Máster Universitario en Ingeniería y Ciencia de Datos

Curso académico 2020–2021

Trabajo Fin de Máster

“Arquitectura de un Toolbox integrado de análisis de datos”

Alberto Blázquez Herranz

Tutor/es

Dº Rafael Pastor Vargas

Co-Tutor/es

Dª Ernestina Menasalvas Ruíz

Agradecimientos

A Rafael Pastor Vargas, por acceder a tutorizarme para este Proyecto de Fin de Máster y tomar las riendas de su gestión desde el primer momento.

A Ernestina Menasalvas, por su labor co-tutora de este trabajo. En especial, por su mentoría a nivel profesional durante mi trabajo en el laboratorio MEDAL del Centro de Tecnología Biomédica de la UPM

A quienes quiero: lo hemos conseguido. Ha sido un año realmente especial en tantos sentidos que casi ni me da la cabeza para recapitularlo todo. El camino hasta aquí no ha sido fácil, pero reparar en el momento en el que estoy mientras escribo esto me hace darme cuenta de que también es el año en el que más agradecido puedo estar por tener la vida que tengo a vuestro lado.

Mi familia, en especial papá y mamá: no hay nada que no hayáis hecho por mí. Habéis llegado a saber lo que necesito sin que yo lo sepa, o sin que lo quiera reconocer. Sois mi vida y mis ángeles de la guarda.

Mis amigos: si en algún momento leéis esto, os quiero.

Carlos: eres un hermano para mí.

Lola: si hay una sorpresa mejor que me pueda encontrar en mi camino, dudaría de si esto es la vida real o una fantasía. Ya me ocurre, pero sea lo que sea quiero más.

Contenido

Índice de Ilustraciones

1. Introducción y objetivos del proyecto

En el mundo actual, casi todas las nuevas tecnologías tienen múltiples sensores que miden diversos datos a nuestro alrededor y están provistos de una conexión constante a Internet. En la explotación conjunta de estas características basan la provisión de sus servicios los teléfonos inteligentes, los dispositivos portátiles, el ámbito de la domótica, los coches inteligentes, etc.

Nos referimos al ámbito referido a este tipo de dispositivos como IoT, o “Internet de las cosas” y, aunque abarca múltiples definiciones, es de especial interés la asociada a la operación de dispositivos físicos con capacidad de procesamiento que están conectados entre sí a través de una red. Esto puede implicar sensores, teléfonos, “wearables”, elementos de domótica… cualquier cosa a la que se le pueda añadir un procesador y que esté conectada a Internet puede formar parte del IoT, creando una cantidad ingente de datos brutos provenientes de distintos ámbitos. Esta producción masiva de datos invita a la revisión de otro concepto: el de "Big Data".

Big Data se refiere a los conjuntos de datos que son demasiado grandes y complejos para ser tratados con un software de procesamiento de datos normal. La definición formal más aceptada habla de ello como "una representación de los activos de información caracterizados por un volumen, una velocidad y una variedad tan elevados que requieren tecnología y métodos analíticos específicos para su transformación en valor" [1]. Uno de los mayores problemas del Big Data, además de su volumen, es el hecho de que suele estar desestructurado, lo que impide su tratamiento a través de enfoques de manipulación de datos tradicionales.

A modo ilustrativo del potencial de explotación del Big Data asociado a IoT, el proyecto AutoMat [2] muestra que en el interior de los vehículos actuales se procesan en torno a 4000 señales de tipo CAN-Bus por segundo. Algunos de los datos asociados a estas señales son:

- Asociados a medio ambiente, movilidad, conductor y pasajeros, estado del vehículo… a través de sensores de a bordo.
- Estado de la carretera a través de sistemas de suspensión automática.

