de datos que están diseñadas para ofrecer una alta flexibilidad y escalabilidad en los modelos de datos (*Bases de datos no relacionales | Bases de datos de gráficos | AWS*, s. f.).

A diferencia de las bases de datos relacionales, los elementos que se guardan en ellas no tienen una estructura predefinida. Consecuentemente, existen varios tipos de bases de datos no relacionales, teniendo cada una de ellas sus virtudes y limitaciones. Entre las principales, se pueden encontrar bases de datos de documentos, de clave-valor y de grafos (*¿Qué es una base de datos NoSQL?*, 2024).

Las principales bases de datos no relacionales son:

* MongoDB (*MongoDB*, s. f.)
* Cassandra (*Apache Cassandra | Apache Cassandra Documentation*, s. f.)
* Redis (*Redis - La plataforma de datos en tiempo real*, s. f.)
* Neo4j (*Neo4j Graph Database & Analytics – The Leader in Graph Databases*, s. f.).

### Análisis de datos

La fase de análisis de datos se compone por los dos tipos principales de algoritmos. Tanto los algoritmos de aprendizaje supervisado como los algoritmos de aprendizaje no supervisado ofrecen diferentes posibilidades de cara a obtener información de utilidad para el usuario final.

Por ello, con el fin de determinar la utilidad de estos algoritmos en el proyecto de “Biker”, es necesario valorar en detalle varias de sus propiedades. La finalidad, las características del conjunto de datos que requieren y los parámetros de configuración son algunos de los muchos aspectos a comparar entre ellos. Por ello, se presentan ambos conjuntos.

#### Algoritmos de aprendizaje supervisado

Son algoritmos que tratan con registros etiquetados. Estos algoritmos utilizan las etiquetas de los datos para entrenarse con el fin de realizar una correcta clasificación o predicción (*¿Qué es el aprendizaje supervisado?*, 2024).

Para ello, se requiere dividir el conjunto de datos en dos grupos. El primer grupo es el de los datos de entrenamiento. Este conjunto es aplicado al algoritmo como base, definiendo las entradas y sus correspondientes salidas. De esta forma, el modelo obtiene un pequeño conocimiento de cómo actuar. El segundo grupo lo conforman los valores de prueba, unos datos que son totalmente independientes de los datos de entrenamiento y que permiten comprobar el funcionamiento real del algoritmo una vez entrenado.

Para evaluar los resultados obtenidos, se utilizan diferentes métricas como la precisión o la pérdida. En caso de que la precisión obtenida no sea la esperada, existen metodologías para intentar mejorarla. La validación cruzada (Cross-Validation) precisamente, consiste en realizar varias iteraciones de entrenamiento al algoritmo, cambiando los datos que componen los conjuntos de entrenamiento y prueba. De esta forma se entrena con un mayor número de datos que ayuda a obtener un mejor entrenamiento (Na8, 2023).

Entre los algoritmos de aprendizaje supervisado se encuentran los siguientes:

* Regresión lineal: el objetivo principal es poder realizar la predicción de valores continuos. Se predice el valor de la variable dependiente (output) a través de la relación existente con las variables independientes (inputs), generando una recta entre 2 puntos que interprete de la mejor forma posible los datos de entrenamiento. De esta forma, al evaluar el modelo con los datos de validación, se obtiene un rendimiento similar.
* Regresión logística: el objetivo principal es poder realizar la predicción de valores categóricos, ocurriendo con cierta frecuencia clasificaciones binarias o multiclase. A diferencia de las regresiones lineales, en este tipo se predice la probabilidad de ocurrencia, obteniendo consecuentemente valores entre 0 y 1. Otra diferencia respecto a la regresión lineal es que, en este caso, la recta que se conforma puede tener curvas.
* Máquinas de soporte vectorial (SVM): el objetivo principal es poder realizar clasificaciones y predicciones de los datos, diferenciando la mayor cantidad de clases posibles. Para ello se define un hiperplano, el cual separa las clases independientemente de las dimensiones que lo conformen.
* Árboles de decisión: el objetivo principal es poder solucionar problemas de clasificación y predicción de los datos. No obstante, son herramientas muy versátiles y que permiten una amplia usabilidad. El uso de estos algoritmos se basa en la división continua del conjunto de datos, hasta alcanzar el criterio de parada definido.
* Redes neuronales: herramienta con una amplia versatilidad que simula el funcionamiento de un cerebro humano, permitiendo un amplio abanico de opciones entre las que destacan principalmente la clasificación y la regresión.

La base de esta herramienta se fundamenta en las neuronas artificiales y las capas que la conforman, así como en las conexiones ponderadas y funciones de activación.

* Naive Bayes: algoritmo sencillo, pero realmente eficaz, con el objetivo principal de poder solucionar problemas de clasificación de variables categóricas. Este algoritmo se basa en el teorema de Bayes (López, 2018).

#### Algoritmos de aprendizaje no supervisado

Son algoritmos que buscan agrupar los datos según propiedades o patrones que el propio algoritmo detecte, no teniendo ninguna etiqueta, a diferencia de los algoritmos de aprendizaje supervisado (*¿Qué es el aprendizaje no supervisado?*, 2023).

Dentro de los algoritmos de aprendizaje no supervisado, las agrupaciones de mayor relevancia son:

* Clustering exclusivo: K-Means.

El objetivo principal del algoritmo K-Means es poder dividir el conjunto de datos en K grupos, intentando minimizar la suma de los cuadrados de las distancias dentro de cada clúster. Para ello se definen los centroides y se itera el cálculo de la distancia existente desde cada dato hasta los centroides. Dicho dato es asignado al clúster con el centroide más cercano.

* Clustering jerárquico: agrupamiento jerárquico aglomerativo.

El objetivo principal del agrupamiento jerárquico aglomerativo es agrupar conjuntos de datos de forma jerárquica. El escenario inicial se sitúa con varios clústeres, los cuales se fusionan hasta converger en un único clúster. El proceso de fusionado de los clústeres está ligado a la distancia existente entre los mismos, agrupando de los clústeres existentes los que mayor similitud tengan.

* Clustering probabilista: Expectation-Maximization (EM).

El objetivo principal del EM se basa en estimar los parámetros incompletos de los modelos estadísticos. Dicho algoritmo se compone de dos fases (expectación y maximización) las cuales se llevan a cabo de forma iterativa. La primera busca obtener una primera aproximación de los valores, siendo dichos valores optimizados por la segunda. Este proceso se realiza durante varias iteraciones hasta que el valor deje de ser cambiante y se estabilice.

* Clustering difuso: Fuzzy C-Means.

El objetivo principal del algoritmo Fuzzy C-Means es poder dividir el conjunto de datos en varios grupos, al igual que realiza el algoritmo K-Means. La principal diferencia respecto a este algoritmo se encuentra en que K-Means únicamente permite asignar un dato a un clúster, mientras que Fuzzy C-Means permite que un dato pertenezca a varios clústeres. Este algoritmo sigue un proceso prácticamente idéntico al algoritmo K-Means, siendo la diferencia más notoria que la asignación de un dato a un clúster se realiza en base a un grado de pertenencia. De esta forma, un único dato puede pertenecer a varios clústeres distintos.

### Visualización de datos

En la fase de visualización de los datos existen infinidad de opciones sobre como mostrar los resultados obtenidos. De igual forma, existen millones de herramientas con las cuales conformar estas visualizaciones. Se valoran seguidamente algunas de ellas.

#### Visualizaciones con Python

Python es un lenguaje de programación ampliamente conocido y utilizado, teniendo un gran crecimiento en los últimos años. Esto se debe a su amplia funcionalidad, así como su facilidad tanto para el aprendizaje como en su uso. A más, existen abundancia de librerías a importar que permiten realizar infinidad de tareas como es el caso de la visualización de datos (Zapata, 2024).

**Gráfico, Gráfico de barras

Descripción generada automáticamenteFigura 5.** *Ejemplo de visualización con librerías de Python.*

Fuente: Elaboración propia.

De las librerías más relevantes dentro de este lenguaje de programación destacan:

* Matplotlib
* Seaborn
* Plotly

#### Visualizaciones con Data-Driven Documents (D3)

D3 es una biblioteca de JavaScript (JS) que permite tratar datos integrándolos en visualizaciones interactivas. Dicha herramienta se compone de infinidad de soluciones pudiendo hasta generar gráficos que deriven de un subconjunto de datos. Además, es una solución que destaca por su flexibilidad, así como por su fácil integración en las páginas web (*¿Qué es D3?*, s. f.).

**Figura 6.** *Ejemplo de visualización con D3.*

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Fuente: Elaboración propia.

#### Visualizaciones con herramientas de BI

Por último, se explora la alternativa de utilizar herramientas de BI, las cuales además de permitir la visualización de los gráficos permite el análisis previo y posterior. Una de las principales ventajas de estas aplicaciones es la alta personalización que ofrece al usuario, difícilmente existiendo dos dashboards iguales (Analytics, 2023). Conociendo su utilidad, se presentan las opciones más relevantes en lo que a herramientas BI se refiere:

* Power BI: herramienta novedosa que ofrece un fácil uso y permite graficar los datos que han podido ser previamente tratados. Ideal para la generación de informes y dashboards de una forma sencilla pero eficaz (*Power BI*, s. f.).
* Tableau: herramienta globalmente conocida y líder absoluta hasta la aparición de Power BI (*Software de análisis e inteligencia de negocios | Tableau*, s. f.). Se requiere de una mayor formación en la herramienta, a pesar de que es intuitiva. En cuanto a funcionalidades, muy similar a Power BI, permitiendo también el tratamiento de datos.
* QlikView: herramienta con cierta similitud a las dos previamente explicadas, pero con un tratamiento de los datos bastante más limitado. Se basa en una exploración más asociativa a pesar de tener una interfaz menos intuitiva que Tableau o Power BI (*QlikView – Analítica y cuadros de mando interactivos y eficaces | Qlik*, s. f.).
* SAP BI: herramienta del Enterprise Resource Planning (ERP) que le añade la funcionalidad especifica de la inteligencia de negocio (*SAP*, s. f.). Las principales diferencias son la integración en el entorno de desarrollo de programas de sistemas de análisis (SAP), el mayor requerimiento de conocimiento para su uso y que en cuanto a coste, ya que requiere una mayor inversión que otras alternativas.

Gráfico, Aplicación

Descripción generada automáticamente**Figura 7.** *Ejemplo de visualización con Power BI.*

Fuente: Elaboración propia.

#### Comparativa de herramientas de visualización

Tabla 2. *Tabla comparativa de herramientas de visualización.*

|  |  |  |
| --- | --- | --- |
| Herramienta | Ventajas | Desventajas |
| Visualizaciones con Python | * Gran personalización a través de diferentes librerías. * Amplia documentación. * Opensource. | * Curva de aprendizaje pronunciada. * Conocimientos de Python. |
| Visualizaciones con D3 | * Altamente personalizable. * Visualizaciones interactivas. * Amplia documentación. * Opensource. | * Curva de aprendizaje muy pronunciada. * Conocimientos de JS. |
| Power BI | * Fácil uso. * Amplia conectividad. * Destinado específicamente para el análisis de datos. * Visualizaciones interactivas. * Amplia documentación. * Versión gratuita existente. | * Menor personalización. * Algunas visualizaciones requieren conexión a internet. |
| Tableau | * Visualizaciones interactivas. * Destinado específicamente para el análisis de datos. * Amplia documentación. * Versión gratuita muy limitada. | * Coste elevado mediante licencias. * Lentitud con grandes conjuntos de datos. * Necesaria cierta adaptación al entorno. |
| QlikView | * Destinado específicamente para el análisis de datos. * Gran personalización. | * Curva de aprendizaje pronunciada. * Coste elevado. |
| SAP BI | * Integración con SAP. * Soporte y actualizaciones. * Destinado específicamente para el análisis de datos. | * Coste elevado. * Requiere gran capacidad de recursos. * Destinado a grandes empresas. |

Fuente: Elaboración propia.

## Conclusiones

Una vez realizado el análisis, se puede afirmar que ninguna de las herramientas presentadas coincide con todos los pilares que conforman dicho proyecto. No obstante, estas opciones pueden llegar a tener un uso similar, a pesar de que su función principal no sea esta.

Los pilares mencionados sobre los que se basa “Biker” se pueden agrupar en un análisis del tráfico y un análisis meteorológico. El primero de estos mide la afluencia de vehículos en las carreteras, incidentes existentes en las mismas y otras propiedades de las vías que puedan afectar al practicante durante el desarrollo de la actividad. En este análisis, se espera abarcar un mayor nivel de detalle que las herramientas competidoras, llegando al punto de facilitar datos muy específicos como la densidad del tráfico esperada.

Por su parte, el análisis meteorológico incluye una serie de mediciones de diferentes índices meteorológicos, como puede ser el viento, la lluvia y la temperatura, permitiendo la planificación y valoración de si es adecuada la práctica de este deporte en ese instante. En este caso, sí que existen diversas herramientas que se pueden aproximar más al nivel de detalle de este ámbito, ya que la fortaleza principal de “Biker” consiste en el análisis conjunto realizado entre ambas agrupaciones de datos.

De todas las opciones valoradas en las diversas fases del ciclo de vida del dato, las elecciones tomadas son presentadas a continuación junto a su respectiva justificación.

En la fase inicial del proceso, se ha decidido utilizar la recolección de datos a través de APIs, ya que se cree que puede ser una solución que aporte grandes beneficios en su uso posterior. En este proyecto se han utilizado APIs más sencillas de integrar y otras más complejas, que han requerido de claves públicas y privadas. Para estas últimas, ha sido necesario realizar una investigación más exhaustiva, dado la pobre documentación que existía al respecto.

Como se puede intuir al ser la segunda etapa del proceso, en la fase de limpieza y procesamiento de los datos, se utiliza un Data Warehouse. Mediante esta estructura, se entiende que existe un procesando y una recogida selectiva de los valores obtenidos desde las APIs, siendo únicamente aquellos elementos considerados de utilidad los que se almacenan en la siguiente fase. Dicho procesamiento y selección es realizado con Python.

Esta elección se toma mayoritariamente por la limitación tecnológica existente en los equipos con los que se desarrolla este proyecto. Al realizar primero una limpieza, se consiguen mover menos datos a la base de datos, siendo aquellos almacenados indispensables para el análisis posterior. Consecuentemente existe un considerable ahorro de tiempo y de capacidad, permitiendo obtener un resultado más eficiente con los medios de los que se disponen.

Para la fase de almacenamiento, se ha seleccionado una base de datos no relacional como es MongoDB dado que en ciertas ocasiones los registros recogidos dentro de una misma colección pueden diferir en formato. Aun así, se podría haber llegado a utilizar una base de datos relacional, ya que la mayoría de los elementos recogidos son procedentes de sensores y se consideran datos estructurados o semiestructurados. No obstante, dentro de la amplia variedad de bases de datos existentes, se ha decidido que MongoDB es la mejor alternativa, tanto por la mayor velocidad en el acceso a los registros, así como por la experiencia previa existente con la herramienta.

Adicionalmente, con el análisis realizado se ha podido conocer que MongoDB dispone de un alto rendimiento gracias a los índices que utiliza y el almacenamiento en memoria del que dispone. Sumado a su flexibilidad, escalabilidad y soporte, lo convierten en una herramienta con un alto potencial, superando en varios puntos a otras herramientas similares, lo que la convierte en la elección para este proyecto.