En la actualidad, la mayor parte de los datos de este tipo no se utilizan, y la situación es similar en otros ámbitos. Generalmente, la accesibilidad a este tipo de datos es limitada e implica desafíos como la eficiencia de acceso a los mismos y el tipo de estructura que presentan (no estandarizada entre proveedores). Sin embargo, el acceso y la utilización de este tipo de datos implica un gran potencial, pudiendo aportar las siguientes ventajas en la industria automovilística en concreto:

- Integración de señales GPS para la generación y mantenimiento de un mapa de rutas actualizado de forma periódica en la nube.
- Medición de características de la carretera (rugosidad, estado, etc) y generar indicadores relevantes asociados a esta información.
- Asistir el ajuste automático de sistemas integrados en el vehículo (e.g. sistema de suspensión) antes de conducir en función de las características de la vía a transitar.
- Asistir el proceso de mantenimiento de carreteras infiriendo estas necesidades a partir de los datos recogidos.

Alejados del sector de la automoción, encontramos aplicaciones basadas en IoT dentro del marco de las siguientes disciplinas:

- Industria sanitaria: generación de datos anónimos para el análisis médico estadístico.
- Meteorología: observación de variaciones en el clima para aumentar la precisión de los informes y predicciones meteorológicas.
- Infraestructura electrónica y “Smart Cities”: observación de variaciones anómalas en los sensores de la infraestructura de la ciudad para descubrir y tratar fallos de funcionamiento con la mayor rapidez posible.

Estos son sólo algunos ejemplos que demuestran los posibles usos del análisis de Big Data intersectorial generado por diversos dispositivos. Al mismo tiempo, la continua proliferación de dispositivos y sensores de medición de algún parámetro de interés dificulta el análisis de esta plétora de información a través de un enfoque tradicional de almacenamiento y manejo de datos estándar.

Otro desafío que tratar surge del hecho de que los datos procedentes de estos dispositivos evolucionan y cambian constantemente (series temporales), lo que hace

necesario su análisis a partir de una aproximación que tome esto en consideración. Se trata de responder así a la cuestión sobre cómo generar análisis valiosos de datos en evolución (o "streaming data").

De esta cuestión surge la motivación de este proyecto de fin de máster, que se discute en la siguiente sección.

1.1. Motivación

Como se indica en la sección de introducción, los flujos de datos generados por el IoT tienen un enorme potencial, pero en su mayor parte está sin explotar. Para aprovechar esta información, deben existir servicios de análisis que integren estos datos intersectoriales. Para que esto funcione, los datos generados deben estar estructurados de alguna manera común a todas las fuentes. Esto lleva a la conclusión de que es necesario algún tipo de modelo común.

La generación de este modelo de datos común requiere dar respuesta, principalmente, a dos desafíos:

- Especificidad de los datos al sector del que provienen, sin una fácil adaptación de su uso a otro ámbito.
- Aunque un modelo de datos común unifique estos datos en una fuente única, no existen herramientas de análisis que basen su ingesta de datos en una fuente de estas características.

Por ello, la generación de un conjunto de herramientas de análisis de datos se presenta como la solución idónea a estos puntos. Compuesta por servicios que procesen, preparen y analicen flujos de datos en evolución, este "Toolbox" podrá conectarse a múltiples fuentes de datos unificados mediante un modelo de datos común, para proporcionar así estas funcionalidades de análisis.

Estos requerimientos, sumados al desafío que implica a la posibilidad de requerir el tratamiento de datos en constante evolución, separa el "Toolbox" a nivel funcional en dos tipos de servicios:

- Análisis de "streaming data", de modo que los servicios asociados empleen como fuente de datos la conexión a canales de transmisión de datos en

vivo.

- Análisis de “batch data”, o lo que es lo mismo, de conjuntos de datos fijos desde un enfoque tradicional.

Todas estas funcionalidades están dirigidas a la extracción de conocimiento valioso para proveedores de servicios IoT, aunque no de forma exclusiva. Tanto el diseño como la implementación de los servicios se ha llevado a cabo desde una aproximación que habilite su usabilidad en cualquier ámbito, siempre y cuando la naturaleza de los datos lo permita.

Este trabajo se ha desarrollado en el marco del proyecto Cross-CPP [3] (“Cross Sectorial Cyber Physical Products”), un proyecto de la Comisión Europea en el que participan varias instituciones de investigación y de la industria.

1.2. Objetivo

El objetivo principal de este Trabajo de Fin de Máster consiste en el desarrollo de un conjunto de herramientas (“toolbox”) que permitan analizar datos. Se trata de un conjunto de servicios de análisis que en particular se aplicará al ámbito de datos provenientes del Internet de las Cosas (IoT), con aplicación específica a edificios y vehículos inteligentes. No obstante, las librerías y servicios que se diseñarán y desarrollarán son de carácter general, pudiéndose aplicar a cualquier otro tipo de datos.

En particular, y atendiendo a la separación de funcionalidades introducida en la anterior sección, se ha definido el siguiente conjunto de funcionalidades para la herramienta:

- Streaming data
    - Análisis de series temporales: a partir de una entrada que represente el comportamiento enel tiempo de la señal de un sensor dado, se ha desarrollado un conjunto de funciones que permitirá:
        - Realizar predicciones sobre el comportamiento de una señal.
        - Analizar la correlación de varias señales.
        - Detectar la variabilidad de una señal con el fin de detectar comportamientos anómalos (“Drift Detection”).

- Batch data
    - Análisis de trayectorias: dado un conjunto de trayectorias definidas por las coordenadas GPS, se generará un conjunto de funciones que permitan:
        - La agrupación de trayectorias similares.
        - La detección de coordenadas GPS anómalas con respecto a un modelo denormalidad previamente generado.
        - Interpolación lineal de coordenadas dentro de una trayectoria.
        - Obtención de estadísticas básicas propias de una trayectoria (velocidad media,distancia recorrida y tiempo de trayecto).
    - Análisis de redes: a partir de un conjunto de señales de sensores y su evolución en el tiempo, se representará una red de nodos (sensores) conectados. Habilitará la identificación de grupos de sensores que muestran una evolución similar. Para ello,esta función proveerá las siguientes métricas asociadas a la red:
        - Excentricidad de los nodos: medida individual de la máxima distancia de un nodo a cualquiera de los demás. Indicador de la cantidad de variabilidadpresente en un nodo.
        - Eficiencia de la red, como métrica de la bondad con la que la información fluye a través de la red. Permite la evaluación la similitud de la evolución de un conjunto de sensores.
        - Diámetro de la red: distancia máxima existente entre cualquier par de nodos. Informa acerca de cómo de compacta es una red, o como de similares son sus nodos.

2. Estado del arte

De acuerdo con el carácter fundamentalmente práctico de este proyecto de fin de máster, el análisis de los trabajos relacionados que se ofrece a continuación se centrará en la inspección de la eficiencia de los enfoques disponibles y su idoneidad para la satisfacción de los objetivos definidos, de modo que no se detendrá en el aspecto científico y los antecedentes de investigación de los mismos.

Con respecto a la estructura de este apartado, dada la amplia gama de funcionalidades propuestas para la aplicación desarrollada, se ha dedicado esta sección a proveer detalle con respecto a conceptos y técnicas que estas implican. En consonancia con la agrupación de funcionalidades planteada en la sección dedicada a la definición de objetivos, esta revisión se estructurará de forma análoga.

En primer lugar, se discutirán las nociones relativas al análisis de datos en evolución ("streaming data") y, a continuación, se hará para el análisis de conjuntos de datos fijos. Cada una de estas secciones circunscribe su planteamiento en torno a las funcionalidades del "Toolbox" que tiene asociadas. Así, la sección de datos en evolución se enmarcará en el ámbito del análisis de series temporales y el apartado de conjuntos de datos fijos se asociará al análisis de trayectorias y de redes complejas.