En la fase de análisis de los datos, se ha decidido utilizar mayoritariamente algoritmos de aprendizaje supervisado. El uso de estos algoritmos ha sido argumentado por la finalidad principal de predecir valores, siendo en la mayoría de los casos índices relacionados tanto con la meteorología como con el tráfico. No obstante, también se ha utilizado el clustering para ciertos conjuntos de datos.

Por último, en la etapa de visualización de datos se ha decidido utilizar Power BI ya que es una herramienta que está en auge y sobre la que cada vez existe una mayor demanda. Al ser una herramienta que dispone de una amplia capa de conectividad con diversas herramientas y al facilitar una versión gratuita, ha sido fácil encontrar soporte sobre la misma. Adicionalmente, era una herramienta que se quería aprender a nivel personal por lo que se ha encontrado el momento idóneo a través de este proyecto.

# Objetivos concretos y alcance

El siguiente apartado presenta los objetivos que se pretenden logran en este Trabajo Fin de Máster (TFM), siendo diferenciado el objetivo principal de los objetivos secundarios.

## Objetivo general

El objetivo principal de este proyecto es diseñar, desarrollar y evaluar una aplicación para ciclistas en el País Vasco que gestione el ciclo de vida completo de datos meteorológicos y referentes al tráfico. Se busca mediante la información facilitada, que los ciclistas sean capaces de tomar decisiones fundamentadas, pudiendo conocer aspectos tan desconocidos como el clima o estado de las carreteras, de forma previa a iniciar la ruta.

Para ello se requiere del ciclo de vida al completo, el cual debe incluir desde la recogida hasta su visualización, pasando por las fases intermedias de limpieza y procesamiento, almacenamiento y análisis.

## Objetivos específicos

Además de cumplir el objetivo principal, el proyecto pretende completar los denominados como objetivos secundarios, los cuales se mencionan a continuación:

* Obtener datos de 2 orígenes diferentes.
* Filtrar los datos útiles antes de ser almacenados, conformando un ETL.
* Almacenar los datos de una forma semiestructurada en la base de datos.
* Obtener un conjunto de datos mayor que 80.000 instancias.
* Utilizar 2 algoritmos de aprendizaje supervisado.
* Utilizar 1 algoritmo de aprendizaje no supervisado.
* Utilizar técnicas de validación cruzada de modelos en los algoritmos utilizados.
* Definir los hiperparámetros de forma óptima para evitar el sobreajuste (overfitting).
* Desarrollar y evaluar la predicción de los principales índices meteorológicos a través de algoritmos de aprendizaje supervisado con un R2-Score superior a 0,8.
* Desarrollar y evaluar la predicción de los principales índices de la calidad del aire a través de algoritmos de aprendizaje supervisado con un R2-Score superior a 0,5.
* Obtener visualizaciones interactivas a través de una herramienta de BI.
* Utilizar 8 tipos de visualizaciones en los informes.

La visión con la que se establecen estos objetivos es la de obtener un proceso completo y detallado sobre la gestión del ciclo de vida de un conjunto de datos. En este caso, el ámbito a tratar es el ciclismo, pudiendo en un futuro transversalizarse a cualquier otro campo del que se dispongan datos.

Asimismo, se pretende corroborar el cumplimiento general de este proyecto a través del cumplimiento individual de cada uno de los objetivos secundarios, siendo pequeñas muestras del cómputo global del proyecto.

## Alcance

Este proyecto tiene como alcance el desarrollo de una aplicación para ciclistas que integre el ciclo de vida completo de un conjunto de datos, desde la recogida de datos hasta su visualización. Sobre este conjunto de datos se pretende llevar a cabo tanto un análisis meteorológico, así como un análisis sobre las carreteras del País Vasco.

En este proyecto se incluye:

1. El diseño del proceso que engloba el ciclo de vida completo del conjunto de datos utilizado.
2. La recogida de los datos desde las APIs de Open Data Euskadi y Euskalmet. Adicionalmente se llevan a cabo una serie de pruebas a través de herramientas externas como Postman para asegurar el correcto funcionamiento de las APIs, al igual que se integra el Bearer Token en los casos que se requiere.
3. El tratamiento y limpieza de datos previo a la inserción en la base de datos. Se manipulan los registros con el fin de prepararlos y convertirlos en datos semiestructurados.
4. El almacenamiento de los datos en una base de datos. Únicamente serán almacenados aquellos elementos que se consideren relevantes, los cuales han sido tratados y estructurados en la fase anterior.
5. Un análisis de datos sobre las técnicas de IA a utilizar, valorando las diferentes propiedades de cada algoritmo y realizando una comparativa entre ellos. El desarrollo de todos los modelos es realizado para su evaluación, tomando posteriormente la decisión sobre que algoritmo se utiliza.
6. La visualización de los datos a través de diferentes informes que permiten al usuario conocer las circunstancias meteorológicas, así como las referentes al tráfico.
7. La documentación del proyecto, incluyendo los resultados de este y la aportación realizada.

# Metodología de trabajo y arquitectura

## Metodología de trabajo

La metodología del trabajo principal en este proyecto sigue los principios de la metodología ágil Scrum, siendo adaptada para ser utilizada con un único desarrollador. Cada uno de los sprints que conforma el proyecto son organizados inicialmente en base al Big Data Lifecycle Managment (Esmerado, 2024, p. 5). Este modelo resume el uso completo de los datos en proyectos Big Data en 5 fases principales, a pesar de que en ciertas ocasiones aparezcan fases adicionales. No obstante, al ser una metodología ágil pueden existir modificaciones en la organización de los sprints durante el transcurso del proyecto, requiriendo dividir alguna fase en varios sprints.

Para gestionar esta metodología se utilizan varias herramientas entre las cuales destaca Github para la gestión de versiones de código. Github es un portal web que funciona como repositorio distribuido utilizando la tecnología Git (Fernández, 2019). Actualmente, esta plataforma es mundialmente conocida y existen millones de usuarios en ella. El uso que se le atribuye a esta herramienta básicamente tiene como objetivo el almacenamiento de la última versión del software, existiendo un registro de las modificaciones realizadas y un control de sus versiones (Casero, 2023). Al ser un proyecto personal, no se utilizan todas las funcionalidades que esta tecnología puede aportar, ya que no es necesario. No obstante, en otras situaciones en la que se involucren más participantes, podría ser interesante utilizar funcionalidades adicionales de esta tecnología como la creación de ramas, destinada principalmente para el desarrollo colaborativo.

Otra herramienta que se utiliza, principalmente para tener una visión más general del proyecto, es GanttProject. Mediante la versión gratuita, se puede realizar la planificación general del proyecto, definiendo las diferentes tareas mediante un cronograma con sus respectivos plazos y recursos. GanttProject permite tener un enfoque más genérico del proyecto, permitiendo su gestión de una forma más organizada y consecuentemente eficiente (*GanttProject*, 2024).

## Arquitectura

En el esquema siguiente se puede apreciar la arquitectura que define las fases de la solución presentada. Estas fases engloban el ciclo de vida completo de un conjunto de datos, pudiendo apreciarse adicionalmente la tecnología utilizada para llevar a cabo dicha fase. En siguientes secciones se desglosa en mayor detalle.

**Figura 8.** *Arquitectura de las fases del ciclo de vida del dato.*

Diagrama

Descripción generada automáticamente con confianza media

Fuente: Elaboración propia.

La primera fase comienza con la obtención de los datos desde APIs públicas, tanto de Open Data Euskadi como de Euskalmet. Esta recogida de datos se ha realizado mediante librerías de Python y los valores devueltos desde las APIs han sido almacenados en variables de dicho lenguaje de programación. Los datos recogidos están relacionados con la meteorología y el tráfico del País Vasco.

La segunda fase consiste en recoger esas variables que contienen la totalidad de los datos y aplicarles una limpieza y tratamiento. Posteriormente, se realiza la selección únicamente de los datos que se crean útiles para el proyecto. Esta fase también es realizada con el lenguaje de programación Python.

La fase de almacenamiento consiste únicamente en almacenar los datos obtenidos tras la limpieza y el procesamiento realizado. Mediante librerías de Python se crea el enlace con MongoDB y posteriormente se añaden los datos tratados a la base de datos.

En la cuarta fase se comienza realizando un EDA a los datos hábiles con Python. Seguidamente se aplican técnicas de IA gracias a librerías más específicas de este mismo lenguaje. Estas técnicas son utilizadas para obtener una información que aporte un mayor valor añadido al análisis realizado, permitiendo obtener información valiosa para la posterior visualización.

En la última fase, la fase de visualización, se recogen los datos tras su análisis y se plasman en un dashboard mediante la herramienta de Power BI. Gracias a los distintos tipos de visualizaciones dentro de este dashboard, se aporta información variada y relevante al usuario final, la cual es de gran utilidad para llevar a cabo la planificación del deporte.

# Marco normativo

Como se ha mencionado a lo largo del escrito, la mayoría de los datos recogidos desde las diferentes APIs son de un carácter público, lo que no exime su uso en este proyecto. No obstante, el resto de datos se ha corroborado que se rigen bajo la licencia Creative Commons (CC), lo que posibilita su uso con fines académicos (*Información legal*, 2010). Es necesario aclarar que este conjunto de datos no es utilizado para ninguna otra finalidad más allá de este proyecto educativo, lo que se posibilita tanto por las licencias que poseen como por las normativas que se describen a continuación.

Otro punto de alta relevancia en el uso de este conjunto de datos es que en la totalidad del proyecto se utilizan datos anónimos por lo que no se requiere la aplicación de la normativa de datos. No obstante, se cumplen las dos principales normas en lo que a protección de datos se refiere. La primera de estas es la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD) (Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, 2018).

El Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos es considerada como la segunda norma a respetar para cumplir con la legislatura en torno a la protección de datos (RGPD) (BOE.es - DOUE-L-2016-80807 Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)., 2018).

Por ello, como se puede conocer en la página oficial de Open Data Euskadi, los datos reutilizados de la propia web son aptos en su totalidad para fines como el empleado en este proyecto, permitiendo su recogida, tratamiento, almacenamiento, análisis y visualización (*Normativa*, 2017).

# Desarrollo específico de la contribución

El ciclo de vida de un conjunto de datos hace referencia a una secuencia de etapas por las que pasan los datos a lo largo de su vida útil. Este proceso engloba desde la generación o recogida hasta la visualización y publicación de estos. Además, son varias las razones que hacen de este proceso una solución altamente relevante, destacando sobre todo en el ámbito empresarial.

La primera razón, que a su vez es la más destacable, se resume en que conforma un procedimiento estandarizado para la gestión eficaz de la información. De esta forma, se obtiene el máximo rendimiento a la explotación de los datos, pudiendo transversalizar el proceso a cualquiera de las situaciones posibles, independientemente del ámbito al que pertenezcan los datos. Además, se garantiza la regulación, así como la seguridad de los datos con los que se trata.

Otra ventaja ciertamente relevante es la amplia disponibilidad de los datos. Mediante este método, se posibilita la obtención de los datos de una forma mucho más veloz, conociendo la distribución en la que se almacenan los elementos, así como la información que reportan. Haciendo mención del punto anterior, se enfatiza la integridad estructural conformada a través de este proceso. Con una estructura claramente definida y las medidas de seguridad pertinentes, se brinda la posibilidad de tomar decisiones de una forma mucho más eficiente.

Por todas las ventajas mencionadas, así como las facilidades a más que puede aportar este proceso, el ciclo de vida de un conjunto de datos es considerado una herramienta con una utilidad inmensa, siendo mayor aun si cabe el potencial del que dispone.

En esta sección se explica el desarrollo realizado en este proyecto, analizando cada fase en detalle y mostrando el valor aportado.

## Recolección de los datos

Tras identificar un boceto con la idea del proyecto, el primer aspecto a tratar ha sido el encontrar un origen del cual obtener los datos para llevar a cabo dicha propuesta. Gracias a otros proyectos previos ya realizados, se conocía el repositorio de datos público de Open Data Euskadi. Además, como consecuencia de la investigación realizada para el estado del arte, se han podido valorar distintas alternativas tanto en el formato de los datos, como en el origen de los mismos.

Finalmente, se ha decidido recoger los datos mediante APIs no solo de Open Data Euskadi, sino que también de Euskalmet, la agencia de meteorología más conocida en el País Vasco. Ambas organizaciones poseen una alta confiabilidad en los datos que facilitan, ya que son entidades oficiales descendentes del Gobierno Vasco. Consecuentemente, están recogidas en la página oficial del Gobierno de España, bajo la supervisión del Ministerio para la transformación digital y de la función pública (*Open Data Euskadi | datos.gob.es*, s. f.).

Mediante estas APIs se han recogido registros referentes al tráfico y a la meteorología del País Vasco, permitiendo conformar la base del proyecto “Biker”. No obstante, la obtención de estos datos no ha sido únicamente mediante la ejecución de un código Python, sino que ha seguido una metodología con sus respectivas fases.

La fase inicial de este procedimiento se ha fundamentado en investigar los diferentes parámetros y usos que permite cada API. Tras conocer estas funcionalidades en cada una de las circunstancias, se ha realizado una serie de pruebas con la aplicación de Postman. Adicionalmente, se ha añadido el Bearer Token para las APIs que lo requerían, que concretamente han sido aquellas facilitadas por Euskalmet. Una vez asegurado el correcto funcionamiento de las APIs, se han incluido dichas consultas en el código Python con el fin de recoger los datos y continuar con el ciclo de vida del conjunto de datos.

Es destacable el hecho de que, durante el proceso de recogida de los datos desde las APIs, se han incluido mecanismos de tolerancia a fallos, permitiendo reintentar la solicitud en varias ocasiones más como medida de prevención a los posibles inconvenientes que pudiesen llegar a ocurrir en el trascurso del proceso.

Para tener una visión más detallada, a continuación, se presentan las APIs que han recogido la totalidad de los datos utilizados en este proyecto.

* Número de páginas en los que se agrupan los puntos de medición:

<https://api.euskadi.eus/traffic/v1.0/meters>

Se utiliza esta API de Open Data Euskadi para conocer el número de puntos de medición existentes en todo el País Vasco, y consecuentemente en cuántas páginas se agrupan. El valor obtenido es utilizado en posteriores APIs.

* Listado de los puntos de medición:

[https://api.euskadi.eus/traffic/v1.0/meters?\_page={config.contador\_pags}](https://api.euskadi.eus/traffic/v1.0/meters?_page=%7bconfig.contador_pags%7d)

Se utiliza esta API de Open Data Euskadi para recoger un listado con los identificadores únicos de todos los puntos de medición existentes en el País Vasco. Estos valores retroalimentan otras APIs que aportan los valores que realmente son utilizados. La variable “*config.contador\_pags*” recorre todas las páginas en las que existen datos para recoger sus respectivos identificadores únicos.

* Número de páginas en los que se agrupan los datos del tráfico por cada punto de medición:

[https://api.euskadi.eus/traffic/v1.0/flows/byYear/{year\_month[0]}/byMeter/{meter}](https://api.euskadi.eus/traffic/v1.0/flows/byYear/%7byear_month%5b0%5d%7d/byMeter/%7bmeter%7d)

Se utiliza esta API de Open Data Euskadi para conocer el número de páginas que agrupan los datos de un punto de medición concreto, referente a los flujos de tráfico. Se obtiene el recuento de los datos recogidos en un año. El valor obtenido es utilizado en la API posterior. Mediante la primera posición del parámetro *“year\_month”* se devuelve el año que interesa conocer y mediante *“meter”*, el punto de medición. Este último itera a través del array que recoge la totalidad de estos.

* Valores de los flujos de circulación:

[https://api.euskadi.eus/traffic/v1.0/flows/byYear/{year\_month[0]}/byMeter/{meter}1?\_page={config.contador\_pags}](https://api.euskadi.eus/traffic/v1.0/flows/byYear/%7byear_month%5b0%5d%7d/byMeter/%7bmeter%7d1?_page=%7bconfig.contador_pags%7d)

Esta API de Open Data Euskadi obtiene la totalidad de los datos de cada punto de medición en un año concreto. En la variable “*year\_month*” se almacenan parejas de valores que hacen referencia a la fecha, siendo el primer valor el año y el segundo el mes. Únicamente se utiliza el primero de estos valores.

Para el otro parámetro, “*meter*”, se utilizan todos los valores recogidos en la API que previamente ha obtenido el listado de los puntos de medición, aplicando cada uno de forma iterativa. De esta forma, se recorren todas las fechas establecidas entre la fecha inicial y fecha final para cada punto de medición. Y de igual forma, este proceso se lleva a cabo para todos los identificadores únicos de los puntos de medición, lo que conforma un amplio catálogo de datos sobre los que trabajar. Hay que tener en cuenta que también se itera el parámetro “*config.contador\_pags*”.

* Valores de la calidad del aire:

[https://api.euskadi.eus/air-quality/measurements/daily/counties/{prov}/from/{inicio}/to/{fin}](https://api.euskadi.eus/air-quality/measurements/daily/counties/%7bprov%7d/from/%7binicio%7d/to/%7bfin%7d)

Mediante esta API de Open Data Euskadi se obtienen los valores referentes a la calidad del aire en diferentes estaciones meteorológicas ubicadas por el País Vasco. Esta API la componen 3 diferentes parámetros. El primero de ellos hace referencia a la provincia en la que se quieren recoger datos. Como tal, cada una de las 3 provincias que componen el País Vasco tiene asociado un código numérico que referencia a dicha zona geográfica y es utilizado en esta ocasión. El segundo y tercer parámetro limitan el uso de la API temporalmente, ya que se define la fecha de inicio y fin de la recogida de dichos datos.

* Listado de las estaciones de Euskalmet:

<https://api.euskadi.eus/euskalmet/stations>

Se utiliza esta API de Euskalmet para recoger un listado con datos de las estaciones existentes en el País Vasco. Los principales elementos que se recogen son tanto los identificadores únicos de las estaciones, así como la fecha de su snapshot, lo que puede asemejarse a una copia de seguridad.

* Datos de las incidencias en las carreteras del País Vasco:

[https://api.euskadi.eus/traffic//v1.0/incidences/byDate/{year}/{month}/{day}](https://api.euskadi.eus/traffic//v1.0/incidences/byDate/%7byear%7d/%7bmonth%7d/%7bday%7d)

Se utiliza esta API de Open Data Euskadi para recoger los datos relacionados con las incidencias en las carreteras del País Vasco. Entre estos registros, se puede encontrar la carretera en la que ha ocurrido el incidente, el tipo de incidente y el intervalo de su duración, a través de la fecha de inicio y de fin. Los parámetros que se utilizan únicamente hacen referencia a la fecha concreta en la que interesa conocer las incidencias ocurridas.

* Valores de las estaciones de Euskalmet:

[https://api.euskadi.eus/euskalmet/stations/{est}/{snap}](https://api.euskadi.eus/euskalmet/stations/%7best%7d/%7bsnap%7d)

Mediante esta API de Euskalmet se obtienen los datos de las estaciones en la fecha de una de sus copias de seguridad realizadas, permitiendo conocer los valores que recogen los diferentes sensores que componen la estación.

El procedimiento que se sigue consiste en alimentar la propia API desde un array que se compone a su vez por varios objetos. Estos objetos más concretamente son parejas de datos, donde se almacena el identificador único de la estación como primer dato y la fecha del snapshot más reciente de esa estación en segunda posición. Estos valores se facilitan a la API mediante los parámetros *“est”* y *“snap”* respectivamente.

* Valores de los sensores ubicados en las estaciones de Euskalmet:

[https://api.euskadi.eus/euskalmet/readings/summarized/byDay/forStation/{estacion}/{sensor}/measures/{tipo\_medida}/{medida}/at/{year}/{month}/{day}](https://api.euskadi.eus/euskalmet/readings/summarized/byDay/forStation/%7bestacion%7d/%7bsensor%7d/measures/%7btipo_medida%7d/%7bmedida%7d/at/%7byear%7d/%7bmonth%7d/%7bday%7d)

Mediante esta última API se obtienen los datos que facilitan los sensores ubicados en las diferentes estaciones de Euskalmet. De todas estas mediciones, se han definido como relevantes algunas métricas en concreto que son las recogidas causa de la iteración de los identificadores únicos de las estaciones, que son parametrizados en la variable *“estación”.* Además, se facilita el sensor del cual recoger los datos mediante la variable *“sensor”.* Sobre la medición que se busca obtener, se conoce tanto el tipo como la medición exacta, gracias a los parámetros *“tipo\_medida”* y *“medida”.* Por último, hacer mención de que cada consulta se realiza en una fecha específica, la cual se conoce a través de las variables *“year”*, *“month”* y *“day”.*

## Limpieza y procesamiento de datos

Una vez recogida la totalidad de los datos desde las diferentes APIs, se les ha aplicado la limpieza y el procesamiento correspondiente, mediante el lenguaje de programación de Python. Para comprender este proceso con una mayor claridad, esta sección se divide en base a los diferentes datos recogidos, explicando los procesos que involucren a cada uno de ellos respectivamente.

Sin embargo, existe un factor altamente relevante que explicar antes de comenzar a describir los procesos a los que se han expuesto los datos. Este punto hace referencia a la definición de las fechas que conforman el periodo en el que se recogen datos, estableciendo una fecha inicial y una fecha final. Este periodo tiene varias utilidades dentro de la aplicación, entre las que destacan la alimentación de diferentes APIs y el procesado de los datos previamente obtenidos.

Dada las limitaciones tanto capacitivas como temporales, el conjunto de datos que se utiliza en “Biker” es del primer semestre de 2024, recogiendo desde el mes de enero hasta el mes de junio, ambos incluidos. Esta decisión viene tomada en primera instancia por el límite de memoria existente en los dispositivos hardware utilizados, lo que al ser contrastado con el número tan elevado de carreteras existentes y su recogida de al menos un registro diario, hace inviable el aumento temporal.

Si a esto, se le suma la inmensa cantidad de indicadores meteorológicos recogidos, se obtiene un conjunto considerablemente amplio, y por consiguiente, pesado. Esto hace inviable la recogida de datos durante un periodo de tiempo mayor.

Ligado a este punto, aparece la limitación temporal, ya que, con el largo tiempo de ejecución destinado a la recogida y tratamiento de datos, no es posible realizar una recogida a mayor escala. No obstante, como línea futura puede valorarse el aumento de los datos históricos, permitiendo así a posteriores técnicas de IA un aprendizaje basado en ciclos temporales como las estaciones. Para ello, se precisarían mayores recursos de computación.

Tras este pequeño inciso, se procede a presentar la limpieza y procesamiento de los diferentes subconjuntos.

### Datos ligados al tráfico

Este subapartado recoge todos los datos que tienen relación directa con el tráfico. Principalmente, el conjunto de datos se puede dividir en 2 agrupaciones como son los registros de las incidencias y de los flujos de las carreteras.

El primero de estos subgrupos recoge ciertas propiedades de los datos devueltos por la API consultada, realizando un filtrado que distingue entre los elementos interesantes o de utilidad y los que no aportan valor al proyecto.

A continuación, se pueden conocer las propiedades de los datos seleccionadas:

* *“\_id”*: identificador numérico único de la incidencia registrada.
* *“sourceId”*: identificador numérico único del origen desde el que se ha detectado la incidencia.
* *“incidenceType”*: tipo de la incidencia detectada.
* *“province”*: provincia en la que se ha detectado la incidencia.
* *“cause”*: causa de la incidencia detectada.
* *“citytown”*: ciudad en la que se ha detectado la incidencia.
* *“startDate”*: fecha de comienzo de la incidencia.
* *“road”*: carretera que ha sufrido la incidencia.

**Figura 9.** *Registros de la colección “incidencias” en MongoDB.*

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Fuente: Elaboración propia.

Seguidamente, se formatean los datos seleccionados y se conforman diccionarios de datos, los cuales siguen una estructura definida. De esta forma se facilita el análisis posterior, almacenando estas estructuras en un array.

En lo que al segundo grupo se refiere, existe un tratamiento de estos mucho más exhaustivo. La principal diferencia viene dada en la existencia de un mayor volumen de datos en comparación con los registros de las incidencias, conformando un proceso más laborioso.

El primer paso consiste en obtener el dato del número de páginas en las que se recogen los diferentes identificadores únicos de los puntos de medición. Este dato se obtiene mediante una API y su única utilidad es alimentar otra API posterior, para conocer la totalidad de estos identificadores únicos.

Seguidamente, se realizan solicitudes a la misma API de forma iterativa, compartiendo como parámetro el número de la página a la que se accedía, incrementado este valor hasta alcanzar el número de páginas máximo en el que se recogen datos. De esta forma se consiguen almacenar todos los identificadores únicos de los puntos de medición.

Con los identificadores previamente recogidos, es posible realizar las solicitudes pertinentes a la última API que se involucra en este conjunto de datos. De manera adicional, las parejas de datos de año y mes confeccionadas previamente también son requeridas para realizar solicitudes a esta API. A continuación, se presentan los datos devueltos por la API tras ser formateados:

* *“\_id”*: identificador único que recoge el identificador único del punto de medición, la fecha de la medición y la hora de esta.
* *“meterId”*: identificador numérico único del punto de medición.
* *“source”*: identificador numérico único del origen desde el que se ha realizado la medición.
* *“fecha”:* fecha de la medición.
* *“año”*: primero de los valores de la variable “*year\_month*”, que almacena el año y es utilizada en la API.
* *“mes”*: el segundo valor de la pareja de datos de la variable “*year\_month*”, que almacena el mes.
* *“vel\_media”*: velocidad media de los vehículos que han recorrido la carretera en esa medición. En caso de no tener dato, se asigna un 0 automáticamente.
* *“vehiculos”*: número de vehículos que recorren la carretera durante la medición.

**Figura 10.** *Registros de la colección “flows” en MongoDB.*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia.

Tras obtener todos los datos de los diversos flujos de tráfico, se detecta que existen varios registros en un mismo día y que se sitúan en una misma carretera. Para simplificar el análisis a realizar posteriormente, se decide agrupar todas las casuísticas que cumplan las condiciones previamente mencionadas. Para ello, se realiza el sumatorio de todos los vehículos recogidos durante cada día en esa vía, así como el cálculo de la media ponderada de la velocidad media alcanzada por los dichos vehículos.

De esta forma se obtienen unos datos más organizados, obteniendo un único diccionario por carretera y día. Estos diccionarios se almacenan en un array.

### Datos ligados a la meteorología

Por otra parte, esta sección involucra todos aquellos elementos del ámbito meteorológico, que completan el conjunto de datos utilizados en el proyecto “Biker”. En rasgos generales, los datos recogidos en este grupo se dividen en registros sobre la calidad del aire, las estaciones meteorológicas y datos recogidos de los diferentes sensores que componen las estaciones.

Respecto a los datos que describen la calidad del aire, son los que menor procesamiento y limpieza necesitan de todo el subconjunto definido. Esto ocurre porque únicamente se accede a una API para la obtención de los mismos, sin requerir ningún otro dato o parámetro adicional.

Sin embargo, es necesario realizar una solicitud a la API por cada una de las provincias sobre las que se quieren obtener datos, repitiendo el proceso en 3 ocasiones. Seguidamente, se muestran los datos de interés recogidos mediante la API:

* *“\_id”*: identificador numérico único de la medición de la calidad del aire en una estación.
* *“fecha”*: fecha de la medición de la calidad del aire.
* *“id\_estacion”:* identificador numérico único de la estación de la que se obtiene la medición de la calidad del aire.
* *“nombre”*: nombre de la estación donde se obtiene la medición de la calidad del aire.
* *“provincia”*: identificador numérico que hace referencia a la provincia. Se actualiza según el bucle que se esté recorriendo, haciendo referencia a la provincia pertinente en cada caso.
* *“measure[‘name’]”*: nombre de la métrica que se recoge de la API en la medición.
* *“unidad\_medicion”*: unidad en la que se mide la métrica recogida.

**Figura 11.** *Registros de la colección “calidad\_aire” en MongoDB.*

*Interfaz de usuario gráfica, Texto

Descripción generada automáticamente*

Fuente: Elaboración propia.

Todos los elementos recogidos, independiente de la provincia de origen, se almacenan en un mismo array. Esto permite agrupar todos los valores organizados en diccionarios estructurados, facilitando el análisis que se realizará en las siguientes fases.

En el subconjunto de datos que se engloban las estaciones meteorológicas, se requiere de un tratamiento ligeramente más exhaustivo. En gran medida, esto ocurre por la gestión de los snapshots.

Este proceso comienza accediendo a la API que devuelve los identificadores únicos de las estaciones y sus snapshots. Cada pareja de estación-snapshot se almacena en un array, que a su vez, se almacena en un segundo array, donde se almacenarán la totalidad de los datos. Como cada estación tiene varios snapshots, existen parejas en los que la estación coincide, diferenciándose únicamente en el snapshot. Estas casuísticas no aportan información, por lo que se realiza una comprobación posterior con la finalidad de almacenar únicamente la pareja de datos que recoja el snapshot más reciente de cada estación.

Para esta comprobación, se conforma un nuevo array con los valores de las estaciones y su último snapshot. El procedimiento a seguir consiste en recorrer el array conformado por todos los datos y verificar la estación de cada pareja. Si se detecta que la estación ha sido previamente revisada, se contrastan los snapshots, almacenando únicamente la pareja que tenga una fecha más reciente. Si por el contrario dicho identificador de la estación no ha sido detectado hasta el momento, es añadido directamente al array.

Con el array ya conformado, se procede a iterar cada uno de esos valores en la última API de las estaciones meteorológicas. Se realiza una solicitud a esta API para cada pareja de datos que consecuentemente recoge una estación diferente. Mediante ella, se obtiene un listado de los nombres de los sensores que se hallan en la propia estación meteorológica. Este listado es añadido al diccionario que se devuelve por cada petición a la API, siendo posteriormente almacenado. Para conocer en mayor detalle la estructura del diccionario tipo, se presentan los parámetros a continuación:

* *“\_id”*: identificador único de la estación, compuesto por valores alfanuméricos.
* *“tipo\_estacion”*: tipo de la estación meteorológica.
* *“nombre”:* nombre de la estación en español, que generalmente hace referencia a la localización de esta.
* *“municipio”*: nombre del municipio en la que se sitúa la estación, en español.
* *“provincia”*: nombre de la provincia en la que se sitúa la estación, en español.
* *“snapshot”*: identificador numérico del snapshot, que recoge la fecha del mismo.
* *“sensores”*: listado anidado de los sensores que conforman la estación.

**Figura 12.** *Registros de la colección “estaciones” en MongoDB.*

*Texto

Descripción generada automáticamente*

Fuente: Elaboración propia.

Al igual que con las casuísticas anteriores, estos diccionarios son almacenados en un array que se utilizará en posteriores etapas del ciclo de vida del conjunto de datos.