2.1. Datos en evolución

2.1.1. Series temporales

Los flujos de datos [4] pueden considerarse procesos estocásticos en los que los eventos se producen de forma continua y con independencia unos de otros. El acceso a este tipo de datos contrasta con el requerido para los modelos de datos relacionales tradicionales por las características que definen su naturaleza:

- La recolección de información requiere de un canal dedicado, conectado a internet, del que extraer los datos.
- No existe control sobre el orden de llegada de los datos.
- El tamaño agregado de los datos recibidos por un canal de "streaming" es potencialmente ilimitado.
- Una vez que un elemento del "streaming" ha sido procesado, no puede ser recuperado fácilmente a menos que se almacene explícitamente en

memoria (inasumible de forma continua en términos de recursos computacionales).

Tomando estas características en consideración, el análisis de "streaming data" [5], tiene como objetivo la predicción de la clase o el valor de las nuevas instancias en el flujo de datos dado el conocimiento que proporcionan las instancias previas en el flujo de datos. Las características de esta noción se pueden vincular intuitivamente con el estudio de series temporales [6] es una serie de valores numéricos que representan la evolución de una cantidad específica a lo largo del tiempo. Tales secuencias de variables aleatorias pueden expresarse matemáticamente para analizar su comportamiento, generalmente para entender su evolución pasada y predecir su comportamiento futuro.

2.1.1.1. Funcionalidades relevantes

Esta sección pretende recolectar las nociones principales con respecto al desarrollo de funciones de análisis de series temporales que ha tenido lugar a lo largo de este proyecto de fin de máster. Se han agrupado en función de su orientación a la predicción del comportamiento de una serie, el estudio de correlación o evaluación de su variabilidad ("Drift Detection").

- **Predicción**:
  - **Modelo ARIMA** [7] método estadístico popular y ampliamente utilizado para la previsión de series temporales. ARIMA es un acrónimo que significa:
    - AR: Autoregresión. Es un modelo que utiliza la relación de dependencia entre una observación y un cierto número de observaciones históricas.
    - I: Integrado. El uso de la diferenciación de las observaciones en bruto para que la serie temporal sea estacionaria.
    - MA: Media móvil. Un modelo que utiliza la dependencia entre una observación y un error residual de un modelo de media móvil aplicado a las observaciones retardadas.

    Cada uno de estos componentes se especifica explícitamente en el modelo como un parámetro. Se utiliza una notación estándar de ARIMA(p,d,q) donde los parámetros del método ARIMA se definen como sigue:

- p: número de observaciones de retardo incluidas en el modelo, también llamado orden de retardo.
- d: número de veces que se diferencian las observaciones brutas. También se denomina grado de diferenciación.
- q: tamaño de la ventana de la media móvil, también llamado orden de la media móvil.
- P: parámetro equivalente a “p”, aunque referido al componente estacionario de la serie temporal bajo estudio.
- D: parámetro equivalente a “d”, aunque referido al componente estacionario de la serie temporal bajo estudio.
- Q: parámetro equivalente a “q”, aunque referido al componente estacionario de la serie temporal bajo estudio.

Se construye un modelo de regresión lineal que incluye el número y el tipo de términos especificados y el tipo de términos, y los datos se preparan mediante un grado de diferenciación para hacerlos estacionarios, es decir, para eliminar la tendencia y las estructuras estacionales que afectan negativamente al modelo de regresión.

La adopción de un modelo ARIMA para una serie temporal supone que el proceso subyacente que generó las observaciones es un proceso ARIMA. Esto puede parecer obvio, pero ayuda a motivar la necesidad de confirmar los en las observaciones brutas y en los errores residuales de las previsiones del modelo.

- **Modelo de redes neuronales** [5]: sistema cuyo diseño se inspira originalmente de forma esquemática inspirado en el funcionamiento de las neuronas biológicas.

Una red neuronal se compone generalmente de una sucesión de capas cada de las cuales toma sus entradas en las salidas de la anterior. Cada capa (i) está compuesta por Ni neuronas, que toman sus entradas en las Ni-1 neuronas de la capa anterior. Cada sinapsis está asociada a un peso sináptico de modo que las Ni-1 se multiplican por este peso, y luego se suman por el nivel i, lo que equivale a multiplicar el vector de entrada por

una matriz de transformación. Poner las diferentes capas de una red neuronal una detrás de otra equivaldría a poner en cascada varias matrices de transformación de transformación y podría reducirse a una sola matriz, producida por otras si no hubiera en cada capa, la función de salida que introduce no linealidad en cada paso. Esto demuestra la importancia de la elección juiciosa elección de una buena función de salida: una red de neuronas cuyas salidas fueran lineales no tendría ningún interés.

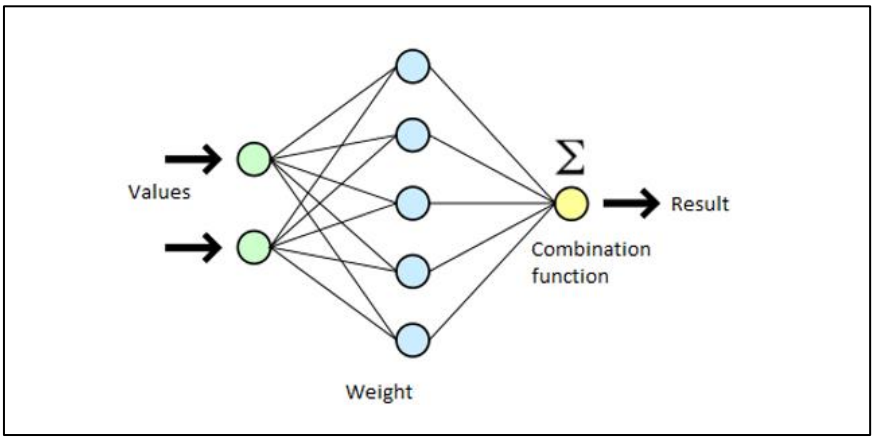

Ilustración 1. Representación de una red neuronal con 8 neuronas en 3 capas.

- **Modelo de árbol de regresión**: método basado en el uso de un árbol de decisión como modelo de predicción. El aprendizaje de árboles de regresión consiste en construir un árbol a partir de un conjunto de aprendizaje formado por n-tuplas etiquetados. Un árbol de decisión puede describirse como un diagrama de flujo en el que cada nodo interno describe una prueba sobre una variable de aprendizaje, cada rama representa un resultado de la prueba y cada hoja contiene el valor de la variable numérica.

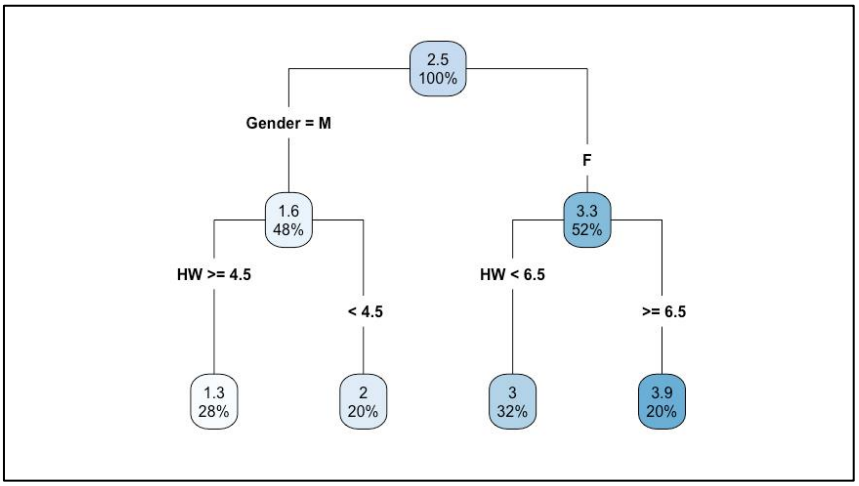

Ilustración 2. Ejemplo de un árbol de decisión.

Normalmente, el árbol de regresión se utiliza sobre una serie temporal estacionaria.

- **Análisis de correlación:**

  - **Correlación de Pearson** [8]: medición en el intervalo [-1, 1] que indica la medida en que dos variables están relacionadas linealmente.