Tratando ya el conjunto de datos más complejo de esta sección, que engloba todos esos elementos recogidos desde los múltiples sensores que contienen las estaciones meteorológicas, se procede a explicar su limpieza y tratamiento.

El procedimiento que se sigue consiste en iterar cada diccionario existente en el array que recoge los datos sobre las estaciones. De cada diccionario, se obtiene el listado de los sensores que componen cada estación, y consecuentemente, de cada sensor se obtienen a su vez todas las mediciones posibles, dentro de las métricas meteorológicas que sean interesantes de analizar. A continuación, se definen dichas métricas:

* Humedad: humedad relativa del aire. La unidad de medición es el porcentaje de agua existente en el aire.
* Temperatura: temperatura del aire. La unidad de medición es el grado Celsius.
* Precipitaciones: cuantía de las precipitaciones. La unidad de medición es el milímetro por metro cuadrado.
* Velocidad media del viento: velocidad promedio del viento. La unidad de medición es metro por segundo.
* Dirección del viento: dirección que toma el viento. La unidad de medición es los grados de ángulo.

Para ello, se realiza una solicitud a la API por cada sensor de cada estación, adjuntando a su vez el tipo de la métrica a obtener, así como la métrica en cuestión. Además, se añade la fecha como parámetro, con un desglose de día, mes y año.

Mediante la respuesta de la API, se recogen los valores considerados como más relevantes de cara al análisis posterior, conformando un diccionario como en los casos anteriores. Posteriormente, se muestran los datos recogidos en lo que sería un diccionario tipo:

* *“\_id”*: identificador único que recoge el año, mes, día junto al nombre de la medida, nombre de la estación y nombre del sensor. Todos estos valores son concatenados en un único nombre, permitiendo diferenciarse con cualquier otro registro.
* *“año”*: año en la que se realizó la medición.
* *“mes”:* mes en la que se realizó la medición.
* *“dia”:* día en la que se realizó la medición.
* *“estacion”:* estación que realizó la medición.
* *“sensor”:* sensor de la estación mencionada que realizó la medición.
* *“tipo\_medida”:* tipo de la métrica que se mide.
* *“medida”:* métrica concreta que se mide.
* *“max”:* de todos los valores obtenidos durante la medición, el valor más elevado.
* *“max\_acumulado”*: el valor máximo acumulado se obtiene en determinadas métricas como el valor máximo registrado durante la medición. En las mediciones que no se recoge esta métrica, se devuelve el valor max.
* *“min”*: de todos los valores obtenidos durante la medición, el valor ínfimo.
* *“media”*: de todos los valores obtenidos durante la medición, el valor promedio.
* *“total”*: de todos los valores obtenidos durante la medición, el valor total.

**Figura 13.** *Registros de la colección “meteo” en MongoDB.*

*Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente*

Fuente: Elaboración propia.

## Almacenamiento de datos

Dando por finalizada la fase de limpieza y procesamiento, se obtiene el conjunto de datos que finalmente será utilizado en el proyecto. Seguidamente, en la fase de almacenamiento, estos elementos son almacenados en una base de datos, la cual es seleccionada dependiendo la necesidad de cada proyecto. En la situación de “Biker”, MongoDB fue la opción más acorde para las necesidades del proyecto, permitiendo tener una mayor flexibilidad en las estructuras de los documentos de datos. Además, la experiencia previa con esta herramienta fue influyente cuanto menos en la decisión de utilizar esta opción.

En lo que a la distribución del almacenamiento de los datos se refiere, se conforman varias colecciones dentro de la base de datos creada bajo el nombre de *“TFM”*.

La primera de ellas se conoce bajo el nombre de *“calidad\_aire”* y se recogen todos los registros de la calidad del aire en las diferentes provincias del País Vasco. La segunda colección tiene el nombre de *“estaciones”*, y se almacenan los datos que tienen relación con las estaciones meteorológicas. La tercera colección se conoce como *“incidencias”*, siendo el almacén de los datos que reflejan las incidencias ocurridas en las carreteras. La cuarta colección toma el nombre de *“flows”* y se recogen todos los valores de los flujos de tráfico. Y por último, existe la colección *“meteo”*, donde se recogen los datos de los sensores de las estaciones meteorológicas. Las métricas almacenadas son las siguientes: humedad, temperatura, precipitaciones, velocidad media y dirección del viento.

**Figura 14.** *Colecciones de la base de datos “TFM” en MongoDB.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Fuente: Elaboración propia.

Además, se ha de mencionar que para cada una de las colecciones se debe definir la inserción de los datos mediante la librería *“pymongo”* de Python, con el fin de establecer una conexión entre la base de datos y el código ejecutado.

## Análisis de datos

Superando el ecuador del ciclo de vida del conjunto de datos, comienza la etapa de análisis. Esta etapa se centra en la explotación de los datos que recogen la calidad del aire, así como todos aquellos elementos que tratan sobre las diferentes métricas de las estaciones meteorológicas y flujos de tráfico. En todas estas ocasiones, y en base a los históricos almacenados, se predicen valores futuros de todos estos datos gracias a diferentes técnicas de IA como la regresión lineal, Random Forest o redes neuronales. A continuación, se define el proceso de análisis realizado de forma completa.

### Análisis exploratorio de datos (EDA)

Inicialmente se realiza un EDA que aplica a todos los conjuntos de datos donde se pueden conocer diferentes características de cada uno.

En un primer instante, se muestran los 5 primeros registros de cada conjunto de datos, permitiendo tener una primera visión general de los datos. Seguidamente se conoce su estructura en mayor detalle, mostrando tanto el número de columnas como de filas que definen cada conjunto de datos. Se indaga en mayor profundidad en las columnas, mostrando el nombre y el tipo de dato que las componen, además de mostrar el número de registros no nulos existentes.

Se visualizan varios estadísticos, como el número total de valores, cuantos de ellos son únicos, el que más veces se ha repetido y el número de repeticiones. Se realiza un desglose por cada una de las columnas que componen el conjunto de datos mostrando las veces que dicho valor ha aparecido. Al final de cada columna, aparece el nombre de la misma, el tipo de dato que se almacena y el número de registros que la compone.

Por último, se genera una matriz de correlación donde se pueden apreciar las relaciones existentes entre las columnas numéricas en cada conjunto de datos.

**Figura 15.** *Ejemplo de una matriz de correlación.*

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Fuente: Elaboración propia.

### Análisis de los subconjuntos de datos

Tras un EDA genérico aplicado a todos los conjuntos de datos, se realiza un análisis más detallado por cada uno de los conjuntos existentes. En ellos se puede encontrar desde técnicas de aprendizaje supervisado como la regresión hasta técnicas de aprendizaje no supervisado como el clustering.

#### Flujos de tráfico

En el conjunto de datos que se tratan los flujos de tráfico, únicamente se utiliza una técnica de aprendizaje no supervisado como es el clustering. El clustering tiene la finalidad de agrupar los datos existentes en un número determinado de grupos, sin la existencia de ninguna etiqueta por cada uno de los grupos conformados.

Por ello, las agrupaciones se realizan en base a la similitud de los elementos. En este caso concreto, se realizan 2 agrupaciones, en base a la afluencia de vehículos y a la velocidad de circulación. De esta forma, se puede clasificar la saturación de vehículos existente en la carretera, así como la velocidad media de circulación en las mismas, permitiendo al usuario final conocer estos datos.

Ambas agrupaciones se han dividido en 5 subclases que varían en el caso de la afluencia de vehículos desde una saturación mínima de las carreteras hasta una afluencia más densa del tráfico. En el caso de la velocidad media ocurre algo similar, existiendo un abanico que abarca desde una velocidad de circulación considerablemente reducida hasta una velocidad elevada. A continuación, se puede apreciar la distribución de ambos clústeres.

Gráfico

Descripción generada automáticamenteGráfico

Descripción generada automáticamente**Figura 16.** *Clustering de los flujos de tráfico.*

Fuente: Elaboración propia.

#### Calidad del aire

En este conjunto de datos se opta por el uso de algoritmos de aprendizaje supervisado para obtener la predicción de algunos indicadores. Para ello, se definen 3 algoritmos, regresión lineal, Random Forest y una red neuronal, con el fin de realizar una comparativa entre sus resultados y concluir que solución obtiene mejores resultados.

El algoritmo de regresión lineal se inicializa a través de la función *“LinearRegression()”* de la librería *“sklearn”*, al igual que el algoritmo Random Forest con la función *“RandomForestRegressor()”*.

En el caso de la red neuronal es necesario un proceso ligeramente más complejo. En primera instancia es necesario definir el tipo de algoritmo, que en este caso es *“Sequential()”*. Seguidamente se añaden tantas capas de neuronas como se quiera. En este caso la red neuronal consta de 3 capas, 2 de ellas son conformadas por 512 neuronas con activación *“relu”* además de la capa de salida que tiene activación *“linear”*, dado que está construida para predecir valores numéricos. Entre las capas se define un *“Dropout”* lo que ocasiona un descarte controlado de resultados para evitar el sobreajuste. Por último, se compila la red neuronal con el optimizador *“adam”* y se definen tanto la perdida como las métricas que evalúan la red neuronal. Se obtiene un resumen final mediante el atributo *“summary”.*

Con la red neuronal definida, se ha de aplicar el algoritmo de *“KerasRegressor”*, definiendo a su vez los ciclos (epochs) a ejecutar, así como la cantidad de datos que se utilizarán en cada ciclo (batch\_size).

Una vez configurados los 3 modelos, se define la validación cruzada a través de KFold. KFold es una técnica utilizada para evaluar el rendimiento de un modelo con la validación cruzada, importada también a través de la librería *“sklearn”.* En este caso se ha definido una validación cruzada en 5 subgrupos, para todos los modelos.

Con los modelos definidos y el KFold listo para aplicarse, únicamente son necesarias unas mínimas transformaciones en el conjunto de datos para poder realizar las predicciones. La primera adaptación consiste en eliminar una serie de columnas que imposibilita la predicción. De igual forma se deben eliminar las instancias de datos inconcluyentes, detectados por el EDA previamente realizado.

Seguidamente se agrupa el conjunto de datos por fecha y provincia, y se rellenan los valores nulos con ceros, dando a entender la ausencia de datos en esas casuísticas.

Con el conjunto ya preparado, se empiezan a realizar las predicciones para cada uno de los indicadores siguientes: Tolueno, S02, PM10, PM2,5, 03 8h y NOX.

El primer paso para cada predicción consiste en dividir la variable a predecir del resto, agrupándolas en las variables X (variables independientes) e Y (variable objetivo). Tras la división, se comienza con la predicción gracias a la función *“cross\_val\_predict”.*  Se obtienen los resultados de la predicción y se calculan las métricas utilizadas para su evaluación, en este caso el Mean Squared Error (MSE), Mean Absolute Error (MAE) y R2 Score.

Tabla 3. *Tabla de resultados* *con las predicciones de la calidad del aire.*

| Medición | | Regresión lineal | Random Forest | Red neuronal |
| --- | --- | --- | --- | --- |
| Tolueno | MSE | 0,86 | 0,87 | 0,68 |
| MAE | 0,6 | 0,53 | 0,45 |
| R2- Score | 0,54 | 0,53 | 0,64 |
| Predicción | 1,12 | 1,14 | 1,1 |
| SO2 | MSE | 0,77 | 0,61 | 0,55 |
| MAE | 0,66 | 0,57 | 0,56 |
| R2- Score | 0,39 | 0,52 | 0,56 |
| Predicción | 4,02 | 4,04 | 3,96 |
| PM2,5 | MSE | 5,72 | 4,68 | 3,41 |
| MAE | 1,76 | 1,49 | 1,29 |
| R2- Score | 0,57 | 0,64 | 0,74 |
| Predicción | 6,78 | 6,74 | 6,68 |
| PM10 | MSE | 22,75 | 24,45 | 21,08 |
| MAE | 3,26 | 3,08 | 2,8 |
| R2- Score | 0,65 | 0,62 | 0,67 |
| Predicción | 13,45 | 13,45 | 13,07 |
| 03 8h | MSE | 196,77 | 171,54 | 113,21 |
| MAE | 10,47 | 9,63 | 7,95 |
| R2- Score | 0,39 | 0,47 | 0,65 |
| Predicción | 63,99 | 64,28 | 61,84 |
| NOX | MSE | 171,06 | 44,48 | 90,61 |
| MAE | 9,49 | 3,09 | 6,22 |
| R2- Score | 0,44 | 0,85 | 0,7 |
| Predicción | 9,9 | 9,71 | 9,59 |

Fuente: Elaboración propia.

#### Incidencias

El conjunto de datos que engloba las incidencias utiliza una técnica de aprendizaje no supervisado como es el clustering, al igual que se hace con los datos de los flujos de tráfico. No obstante, existen algunas diferencias en ambos procedimientos. La más notoria es que se requiere de una adaptación de los tipos de fecha existentes ya que algunos registros contienen un formato con segundos mientras que otros los omiten.

Para ello se unifica el formato, añadiendo los segundos a aquellos registros faltantes y se convierte dicha columna a tipo fecha. Posteriormente se conforma un nuevo conjunto de datos agrupando por la fecha, provincia y tipo de incidencia.

Con esta agrupación realizada, se obtienen 3 clústeres en base al número de incidentes registrados, pudiendo considerar alta, media o baja la cantidad de incidentes ocurridos.

Se muestra la asignación en la siguiente ilustración.

**Figura 17.** *Clustering de las incidencias.*

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Fuente: Elaboración propia.

Una vez realizada la asignación de a qué clúster pertenece cada registro, se ha de hacer una unión para poder corresponder la información con el conjunto de datos previo a la agrupación. De esta forma se genera una clave común en el conjunto de datos sin agrupar y en el agrupado para obtener la totalidad de la información y poder realizar las visualizaciones posteriores.

#### Datos meteorológicos

Y para el último conjunto de datos, el cual engloba todos los elementos correspondientes a los sensores meteorológicos, se sigue una estructura relativamente parecida a la aplicada en el conjunto de datos de la calidad del aire.

En primera instancia, existe una subdivisión en el conjunto de datos, causa de las diferentes mediciones que le aplica a cada uno de ellos. Por ello, se genera un conjunto de datos individual para cada una de las siguientes mediciones: humedad, temperatura, precipitaciones, velocidad media y dirección del viento.

Con los conjuntos conformados, se definen los mismos 3 algoritmos utilizados en el conjunto de datos de la calidad del aire, la regresión lineal, Random Forest y una red neuronal. Se ha de mencionar la existencia de pequeñas diferencias en su configuración, como la diferente dimensión de entrada en la red neuronal. Adicionalmente, se aplica el KFold, posibilitando la aplicación de la validación cruzada.

Tras definir los modelos y el KFold, da comienzo la iteración de las predicciones de los distintos conjuntos de datos. Primero se diferencian las variables independientes y objetivo, seguidamente se obtiene la predicción de cada conjunto y con cada modelo y por último se obtienen las métricas. Una vez obtenidos todos los resultados, se valora en líneas generales cual es el modelo que mejor rendimiento ha tenido.

Tabla 4. *Tabla de resultados* *con las predicciones de los datos meteorológicos.*

| Medición | | Regresión lineal | Random Forest | Red neuronal |
| --- | --- | --- | --- | --- |
| Humedad | MSE | 15,47 | 10,67 | 24,57 |
| MAE | 0,93 | 0,34 | 2,5 |
| R2- Score | 0,93 | 0,95 | 0,89 |
| Predicción | 79,65 | 79,63 | 77,42 |
| Temperatura | MSE | 0,24 | 12,13 | 3,83 |
| MAE | 0,19 | 0,1 | 1,55 |
| R2- Score | 0,95 | 0,7 | 0,91 |
| Predicción | 11,45 | 11,46 | 12,9 |
| Precipitaciones | MSE | 2,2 | 4,34 | 19,62 |
| MAE | 0,44 | 0,12 | 2,52 |
| R2- Score | 0,99 | 0,99 | 0,98 |
| Predicción | 25,64 | 25,64 | 23,6 |
| Velocidad media del viento | MSE | 2,77 | 4,37 | 15,68 |
| MAE | 0,44 | 0,12 | 2,25 |
| R2- Score | 0,99 | 0,99 | 0,99 |
| Predicción | 25,64 | 25,63 | 23,88 |
| Dirección del viento | MSE | 2,03 | 4,46 | 11,46 |
| MAE | 0,44 | 0,12 | 1,88 |
| R2- Score | 0,99 | 0,99 | 0,99 |
| Predicción | 25,64 | 25,63 | 24,06 |

Fuente: Elaboración propia.

### Exportar resultados del análisis

Con el análisis finalizado, se guardan los resultados en nuevos diccionarios, a través de los cuales son insertados en nuevas colecciones de la base de datos.

**Figura 18.** *Colecciones del análisis en la base de datos “TFM” en MongoDB.*

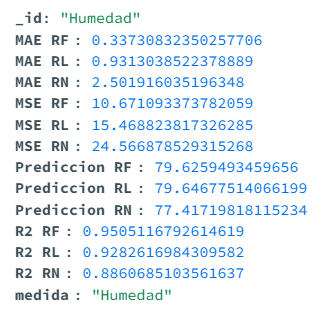


Fuente: Elaboración propia.

A su vez, dichos datos se exportan en formato CSV, con la finalidad posterior de ser importados en Power BI para su visualización. Entre todos los resultados existen 2 variantes, los resultados de las predicciones y los del clustering.

Los resultados obtenidos de las predicciones conforman un diccionario donde se almacenan las propias predicciones y las métricas que referencian. A continuación, se presenta un ejemplo.

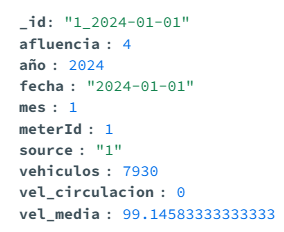
**Figura 19.** *Registro de las predicciones en la base de datos.*



Fuente: Elaboración propia.

Por otro lado, en los casos que se aplica clustering, se recoge el propio conjunto de datos, pero con la información adicional obtenida. Por ello, únicamente se utiliza este conjunto en la visualización, a diferencia de las predicciones, que utilizan tanto el conjunto original como los resultados del mismo.

**Figura 20.** *Registro del clustering en la base de datos.*



Fuente: Elaboración propia.

## Visualización de datos

Dando comienzo a la última fase del proyecto, que no es otra que la visualización de los resultados, se presenta la siguiente solución como interfaz gráfica de dicho proyecto:

<https://app.powerbi.com/groups/me/reports/0081ba80-08e1-4d4c-910c-d4011ce5033c/b66579ccba4ad3256de1?experience=power-bi>.

Dicha solución consta de 4 pestañas diferentes, en las que se utilizan diferentes tipos de visualizaciones para graficar cada uno de los subconjuntos de datos.

### Análisis meteorológico

En la primera pestaña, se muestra un análisis meteorológico. Dicho análisis se vuelve dinámico gracias a los filtros existentes en la esquina superior derecha, pudiendo obtener un conjunto de datos distinto según las circunstancias lo requieran.

En la esquina superior izquierda se haya una tabla en la que observar las distintas predicciones obtenidas por las técnicas de IA aplicadas, resultado un valor por cada una de las medidas meteorológicas existentes. Adicionalmente, se visualiza a través de tarjetas el R2- Score de las predicciones realizadas por las diversas técnicas de IA.

En la parte central de la visualización se distinguen los 3 grafismos principales. El primero de ellos consta de un gráfico de líneas donde se muestra la evolución del promedio de las diferentes mediciones, a lo largo del primer semestre de 2024. El segundo gráfico es un gráfico circular y muestra la distribución de los datos en base al tipo de medida que representan. Y la última representación es un gráfico de barras agrupadas horizontalmente donde se muestra la cantidad de datos que engloba cada medida.

**Figura 21.** *Dashboard del análisis meteorológico.*

Interfaz de usuario gráfica

Descripción generada automáticamente

Fuente: Elaboración propia.

### Análisis de la calidad del aire

En la segunda pestaña, se encuentra la visualización que hace referencia al análisis de la calidad del aire. Esta pestaña contiene una distribución similar a la anterior, con los filtros también ubicados en la esquina superior derecha y bajo los mismos se hayan las tarjetas con el R2-Score de las predicciones. Junto a estas, existe una tabla con las predicciones obtenidas.

En lo que a los gráficos principales se refiere, como en la pestaña anterior, existen 3 visualizaciones: un gráfico de líneas, un gráfico circular y un gráfico de columnas apiladas. El primero, muestra la evolución de las métricas referentes a la calidad del aire. El segundo, muestra la distribución de los datos según el tipo de métrica que es. Y el último, muestra el total de elementos por mes y la subdivisión en base a la métrica que pertenece.

**Figura 22.** *Dashboard del análisis de la calidad del aire.*

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Fuente: Elaboración propia.

### Análisis de los flujos de tráfico

En el tercer dashboard se aprecia el análisis de los flujos de tráfico. Como no podía ser de otra forma, se mantiene la distribución que sitúa a los filtros en la esquina superior derecha, permitiendo distinguir debajo de los mismos unas tarjetas que muestran la velocidad de circulación y la afluencia de vehículos.

Debajo del título de la visualización se muestran 3 medidores, representando la velocidad mínima, media y máxima registrada en los datos sobre los que se realiza la visualización. A medida que dicho conjunto de datos cambie por los filtros aplicados, este valor se actualizará.

Por último, hacer énfasis en los 4 gráficos situados en la zona central de la visualización. El primero de ellos lo conforma un gráfico de líneas, donde se plasma la evolución de la velocidad media de circulación. El segundo representa un grafismo muy similar al primero, representando esta vez el número de vehículos. Y para finalizar, las dos figuras restantes son gráficos de columnas apiladas, donde se muestra la distribución tanto del clúster conformado para clasificar la velocidad media de circulación como del clúster de afluencia de tráfico.

**Figura 23.** *Dashboard del análisis de los flujos de tráfico.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia.

### Análisis de incidencias

Para finalizar con la última pestaña, se presenta el análisis de incidencias. En este caso, se utiliza toda la zona superior para agrupar los diferentes filtros a utilizar, dado la gran cantidad existente.

Adicionalmente, en esta pestaña existen 5 variantes de visualizaciones, desde un gráfico de columnas agrupadas que muestra la distribución del clúster de incidencias hasta unas tarjetas horizontales donde conocer los datos de las propias incidencias. También existe un gráfico de barras agrupadas, donde se da a conocer el top de tipos de defectos que más incidencias engloban.

En la zona inferior del dashboard, se muestra la evolución temporal del número de incidencias, así como un mapa donde se conoce la localización geográfica de la incidencia y el clúster al que pertenece, en base a un color verde (bajo), amarillo (medio) o rojo (alto).

**Figura 24.** *Dashboard del análisis de incidencias.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia.

# Código fuente y datos analizados

## Repositorio

Durante el transcurso del proyecto, se ha ido publicando en el repositorio Github del proyecto las novedades y actualizaciones del proyecto con el fin de tener una copia de seguridad en la nube. De igual forma, si hubiera sido un proyecto en equipo, mediante este repositorio se podría haber compaginado el desarrollo del mismo de forma paralela, gracias a las herramientas que el mismo facilita. Se puede acceder a dicho repositorio mediante el siguiente hipervínculo: <https://github.com/iker-sebastian/TFM>.

## Código fuente

A continuación, se presenta la estructura organizativa que se ha utilizado para la gestión del software, así como los fragmentos más relevantes del mismo, con el fin de facilitar la comprensión de dicho repositorio.

**Figura 25.** *Estructura organizativa del software.*

Texto

Descripción generada automáticamente

Fuente: Elaboración propia.

Como se puede apreciar, existe una amplia cantidad de archivos que conforman el software del proyecto, destacando en su inmensa mayoría los archivos en lenguaje Python. No obstante, también existen otro tipo de archivos, como los archivos de texto o incluso la propia documentación del proyecto en formato *“.docx”*.

En lo que al software se refiere, se puede realizar una distinción en 2 grupos principales. El primero de estos recoge las 3 primeras fases del proyecto: recolección, limpieza y procesamiento y por último, almacenamiento de datos. Por otro lado, el segundo grupo engloba el análisis posterior de los datos.

Este primer grupo, se ejecuta a través del archivo *“main.py”*, donde se invocan de forma paralela los 3 archivos que derivan en la recogida de los datos.

### main.py

#### \_10\_flows.py

En este archivo se organizan las llamadas a los diferentes métodos desarrollados en el archivo de Python *“\_11\_met\_flows.py”*, los cuales recogen datos de las diferentes APIs relacionadas con los diversos flujos del tráfico.

Inicialmente se recoge la fecha de inicio y fin de los datos a recoger. Con el rango temporal establecido, y dado que existe una cantidad de elementos considerable y varias APIs a las que acceder, es necesario realizar varias consultas. De la primera API, se guardan los números de páginas que ocupan todos los identificadores únicos de las mediciones realizadas entre las fechas establecidas, con el fin de alimentar una API posterior.

# Paginas meter\_Id

def API\_meterId\_pags():

# API de las paginas de meterId

url\_meterId\_pags = 'https://api.euskadi.eus/traffic/v1.0/meters'

# Solcitud

response = requests.get(url\_meterId\_pags)

# Respuesta OK

if response.status\_code == 200:

# Formatear respuesta a json

data = response.json()

data\_totalPages = data['totalPages']

return data\_totalPages

Con la siguiente API, se accede a cada una de las páginas recogidas, obteniendo el id de cada medición. Dicho dato se almacena en un array que se utiliza posteriormente.

Con todos los identificadores únicos recogidos, se asegura que no existan duplicados. Posteriormente, por cada uno de los id recogidos, se accede a otra API que recoge las páginas totales en las que se agrupan los datos. Con este nuevo dato, se accede finalmente a los datos de cada medición, donde a su vez, se formatea.

Dada la complejidad y cantidad de los datos recogidos, se opta por unificar los valores en un único registro por día, facilitando la inserción en la base de datos, así como el análisis y visualización posterior.

# Obtener un unico registro por dia

def unificar\_Flows():

# Se crea un nuevo elemento con formato diccionario

diccionarios\_agrupados = defaultdict(list)

# Se recorren los diccionarios almacenados

for dic in config.array\_dic\_flows:

# Se aplica el filtro del meterId y de la fecha

filtro = (dic['meterId'], dic['fecha'])

# El objeti resultante se añade al nuevo diccionario

diccionarios\_agrupados[filtro].append(dic)

# Aplicacion del filtro para los elementos

for filtro, lista\_dic in diccionarios\_agrupados.items():

# Filtro de la agrupacion

meterId, fecha = filtro

# Se suman los vehiculos de un dia y un determinado meterId

suma\_vehiculos = sum(int(dic['vehiculos']) for dic in lista\_dic)

# Se suman la velocidad media en un dia y un determinado meterId

suma\_vel\_media = sum(int(dic['vel\_media']) for dic in lista\_dic)

# Se divide la velocidad total obtenida entre el numero de mediciones

media\_vel\_media = suma\_vel\_media / len(lista\_dic)

# Se obtiene el primer source de todos los elementos

source = lista\_dic[0]['source']

# Se obtiene el primer año de todos los elementos

año = lista\_dic[0]['año']

# Se obtiene el primer mes de todos los elementos

mes = lista\_dic[0]['mes']

# Se conforma el diccionario

dic\_unificado = {

'\_id': str(meterId) + '\_' + fecha,

'meterId': meterId,

'source': source,

'fecha': fecha,

'año': año,

'mes': mes,

'vel\_media': media\_vel\_media,

'vehiculos': suma\_vehiculos

}

# Añadir el diccionario a un array

config.array\_dic\_flows\_unificados.append(dic\_unificado)

Finalmente se insertan en una colección de la base de datos.

# Insercion flows

for doc in config.array\_dic\_flows\_unificados:

# Actualiza el documento si existe '\_id', si no inserta datos

bdd.coleccion\_flows.update\_one({'\_id': doc['\_id']}, {'$set': doc}, upsert=True)

#### \_20\_calidad\_aire.py

Este archivo tiene una función similar a *“\_10\_flows.py”*, obteniendo en primera instancia la fecha del inicio y final del periodo de recogida de datos. Posteriormente, se accede a los métodos definidos en *“\_21\_met\_calidad\_aire.py”* por cada una de las provincias involucradas, obteniendo así los datos de los sensores como muestra el fragmento de código Python siguiente:

# Bucle para recorrer todas las provincias

for provincia in config.array\_provincia:

# Llamada a la API calidad aire

\_21\_met\_calidad\_aire.API\_calidad\_aire(provincia, config.fecha\_inicial, config.fecha\_hoy)

Es de mencionar que existe una notable diferencia respecto al archivo *“\_10\_flows.py”*, ya que en este caso únicamente se requiere de una API, lo que facilita la recogida de los datos considerablemente. De igual forma, por cada uno de los registros, se conforma un diccionario, que se añade a su vez a una lista, la cual se importa posteriormente en una colección diferente de la base de datos.

#### \_30\_est\_meteorologicas.py

Este archivo es el que mayor cantidad de datos involucra, ya que se realizan llamadas a su vez a 3 archivos más: *“\_31\_met\_est\_meteorologicas.py”*, *“\_32\_incidencias.py”* y *“\_33\_meteorologia.py”*. Sin embargo, el primer paso que se toma, como en las dos ocasiones anteriores es el obtener la fecha de inicio y fin de la recogida de datos.

Teniendo las fechas definidas, se accede a la primera API, la cual recoge los datos de las diferentes estaciones que tiene Euskalmet por el territorio vasco. Esta API, entre otros muchos valores, devuelve los snapshots de cada estación. De estos snapshots, se almacena el más reciente, ya que es considerado el de mayor utilidad.

Con un único snapshot por estación, se almacenan parejas de estación y snapshot en una lista, con la finalidad de poder utilizar estas parejas como parámetros de siguientes APIs.

Por ejemplo, se accede a una de estas APIs en el método *“API\_estaciones\_snapshot()”*, que recorre cada pareja almacenada en la lista anterior para recoger registros concretos de una estación y un snapshot. De esta forma, se obtendrán datos para todas las estaciones de Euskalmet existentes. Por consiguiente, esto requiere de una gran capacidad de cómputo, por lo que se invoca de este método de forma paralela mediante *“ThreadPoolExecutor”,* para obtener una mayor eficiencia en este tipo de consultas tan costosas. Estos elementos, son almacenados en diccionarios que a su vez se almacenan en una lista. Esta lista es insertada en la colección *“estaciones*” de la base de datos, siendo un proceso también llevado a cabo de forma paralela. A continuación, se muestra el formato del diccionario tipo y como se añade a la lista previamente mencionada.

# Crear un diccionario con los datos que interesan

doc = {

'\_id': est,

'tipo\_estacion': data['stationType'],

'nombre': data['name']['SPANISH'],

'municipio': data['municipality']['SPANISH'],

'provincia': data['province']['SPANISH'],

'snapshot': snap,

'sensores': list(config.array\_sensores\_por\_estacion)

}

# Añadir el diccionario a un array

config.array\_dic\_estaciones.append(doc)

Además de la información referente a las estaciones, también son de interés los datos de las incidencias de tráfico. Para ello, siguiendo un método de trabajo en paralelo por la gran cantidad de datos, se accede a otra API por cada día, definida en el método *“API\_incidencias()”*. Esta API guarda los datos en formato de diccionario, siendo añadidos posteriormente a una lista. Por lo tanto, se sigue el procedimiento explicado anteriormente, añadiendo esta lista a una colección de la base de datos, como se muestra seguidamente.

Se ha de mencionar, que en este caso se previene de algún posible valor nulo o erróneo y por ello se establecen en ciertos valores de los diccionarios un valor por defecto en caso de la inexistencia de dicho campo.

for documento in data\_incidencias:

cityTown = documento.get('cityTown', 'road')

causa = documento.get('cause', 'Unknown cause')

carretera = documento.get('road', 'Unknown road')

# Crear un diccionario con los datos que interesan

doc = {

'\_id': documento['incidenceId'],

'sourceId': documento['sourceId'],

'incidenceType': documento['incidenceType'],

'province': documento['province'],

'cause': causa,

'cityTown': cityTown,

'startDate': documento['startDate'],

'road': carretera

}

# Añadir el diccionario a un array

config.array\_dic\_incidencias.append(doc)

Finalmente, los últimos datos que se obtienen son los obtenidos por los sensores meteorológicos, que se recogen en la iteración continua de los días entre las fechas establecidas. Esta iteración por cada día recorre todos los diccionarios. Por cada diccionario se recorren todos los sensores almacenados en el mismo y a su vez por cada sensor, se obtienen todas las mediciones de este. Se presenta el formato resultante de los datos recogidos por cada una de estas iteraciones.

# Crear un diccionario con los datos que interesan

doc = {

'\_id': medida + '\_' + str(year) + str(month) + str(day) + '\_' + estacion + '\_' + sensor,

'año': year,

'mes': int(month),

'dia': int(day),

'estacion': estacion,

'sensor': sensor,

'tipo\_medida': tipo\_medida,

'medida': medida,

'max': data['max']['value'],

'max\_acumulado': maximo\_acumulado,

'min': data['min']['value'],

'media': data['mean'],

'total': data['total']

}

# Añadir el diccionario a un array

config.array\_dic\_datos\_meteo.append(doc)

Y a su vez, se muestra el código que invoca tanto la recogida de datos de las incidencias como de los sensores meteorológicos, el cual está ubicado dentro del bucle temporal. Por lo tanto, este código se ejecuta por cada día entre las fechas establecidas.

def procesar\_datos\_dia(fecha):

year, month, day, \_ = config.Fecha\_Setting(fecha)

\_32\_incidencias.API\_incidencias(year, month, day)

with concurrent.futures.ThreadPoolExecutor() as executor:

futures = []

for diccionario in config.array\_dic\_estaciones:

for sensor in diccionario['sensores']:

for medicion in config.array\_mediciones:

futures.append(executor.submit(\_33\_meteorologia.API\_datos, diccionario['\_id'], sensor, medicion[0], medicion[1], year, month, day))

for future in concurrent.futures.as\_completed(futures):

future.result()

### Archivos generales

Además, de los archivos descendientes de *“main.py”* y de los involucrados con el análisis de los datos que se comentarán posteriormente, existen archivos con un carácter más general que se presentan a continuación.

#### bdd.py

En este archivo se almacenan los parámetros que posibilitan la conexión con la base de datos de MongoDB. Además, se define también el nombre de la base de datos a utilizar y sus colecciones.

#### config.py

Se puede considerar como el típico archivo de configuración donde se definen diferentes variables y métodos relevantes. En primera instancia, en este archivo se encuentra la definición de las fechas de inicio y fin de la recogida de datos.

También se aprecian 2 métodos: *“Fecha\_Setting()”* y *“Year\_Month\_Setting()”*. El primer método obtiene la fecha dividida en día, mes y año, además de un campo numérico con la fecha completa. Por otro lado, el segundo método posibilita recorrer cada mes del intervalo definido, utilizado para alguna API.

Finalmente, se muestran distintas variables que recogen listados de diccionarios, parámetros definidos, iteradores y demás objetos que son utilizados durante el propio código.

#### token\_jwt\_euskalmet.py

Archivo que sirve para utilizar la clave pública y privada de la API de Euskalmet. Se complementa de los archivos ubicados en la carpeta API\_Euskalmet.

### analisis.py

Abordando el segundo grupo previamente definido, el cual se centra en su totalidad en el análisis de los datos, se presenta una estructura con ciertas similitudes al primer grupo. En este caso, el archivo desde el cual se ejecutan las llamadas a funciones es *“análisis.py”*. No obstante, además de estas llamadas, en este mismo archivo se realiza un análisis exploratorio de datos. A continuación, se muestra el análisis que se realiza para cada conjunto, siendo iterado en un bucle.

# Visualizar las primeras lineas del df

print(array.head())

print('--------------------------------------------------------------------')

# Estrcutura del conjunto de datos

print(array.shape)

print('---------------------------------------------------------------------')

# Informacion detallada del conjunto de datos

print(array.info())

print('---------------------------------------------------------------------')

# Descripcion del conjunto de datos

print(array.describe())

print('---------------------------------------------------------------------')

# Se obtienen las columnas del conjunto de datos

columnas = list(array.columns)

# Bucle para cada columna

for columna in columnas:

print(array[columna].value\_counts())

Adicionalmente, se grafica una matriz de correlación por cada conjunto de datos, similar a la que se ha podido observar en capítulos anteriores de este mismo documento. Se excluyen dos conjuntos de datos de este grafismo por falta de valores numéricos.

Por último, se realizan las llamadas al resto de documentos donde se realiza el análisis especifico de los datos de dicha categoría.

#### analisis\_10\_flows.py

En este archivo, el primer aspecto que se trata es la conformación de los clústeres, tanto para la afluencia de vehículos, así como para el cálculo de la velocidad media de circulación. Se adjunta una de las dos definiciones de clúster.

# Columna a agrupar por clusteres los vehiculos

X\_flows\_vehiculos = df\_flows[['vehiculos']]

# Definicion del algoritmo

clustering\_flows\_vehiculos = KMeans(n\_clusters=5)

df\_flows['afluencia'] = clustering\_flows\_vehiculos.fit\_predict(X\_flows\_vehiculos)

Con los clústeres definidos, se visualizan los mismos.

plt.scatter(df\_flows['vehiculos'], np.zeros\_like(df\_flows['vehiculos']), c=df\_flows['afluencia'], cmap='viridis')

    plt.xlabel('Afluencia de vehiculos')

    plt.title('Clusters')

    plt.yticks([])

    plt.show()

Y por último, se almacenan en diccionarios y se insertan en la base de datos, con el fin de graficarlos posteriormente en sus respectivos dashboards.

#### analisis\_20\_calidad\_aire.py

En lo que al análisis de la calidad del aire se refiere, en primera instancia se define la red neuronal que se utiliza posteriormente para realizar las predicciones.

# Red neuronal

modelo\_red\_neuronal = Sequential()

modelo\_red\_neuronal.add(Dense(512, activation='relu', input\_dim=5))

modelo\_red\_neuronal.add(Dropout(0.1))

modelo\_red\_neuronal.add(Dense(512, activation='relu'))

modelo\_red\_neuronal.add(Dropout(0.1))

modelo\_red\_neuronal.add(Dense(1, activation='linear'))

# Compilar modelo

modelo\_red\_neuronal.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Resumen del modelo

modelo\_red\_neuronal.summary()

Como se aprecia, existe una red neuronal de 3 capas, con 512 neuronas en las dos primeras y una única neurona en la capa de salida. En las 2 primeras capas se utiliza una activación *“relu”*, que es uno de los parámetros que mejores resultados obtiene en las predicciones, mientras que para la capa de salida se utiliza una activación linear, ya que se quiere predecir un valor numérico.

Con la red neuronal definida, se inicializan los 3 mecanismos a través de los que se obtendrán predicciones como son la regresión lineal, Random Forest y la propia red neuronal. De igual forma se define KFold para realizar la validación cruzada de datos. Seguidamente se realizan unas pequeñas conversiones para que los datos sean útiles en las predicciones a realizar.

Comenzando con las predicciones, el primer paso a realizar es la división entre las variables independientes y las variables objetivo. Por cada una de las medidas a predecir, se obtienen 3 predicciones, una por método, posibilitando a su vez el cálculo de sus respectivas métricas. Se adjunta el código utilizado para la predicción del Tolueno con la red neuronal.

# Cross Validation RN

predicciones\_RN\_Tolueno = cross\_val\_predict(modelo\_RN, X\_Tolueno, Y\_Tolueno, cv=kf)

# Prediccion RL y conversion a float estandar

prediccion\_RN\_Tolueno = np.mean(predicciones\_RN\_Tolueno).item()

# Metricas RN

    mse\_RN\_Tolueno = mean\_squared\_error(Y\_Tolueno, predicciones\_RN\_Tolueno)

    mae\_RN\_Tolueno = mean\_absolute\_error(Y\_Tolueno, predicciones\_RN\_Tolueno)

    r2\_RN\_Tolueno = r2\_score(Y\_Tolueno, predicciones\_RN\_Tolueno)

En la imagen adjunta se presentan las predicciones obtenidas por cada método.

**Figura 26.** *Resultados de las predicciones de la calidad del aire.*

Texto

Descripción generada automáticamente

Fuente: Elaboración propia.

Con estos resultados y los obtenidos con la regresión lineal y Random Forest, se crea un diccionario que se inserta en una lista. Esta lista recoge todas las predicciones de todas las medidas, añadiéndolas a su vez a otra colección de la base de datos para ser utilizado en la fase de visualización.

#### analisis\_32\_incidencias.py

El análisis realizado en el archivo de las incidencias es ligeramente diferente a los presentado hasta el momento, ya que el principal esfuerzo en este caso son las transformaciones requeridas para poder obtener un clustering fiable.

Para ello, se reemplazan varios valores para conseguir una mayor homogeneidad en los elementos, así como se completan algunos valores de fechas, a los que les faltaba la parte horaria. Seguidamente agrupan los datos en base a la fecha, provincia y tipo de incidencia, con el fin de reducir la dimensión del dataset y convertirlo en una unidad más accesible.

Tras dicha conversión, se calcula el clúster, se grafica y se devuelve el clúster asignado a cada instancia del conjunto. Por último, dichos datos se añaden a una nueva colección.

#### analisis\_33\_meteo.py

El análisis meteorológico tiene una estructura muy similar al análisis realizado para la calidad del aire, ya que se ha optado por predecir valores en lugar de aplicar técnicas de aprendizaje no supervisado como en otros conjuntos de datos. No obstante, este análisis da comienzo con la división del conjunto de datos en subconjuntos que engloban las diferentes medidas, las cuales son tomadas de referencia para las distintas predicciones.

A partir de este punto, se sigue el mismo proceso que con el conjunto de la calidad del aire. Se define la red neuronal, teniendo una composición muy similar a la definida previamente para el conjunto de datos de calidad del aire. La diferencia más notoria consiste en las dimensiones de entrada, que en este caso asciende hasta el valor de 7.

Seguidamente se inicializan los 3 métodos de predicción y se calcula la predicción de cada medida. Se presenta un ejemplo con la humedad.

# Division de columnas del DF

X\_humedad = df\_humedad[['año', 'dia', 'max', 'max\_acumulado', 'mes', 'min', 'total']]

Y\_humedad = df\_humedad['media']

# Cross Validation RL

predicciones\_RL\_humedad = cross\_val\_predict(modelo\_RL, X\_humedad, Y\_humedad, cv=kf)

# Prediccion RL

prediccion\_RL\_humedad = predicciones\_RL\_humedad.mean()

# Metricas RL

mse\_RL\_humedad = mean\_squared\_error(Y\_humedad, predicciones\_RL\_humedad)

mae\_RL\_humedad = mean\_absolute\_error(Y\_humedad, predicciones\_RL\_humedad)

r2\_RL\_humedad = r2\_score(Y\_humedad, predicciones\_RL\_humedad)

Con todos los resultados recogidos, se conforman diccionarios y se añaden a una lista, que posteriormente es añadida a una colección de la base de datos.

Al igual que con la calidad del dato, se adjuntan los valores obtenidos de las predicciones, y sus respectivas métricas.

**Figura 27.** *Resultados de las predicciones de las mediciones meteorológicos.*

Texto

Descripción generada automáticamente

Fuente: Elaboración propia.

## Datos Analizados

Tanto los registros recogidos en bruto como los obtenidos del análisis han sido exportados de la base de datos y se ha sido insertado una copia en el repositorio utilizado para este proyecto.

**Figura 28.** *Estructura organizativa de la copia de los datos.*

Texto

Descripción generada automáticamente

Fuente: Elaboración propia.

# Plan de trabajo y gestión del proyecto

En un proyecto es esencial el contenido que se recoge en el mismo, así como su planificación. En esta sección se realiza un enfoque mayormente centrado en la gestión del trabajo, describiendo la estructura y planificación que conforma al proyecto.

## Planificación del proyecto

Este proyecto se ha decidido planificar siguiendo una metodología ágil, en este caso Scrum. Mediante esta metodología, se han ido desarrollando las diferentes fases que componen el ciclo de vida de un conjunto de datos de forma iterativa.

### Cronograma

El cronograma muestra la planificación y la división del proyecto en sprints, a través de los diferentes colores. Se comienza con un sprint enfocado a la investigación, con cierta utilidad para el estado del arte. A partir de este punto, se define un sprint por cada etapa de ciclo de vida de los datos, iniciando con la recogida, seguido de la limpieza y procesamiento, almacenamiento, análisis y visualización. Cada uno de estos sprints, lo componen subfases de investigación, desarrollo y pruebas. Finalmente, se agrega un sprint de revisión para corroborar el correcto funcionamiento y pulir pequeños matices del proyecto.

**Figura 29.** *Cronograma y su división por sprints.*

Gráfico

Descripción generada automáticamente

Fuente: Elaboración propia.

### Análisis DAFO

Para tener una mejor perspectiva del proyecto, se opta en primera instancia por la realización de un análisis DAFO, donde también se pueden encontrar puntos interesantes que tener en cuenta para versiones futuras.

**Figura 30.** *Análisis DAFO.*

Gráfico, Escala de tiempo

Descripción generada automáticamente

Fuente: Elaboración propia.

En lo que a debilidades se refiere, están claramente detectadas y analizadas. El núcleo de este punto se basa en la inexistencia de un análisis de datos en tiempo real y el gran requerimiento de recursos que se necesita tanto para la limpieza y transformación como para el análisis de los datos.

Por el contrario, las fortalezas emergen de la aparición de una solución para un caso de uso específico, donde se solventa una necesidad que en el mercado actual no se había conseguido satisfacer. Asimismo, muchas de las funcionalidades que componen esta aplicación, como la visualización gráfica o la capacidad predictiva, desembocan en beneficios al usuario como un menor tiempo de planificación.

En lo que a amenazas se refiere, se valora la posible obsolescencia de la aplicación, así como la extinción de los orígenes desde los cuales se recogen los datos o el no mantener el software actualizado. De igual forma, puede darse el caso en el que surja alguna aplicación similar con condiciones mejoradas y haga una competencia directa a la aplicación.

Por último, abordando las oportunidades, mencionar el amplio horizonte por explorar que existe, pudiendo transversalizar este modelo de aplicación a infinidad de ámbitos o incluso localizaciones geográficas. De igual forma, el permitir una mayor personalización de la aplicación mediante nuevas funcionalidades puede ser una razón muy probable de incremento en el uso de la misma. La razón principal de esta sentencia se basa en que cada vez más, los usuarios buscan la comodidad por encima de muchas otras características, lo cual con la personalización se consigue en una amplia medida.

### Análisis CAME

Una vez analizado el sistema a implementar con un análisis DAFO, se complementa con un análisis CAME, el cual se podría definir como una extensión del propio análisis DAFO. En este estudio, se busca tomar acciones para corregir las debilidades, afrontar las amenazas, mantener las fortalezas y explotar las oportunidades. Es decir, se pretenden fomentar tanto las fortalezas y oportunidades, así como minimizar las debilidades o amenazas.

Con el fin de corregir las debilidades existentes, se plantea definir procesos automatizados como pueden ser los pipelines, con el fin de intentar solventar el problema de la tan costosa preparación de los datos. Para el alto coste de computación, pudiendo integrarse a su vez con los pipelines, se valora la ejecución del código en proveedores Cloud, únicamente pagando por los recursos utilizados durante su tiempo de ejecución. Por último, y ante la problemática más compleja, el cómo utilizar datos en tiempo real, se requiere de un aumento considerable tanto en la velocidad de acceso a las APIs como en la capacidad de cómputo. Por ello se requeriría mejorar la latencia con la red de origen, así como utilizar una máquina virtual escalable en un proveedor Cloud para satisfacer la demanda de peticiones pertinente. Se podría valorar la actualización diaria de registros, simulando un conjunto de datos en tiempo real.

Para afrontar las amenazas detectadas, la principal medida a tomar es el mantenerse informado de las últimas versiones y actualizaciones del código, con la finalidad de que, si alguna función de las utilizadas se modifica o extingue, poder redefinirla antes de que deje de estar operativa. De cara a erradicar la posible competencia emergente, es importante aportar actualizaciones o funcionalidades periódicamente.

Muy relacionado con este último punto se encuentra la acción de mantener las fortalezas, donde con una continua evolución y mejora en la capacidad predictiva, en la forma de utilizar los diferentes datos o en las visualizaciones, es suficiente para mantener o incluso en ciertos casos potenciar los aspectos destacables de la aplicación.

Para finalizar con el análisis CAME, se presentan las opciones de explotar más aun las fortalezas que puede llegar a tener esta aplicación. Principalmente, se deben centrar esfuerzos en la transversalización de este tipo de aplicación a otros conjuntos de datos, ya que únicamente cambiando el conjunto de datos y realizando pequeñas adaptaciones se obtiene una aplicación con una funcionalidad totalmente diferente y que puede generar un gran interés. Adicionalmente, se debe trabajar en la cada vez más detallada personalización que el usuario debe tener junto a pequeños aditivos que le muestren una continua evolución en la aplicación.

## Problemas encontrados, limitaciones y desvíos

En el transcurso de este desarrollo han surgido varias dificultades o modificaciones en base a la idea inicial. No obstante, se han conseguido solucionar sin impactar la usabilidad final de la aplicación. Para exponer estos puntos, se listan en función de la fase del ciclo de vida en la que hayan surgido.

### Recolección de los datos

El principal problema encontrado en esta fase ha sido la larga ejecución de los procesos de recogida de datos. Dada la gran variedad de APIs utilizadas y la infinidad de consultas realizadas, los procesos resultantes han sido extremadamente duraderos. Este punto derivó en gran medida por las limitaciones en cuanto a las APIs existentes, ya que para tener la información requerida era necesario realizar al menos 3 solicitudes diferentes. A este punto, hay que sumarle el factor de que el hardware utilizado no ha sido un equipo con una gran capacidad de cómputo, por lo que ralentizaba la obtención del resultado final.

A su vez, han ocurrido varias incidencias durante las ejecuciones, causadas en esta fase por el propio proveedor de los datos. Con esta sentencia se hace referencia a que, durante la recogida de los datos, que como bien se ha hecho saber han sido procesos de varios días, han existidos cortes de red por parte del proveedor, imposibilitando la recogida de los datos y generando un error en la ejecución del código. Por ello, como medidas de contención se han instaurado excepciones ante la incomunicación en la red, permitiendo reintentar en varias ocasiones las solicitudes fallidas y posibilitando una alternativa que no generase la detención de la ejecución del código.

### Limpieza y procesamiento de los datos

En la fase de limpieza y procesamiento de los datos, las principales complicaciones encontradas han aparecido en la gestión de los registros que tenían alguna diferencia respecto a la mayoría. Sobre todo, resaltar la complejidad de la estructuración en los diccionarios, pudiendo darse el caso de que en algún registro no exista algún valor o los datos recogidos en las APIs sean distintos.

De ahí que se tomen medidas como establecer un valor por defecto en algún campo para que en caso de que no exista, se utilice dicho valor. También se justifica en parte el uso de MongoDB, permitiendo tener diferencias entre elementos de la misma colección.

Una limitación existente en esta fase es la decisión de realizar un Data Warehouse en lugar de un Data Lake. Esto se debe a que, conociendo la inmensa cantidad de registros existentes, y en vista de que con los recursos existentes sería complicado mover todos los datos, se decidió realizar un Data Warehouse, limpiando y procesando los datos antes de su almacenamiento.

### Almacenamiento de los datos

En lo que al almacenamiento de datos se refiere no han existido incidencias ni desvíos ya que las fases complejas han sido las anteriores. Una vez recogidos los datos y procesados, se han insertado sin mayor problema.

La única posible limitación, aunque tampoco ha afectado al transcurso del proyecto, podría considerarse el necesitar una base de datos no estructurada por posibles diferencias en algunos de los campos existentes de los registros obtenidos.

### Análisis de los datos

En el análisis de datos únicamente ha existido una problemática, a la hora de obtener la predicción con las redes neuronales. Básicamente, el formato en el que se presentaba el resultado obtenido por las redes neuronales no era válido para almacenarlo en la base de datos, causado por el propio modelo previamente conformado. Esta problemática fue compleja de detectar hasta conocer el tipo de variable en el que se englobaba el valor, pudiendo transformarlo posteriormente y continuar con la correcta ejecución del análisis.

### Visualización de los datos

Finalizando con la fase de visualización, no existió ninguna incidencia ni desvío más allá de las limitaciones en las visualizaciones que la herramienta de Power BI facilitaba.

# Resultados

Para valorar los resultados obtenidos con este proyecto, se deben evaluar diferentes ámbitos dentro del mismo. No obstante, gran parte de la valoración debería venir dada en base a una primera experiencia con el uso de la herramienta, pudiendo realmente reconocer donde es útil la aplicación y donde se requiere una mejora. Esta experiencia sería considerablemente mejor si fuera realizada por una pequeña población controlada, en la que se engloben perfiles diferentes y complementarios.

Aun así, se pueden realizar una serie de estimaciones en base a lo que se espera mejorar con la solución presentada:

* Reducción de tiempo en la búsqueda de información: previo a la existencia de esta herramienta, el usuario debía buscar en diferentes ubicaciones la información del tráfico y la información meteorológica. Con esta aplicación, se espera que el tiempo de acceso se reduzca al menos un 50% del tiempo destinado, unificando en una única aplicación toda la información a conocer.
* Mejora en la experiencia del usuario: relacionado con el punto anterior, el usuario únicamente tendrá que acceder a una aplicación para su planificación, siendo notablemente más confortable.
* Claridad en la presentación de los resultados: se muestra una representación visual y gráfica, siendo considerablemente más intuitiva y clarificadora.
* Toma de decisiones informada: al presentarse ambos conjuntos integrados en un mismo entorno, se posibilita una mayor explotación de los datos, que a su vez se ve reflejado en una toma de decisiones en la que se presenta una mayor información al usuario.
* Información predictiva: se presentan predicciones con cierto grado de fiabilidad, que permiten tener una orientación de lo que se puede esperar, complementando el punto anterior.

## Cumplimiento de los objetivos

Tras la finalización del proyecto, se valoran los objetivos que fueron establecidos para el mismo y su nivel de cumplimiento.

Tabla 5. *Tabla de cumplimiento de objetivos.*

| Objetivo | ¿Completado? |
| --- | --- |
| Obtener 2 orígenes de datos. | Sí. |
| Conformar un ETL. | Sí. |
| Almacenar de forma semiestructurada los datos. | Sí. |
| Tener más de 80.000 instancias. | Sí, más de 95.000 instancias. |
| Usar 2 algoritmos supervisados. | Sí. |
| Usar 1 algoritmo no supervisado. | Sí. |
| Aplicar técnicas de validación cruzada. | Sí. |
| Evitar el sobreajuste (overfitting). | Sí. |
| Obtener un R2-Score superior a 0,8 en las predicciones de los índices meteorológicos. | Sí, al menos 1 de los 3 métodos por predicción. |
| Obtener un R2-Score superior a 0,5 en las predicciones de la calidad del aire. | Sí, al menos 1 de los 3 métodos por predicción. |
| Conformar visualizaciones interactivas gracias a una herramienta BI. | Sí. |
| Utilizar 8 tipos de visualizaciones. | Sí. |

Fuente: Elaboración propia.

Como se puede apreciar en la tabla anterior, el grado de cumplimiento de los objetivos establecidos es del 100%. Por ello se puede concluir que dicho proyecto ha sido completado satisfactoriamente en cuanto a objetivos.

# Conclusiones

Tras la finalización del proyecto, se obtiene una herramienta que permite planificar las rutas ciclistas a realizar, posibilitando la toma de decisiones en base a un conocimiento obtenido de una forma rápida, cómoda y sencilla. Esta solución cumple con el objetivo general del proyecto, así como con cada uno de los objetivos secundarios definidos. Consecuentemente, se confirma la relevancia de tener una aplicación que integre tanto el tráfico como el ámbito meteorológico en un único entorno. De esta forma, la planificación de las rutas se simplifica considerablemente, permitiendo una menor pérdida de tiempo a la vez que un análisis más extenso y detallado.

En lo que a la aplicación respecta, se ha conseguido conformar una herramienta que englobe el ciclo completo de los datos. Comenzando desde la recogida, continuando con la limpieza y el procesamiento aplicado y finalmente almacenando más de 95.000 registros, termina el conjunto de etapas donde han sido los datos el elemento a gestionar. Tras la fase de análisis, estos datos se convierten en información, que posteriormente al visualizarla permite al usuario final, encontrar el conocimiento que buscaba.

Con este proyecto he podido desarrollar de inicio a fin una aplicación completa con una notable funcionalidad final, desde un punto de vista teórico y sobre todo práctico. A nivel personal, ha tenido una gran importancia el ser capaz de gestionar un conjunto de datos durante su ciclo de vida completo, desde la recogida hasta la visualización. El aumento exponencial de los datos en los últimos años ha sido un factor esencial en esta inquietud que, junto a la tendencia vislumbrada, promete ser una solución recurrente en el futuro.

Entrando en detalle sobre el desarrollo del proyecto, mencionar la gran relevancia que tiene una buena planificación en este, conformando una base sólida sobre la que construir. Además, facilita la gestión ante posibles desvíos o incidencias que puedan ocurrir. Destacar todos los conceptos técnicos aprendidos a lo largo del desarrollo, implementando diversas tecnologías que colaborando entre sí llegan a conformar potentes soluciones.

En resumen, se valora el resultado del proyecto como satisfactorio por la gran cantidad de ventajas aportadas. De igual forma, mencionar que personalmente el proyecto me ha servido para afianzar conocimientos ya existentes y obtener muchos otros nuevos, que me serán de utilidad en un futuro profesional.

## Lecciones aprendidas

Una vez finalizado el proyecto, se ha tomado nota de ciertos puntos que han tenido un impacto considerable. Seguidamente, se comentarán estos puntos a tener en cuenta para futuros proyectos.

* Gestión del tiempo: uno de los puntos más críticos del proyecto, sobre todo en la fase de recogida de datos. Aunque existía una planificación definida, la enorme cantidad de datos sumado a las incidencias que ocurrieron durante el desarrollo, produjo un leve retraso en el proyecto. Finalmente se pudo solventar correctamente, gracias a los buenos márgenes contemplados en la planificación.
* Planificación detallada: es indispensable una buena planificación para tener al menos unas tareas, plazos o recursos iniciales asignados. De hecho, en este tipo de proyectos es imprescindible el no dar nada por hecho para su correcta ejecución. No obstante, estos plazos iniciales pueden variar por imprevistos o demás circunstancias, pero aun así es recomendable tener una idea principal definida.
* Adaptabilidad: un proyecto de estas características debe de requerir cierta capacidad de adaptación. En este caso, no han ocurrido desviaciones con un gran impacto en el proyecto, pero podría haberse dado el caso en el que se necesitaran de otras alternativas, y debe existir predisposición para llevar a cabo estos cambios.
* Mejora continua: aunque las herramientas utilizadas para la realización de este proyecto se hayan definido, se ha de seguir investigando y conociendo las nuevas tecnologías y metodologías emergentes. Este punto es fundamental para la longevidad de la aplicación, así como para encontrar oportunidades que mejoren la funcionalidad actual.

En conclusión, analizando las lecciones aprendidas mencionadas previamente, se destaca la relevancia de una correcta gestión del tiempo y de una planificación seria, lo que va relacionado directamente con la adaptabilidad que ha de existir durante el proyecto. Con un enfoque a más largo plazo, es esencial el mantenimiento de la aplicación a través de actualizaciones o incluso añadiendo funcionalidades nuevas.

# Trabajo futuro

Dentro del amplio abanico de posibilidades que permite seguir desarrollando este proyecto en el futuro, se destacan las siguientes.

La primera de ellas, y la que quizás mayor impacto pueda llegar a tener es la recolección de datos en tiempo real. Si se consiguen actualizar los registros continuamente, el índice de uso de la herramienta crecerá exponencial y drásticamente, ya que se ajustará mucho mejor a lo que los usuarios requieren, inmediatez (Linares, 2024). Para ello, es esencial la exhaustiva valoración de todos los potenciales impactos que puedan afectar. Algunos ejemplos que valorar podrían ser la cantidad de datos que se recogerá y la capacidad de procesamiento que se requerirá, por ejemplo.

Otro punto a tratar en un futuro puede ser el almacenamiento de los datos. En este proyecto, se ha decantado por la funcionalidad en local de una base de datos, almacenando copias de seguridad en local y en un repositorio. Sin embargo, en una magnitud mayor, se debería valorar un clúster como almacenamiento de datos, incluyendo incluso replicas automáticas para las copias de seguridad.

Como ya se ha mencionado en varias ocasiones, la transversalización de este proceso está directamente ligada con el progreso del mismo. Existen infinidad de formas de aplicarlo, desde la expansión del límite geográfico que actualmente se delimita en el País Vasco hasta la inserción de otros tipos de datos que ayuden en el análisis. En cualquiera de las casuísticas presentadas, es recomendable actualizar varias de las fases del proceso, especialmente el análisis y la visualización de los resultados.

Referencias bibliográficas

*45 personas murieron en las carreteras en Euskadi en 2023, casi la mitad motoristas, viandantes o ciclistas. (2024, enero 4). elDiario.es. https://www.eldiario.es/euskadi/45-personas-murieron-carreteras-euskadi-2023-mitad-motoristas-viandantes-ciclistas\_1\_10813252.html*

*Agencia Estatal de Meteorología. (s. f.). MeteoRuta—Agencia Estatal de Meteorología—AEMET. Gobierno de España. Recuperado 15 de abril de 2024, de https://www.aemet.es/es/eltiempo/prediccion/meteoruta*

*Analytics, B. (2023, septiembre 14). Business intelligence y visualización de datos: Beneficios y herramientas imprescindibles - Bimex Analytics. Business intelligence y visualización de datos: beneficios y herramientas imprescindibles - Bimex Analytics. https://bimexanalytics.com/blog/la-visualizacion-de-datos-es-un-elemento-fundamental-del-business-intelligence-bi-se-trata-de-representar-graficamente-datos-complejos-para-convertirlos-en-informacion-comprensible-y-accesible/, https://bimexanalytics.com/blog/la-visualizacion-de-datos-es-un-elemento-fundamental-del-business-intelligence-bi-se-trata-de-representar-graficamente-datos-complejos-para-convertirlos-en-informacion-comprensible-y-accesible/*

*Apache Cassandra | Apache Cassandra Documentation. (s. f.). [Apache Cassandra]. Recuperado 19 de mayo de 2024, de https://cassandra.apache.org/\_/index.html*

*Base de datos rentable y de alto rendimiento. (s. f.). [Oracle]. Recuperado 19 de mayo de 2024, de https://www.oracle.com/es/database/*

*Bases de datos no relacionales | Bases de datos de gráficos | AWS. (s. f.). Amazon Web Services, Inc. Recuperado 19 de mayo de 2024, de https://aws.amazon.com/es/nosql/*

*BOE.es - DOUE-L-2016-80807 Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)., Reglamento (UE) 2016/679 § 1 (2018). https://www.boe.es/buscar/doc.php?id=DOUE-L-2016-80807*

*Casero, A. (2023, septiembre 14). Control de versiones con Git: Conceptos básicos. Control de versiones con Git: conceptos básicos. https://keepcoding.io/blog/control-de-versiones-con-git/*

*Castro, D. (2022, febrero 21). ¿API? Todo lo que debes saber. Medium. https://davidcasr.medium.com/api-todo-lo-que-debes-saber-d8b011d30aff*

*Cesvimap. (2021, septiembre 4). Influencia de la climatología en los accidentes de tráfico. Revista CESVIMAP. https://www.revistacesvimap.com/influencia-de-la-climatologia-en-los-accidentes-de-trafico/*

*Comobity, la App «segura» para ciclistas. (2015, noviembre 13). 13 de noviembre de 2015. https://revista.dgt.es/es/reportajes/2015/11NOVIEMBRE/1113Comobity-la-App-segura-para-ciclistas.shtml*

*Daniel. (2022, enero 10). Data Warehouse: ¿qué es y cómo utilizarlo? Formación en ciencia de datos | DataScientest.com. https://datascientest.com/es/data-warehouse-que-es-y-como-utilizarlo*

*Datademia. (2021, septiembre 6). ¿Qué es Web Scraping? Datademia. https://datademia.es/blog/que-es-web-scraping*

*Esmerado, J. (2024, enero 20). Las 5 etapas del ciclo de vida de los datos. Medium. https://medium.com/@esmeradovela/las-5-etapas-del-ciclo-de-vida-de-los-datos-f4ddc3c1ed3f*

*Eurosport. (2023, noviembre 24). Muere un ciclista vasco de 19 años al ser atropellado mientras entrenaba—Eurosport. Eurosport Espana. https://www.eurosport.es/ciclismo/muere-un-ciclista-vasco-de-19-anos-horas-despues-de-ser-arrollado-por-un-coche-mientras-entrenaba\_sto9894016/story.shtml*

*Fernández, Y. (2019, octubre 30). Qué es Github y qué es lo que le ofrece a los desarrolladores. Xataka. https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores*

*GanttProject: Características, ventajas y desventajas. (2024, abril 4). Gantt Chart GanttPRO Blog. https://blog.ganttpro.com/es/ganttproject-caracteristicas-ventajas-y-desventajas/*

*Información legal. (2010, marzo 17). [Open Data Euskadi]. https://opendata.euskadi.eus/general/-/informacion-legal-opendata/*

*Información meteorológica tráfico—Incidencias en carretera. (s. f.). www.tutiempo.net. Recuperado 15 de abril de 2024, de https://www.tutiempo.net/trafico.html*

*Lanzan una app para detectar automáticamente los accident... (2022, noviembre 26). Ciclismo a Fondo. https://www.ciclismoafondo.es/noticias/lanzan-app-detectar-automaticamente-accidentes-ciclistas\_268495\_102.html*

*Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, Pub. L. No. Ley Orgánica 3/2018, BOE-A-2018-16673 119788 (2018). https://www.boe.es/eli/es/lo/2018/12/05/3*

*Linares, L. (2024, junio 25). La inmediatez, el principal problema del siglo XXI - Jot Down Cultural Magazine. Jot Down. https://www.jotdown.es/2024/06/la-inmediatez-el-principal-problema-del-siglo-xxi/*

*López, J. F. (2018, febrero 21). Teorema de Bayes—Qué es, fórmula y ejemplos. Economipedia. https://economipedia.com/definiciones/teorema-de-bayes.html*

*Mapa interactivo de la DGT y AEMET, esencial para planificar tu ruta cuando hace mal tiempo. (2023, febrero 24). La Vanguardia. https://www.lavanguardia.com/motor/consejos/20230224/8780046/mapa-carretera-interactivo-aemet-dgt-planificar-mejor-ruta-nieve-lluvia.html*

*Michael T. Gibson, Todd Curtin, & Amit Jhalli. (2020, junio 30). Los accidentes de bicicleta más comunes | Michael T. Gibson P.A., Auto Justice Attorney. Michael T. Gibson P.A. Auto Justice Attorney. https://autojusticeattorney.com/es/common-bicycle-accidents/*

*MongoDB: The Developer Data Platform. (s. f.). MongoDB. Recuperado 19 de mayo de 2024, de https://www.mongodb.com*

*MySQL. (s. f.). [MySQL]. Recuperado 19 de mayo de 2024, de https://www.mysql.com/*

*Na8. (2023, marzo 3). Sets de Entrenamiento, Test y Validación | Aprende Machine Learning. Aprende Machine Learning en Español. https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/*

*Navarro, S. (2022, abril 25). ¿Qué son Data Lakes? | KeepCoding Bootcamps. ¿Qué son Data Lakes? https://keepcoding.io/blog/data-lakes/*

*Neo4j Graph Database & Analytics – The Leader in Graph Databases. (s. f.). [Neo4j]. Graph Database & Analytics. Recuperado 19 de mayo de 2024, de https://neo4j.com/*

*Normativa. (2017, junio 14). [Normativa]. Normativa. https://opendata.euskadi.eus/sobre-open-data/-/normativa-open-data/*

*Open Data Euskadi | datos.gob.es. (s. f.). Recuperado 7 de junio de 2024, de https://datos.gob.es/es/iniciativas/open-data-euskadi*

*Ortega, C. (2019, agosto 21). Métodos de recolección de datos más efectivos. QuestionPro. https://www.questionpro.com/blog/es/metodos-de-recoleccion-de-datos/*

*PostgreSQL Global Development Group. (2024, mayo 19). PostgreSQL. PostgreSQL. https://www.postgresql.org/*

*Power BI: Visualización de datos | Microsoft Power Platform. (s. f.). [Power BI]. Recuperado 20 de mayo de 2024, de https://www.microsoft.com/es-es/power-platform/products/power-bi*

*Press, E. (2023, enero 7). Los accidentes con ciclistas en Euskadi crecen un 20% en los últimos cinco años y los heridos graves y fallecidos un 9%. 7 de enero de 2023. https://www.europapress.es/euskadi/noticia-accidentes-ciclistas-euskadi-crecen-20-ultimos-cinco-anos-heridos-graves-fallecidos-20230107120751.html*

*QlikView – Analítica y cuadros de mando interactivos y eficaces | Qlik. (s. f.). QlikView – Analítica y cuadros de mando interactivos y eficaces | Qlik. Recuperado 20 de mayo de 2024, de https://www.qlik.com/es-es/products/qlikview*

*¿Qué es D3? | D3 por observable. (s. f.). ¿Qué es D3? Recuperado 20 de mayo de 2024, de https://d3js.org/what-is-d3*

*¿Qué es el aprendizaje no supervisado? | IBM. (2023, mayo 4). [IBM]. https://www.ibm.com/es-es/topics/unsupervised-learning*

*¿Qué es el aprendizaje supervisado? | IBM. (2024, mayo 10). [IBM]. https://www.ibm.com/es-es/topics/supervised-learning*

*¿Qué es una base de datos NoSQL? | IBM. (2024, abril 14). [IBM]. https://www.ibm.com/es-es/topics/nosql-databases*

*¿Qué es una base de datos relacional? (s. f.). Oracle. Recuperado 19 de mayo de 2024, de https://www.oracle.com/es/database/what-is-a-relational-database/*

*¿Qué son los almacenes de datos? (s. f.). Google Cloud. Recuperado 19 de mayo de 2024, de https://cloud.google.com/learn/what-is-a-data-warehouse?hl=es*

*¿Qué son los lagos de datos? (s. f.). Google Cloud. Recuperado 19 de mayo de 2024, de https://cloud.google.com/learn/what-is-a-data-lake?hl=es*

*Redis—La plataforma de datos en tiempo real. (s. f.). [Redis]. Redis. Recuperado 19 de mayo de 2024, de https://redis.io/es/*

*SAP BusinessObjects | Plataforma y suite de Business Intelligence (BI). (s. f.). SAP. Recuperado 20 de mayo de 2024, de https://www.sap.com/spain/products/technology-platform/bi-platform.html*

*Software de análisis e inteligencia de negocios | Tableau. (s. f.). [Tableau]. Recuperado 20 de mayo de 2024, de https://www.tableau.com/es-es*

*Soluciones de data lake | IBM. (s. f.). [IBM]. Recuperado 19 de mayo de 2024, de https://www.ibm.com/es-es/data-lake*

*SQL Server 2022 | Microsoft. (s. f.). [Microsoft SQL Server]. Recuperado 19 de mayo de 2024, de https://www.microsoft.com/es-es/sql-server/sql-server-2022*

*Tehreem Naeem. (2020, noviembre 7). ¿Qué es el repositorio de datos? (Definición, tipos y beneficios). Astera. https://www.astera.com/es/type/blog/data-repository/*

*Zapata, J. R. (2024, marzo 14). Visualización de Datos con Python. Jose Ricardo Zapata. https://joserzapata.github.io/courses/python-ciencia-datos/visualizacion/*

Glosario

Por orden alfabético:

* Array: estructura de datos en programación donde se almacenan varios elementos del mismo tipo.
* Batch size: número de ejemplos utilizados para una iteración del entrenamiento de la red neuronal.
* Bearer Token: clave de autentificación para acceder a determinadas APIs.
* Big Data Lifecycle Managment: proceso de gestión del ciclo de vida de los datos.
* Cassandra: base de datos no estructurada de tipo clave-valor.
* Cloud: entrega de una amplia variedad de servicios a través de internet.
* Clustering: técnica de aprendizaje no supervisado donde se agrupan los elementos en base a la similitud de sus características.
* Cross-Validation: técnica utilizada para dividir el conjunto de datos e iterar los mismos entre el entrenamiento y test.
* Dashboard: panel de visualización que contiene gráficos.
* Data Lake: repositorio de grandes volúmenes de datos donde se almacenan previo a su tratamiento.
* Data Warehouse: repositorio de grandes volúmenes de datos donde únicamente se almacenan aquellos datos de interés, posterior a su tratamiento.
* Dataset: conjunto de datos.
* Epochs: iteración completa de la fase de entrenamiento.
* Euskalmet: servicio de meteorología del Gobierno Vasco.
* GanttProject: aplicación opensource diseñada para la planificación de proyectos y conformación de diagramas de Gantt.
* Git: sistema de control de versiones distribuido
* Github: plataforma a modo de repositorio que utiliza la tecnología Git.
* Input: entrada.
* KFold: técnica de validación cruzada.
* Microsoft SQL Server: base de datos estructurada de Microsoft.
* MongoDB: base de datos no estructurada en la que se almacenan documentos.
* MySQL: base de datos estructurada de código abierto.
* Neo4j: base de datos no estructurada de grafos.
* Open Data Euskadi: plataforma del Gobierno Vasco donde se ofrecen datos en abierto.
* Oracle Database: base de datos estructurada de Oracle.
* Output: salida.
* Overfitting: problema donde el modelo no generaliza bien sobre cualquier conjunto de datos, adaptándose demasiado a los datos de entrenamiento.
* Pipeline: proceso automatizado.
* PostgreSQL: base de datos estructurada de código abierto más avanzada que MySQL.
* Postman: herramienta que facilita el uso y testeo de las APIs.
* Power BI: herramienta de visualización que ofrece un amplio catálogo de visualizaciones.
* Pymongo: librería de Python que permite conectar MongoDB a través del código de Python.
* Python: lenguaje de programación de alto nivel.
* R2-Score: métrica entre 0 y 1 que mide la proporción de la varianza, siendo el 1 un ajuste perfecto.
* Random Forest: algoritmo de aprendizaje automático. En este proyecto se ha utilizado su regresión.
* Redis: base de datos no estructurada de clave-valor.
* SAP: ERP de una multinacional alemana que ayudar en la optimización y automatización de procesos.
* Scrum: metodología ágil para la gestión de proyectos y desarrollo de software.
* Snapshot: una copia de seguridad de un momento exacto.
* Sprint: un ciclo de trabajo corto dentro de la metodología Scrum, que tiene un objetivo muy concreto.

Lista de siglas y acrónimos

Por orden alfabético:

* ACID: Atomicity, Consistency, Isolation, Durability
* AEMET: Agencia Estatal de Meteorología
* API: Application Programming Interface
* BI: Business Intelligence
* CAME: Corregir debilidades, Afrontar amenazas, Mantener fortalezas, Explotar oportunidades
* CC: Creative Commons
* CSV: Comma Separated Values
* D3: Data Driven Documents
* DAFO: Debilidades, Amenazas, Fortalezas, Oportunidades
* DGT: Dirección General de Tráfico
* EDA: Exploratory Data Analysis
* ELT: Extract, Load, Tranform
* EM: Expectation-Maximization
* ERP: Enterprise Resource Planning
* ETL: Extract, Transform, Load
* FIA: Federación Internacional del Automóvil
* HTML: HyperText Markup Language
* IA: Inteligencia Artificial
* IoT: Internet of Things
* JS: JavaScript
* JSON: JavaScript Object Notation
* LOPDGDD: Ley Orgánica de Protección de Datos y Garantía de los Derechos Digitales
* MAE: Mean Absolute Error
* MSE: Mean Squared Error
* RGPD: Reglamento General de Protección de Datos
* SQL: Structured Query Language
* SVM: Support Vector Machine
* TFM: Trabajo Fin de Máster
* UE: Unión Europea
* XML: Extensible Markup Language