    Asume diferentes características en los datos como la normalidad de las distribuciones de homocedasticidad, se pueden obtener valores inconsistentes en los datos no cumplen estos requisitos.
  - **Correlación de Spearman** [9]: esta métrica está orientada a la evaluación de la asociación ordinal de dos variables distintas, así como a la valoración de si dos variables pueden describirse o no mediante una función monótona. Es equivalente a la correlación de Pearson entre los valores de rango.

- **"Drift Detection":**
  - **Irreversibilidad:** métrica para la evaluación del grado de irreversibilidad de una serie temporal. Una señal se define como reversible si no hay diferencias estadísticas entre la señal original y su versión invertida en el tiempo; por el contrario, se define como irreversible si alguna de sus propiedades estadísticas puede utilizarse para definir una flecha natural del tiempo.

    El interés de esta métrica reside en su relación con la memoria. En

general, si una serie temporal es irreversible, suele significar que el sistema subyacente tiene memoria, es decir, que los valores futuros están limitados por los pasados.

- **Entropía de permutación** ("Permutation Entropy"): métrica basada en la comparación de valores vecinos en una serie temporal, para luego encontrar los patrones de orden que resultan en secuencias ordenadas (ascendentes), y finalmente en el estudio de la distribución de probabilidad correspondiente [10].

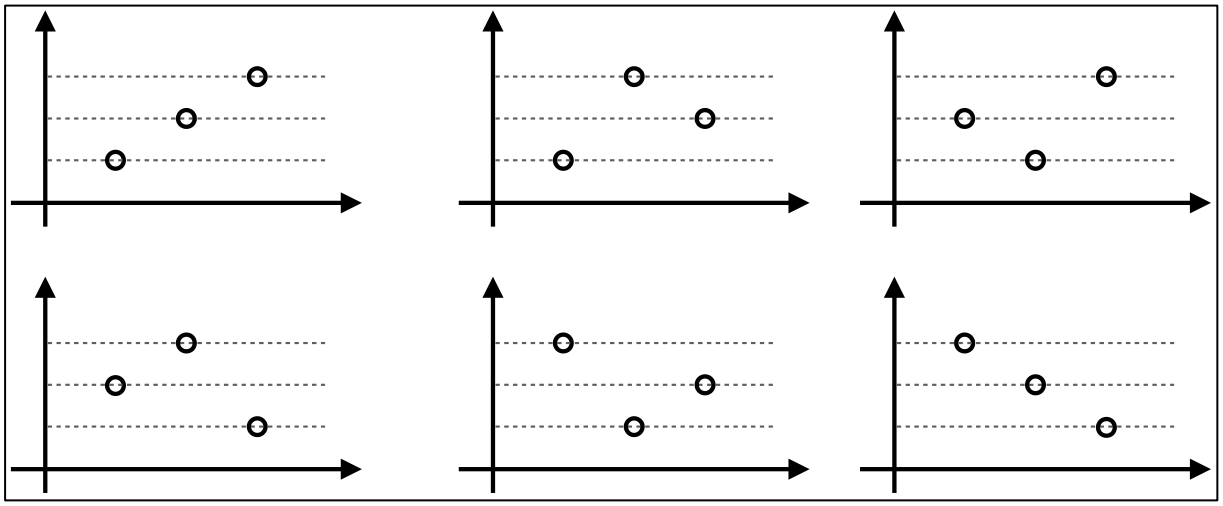

Ilustración 3. Ejemplo de seis patrones de permutación distintos que pueden encontrarse en un segmento de una serie temporal.

Más concretamente, supongamos que nos interesa analizar las relaciones entre el subconjunto de tres valores que componen la serie temporal. Su magnitud relativa puede dar lugar a sólo 6 patrones de permutación. Se puede entonces contar cuántas veces se ha observado cada patrón y, finalmente, calcular la entropía de dicha distribución.

- **Entropía muestral** ("Sample Entropy"): métrica orientada a la estimación de la complejidad de una serie temporal. De manera sumariada, esta métrica mide lo predecible que es una serie temporal, evaluando si algunos patrones aparecen de forma regular.

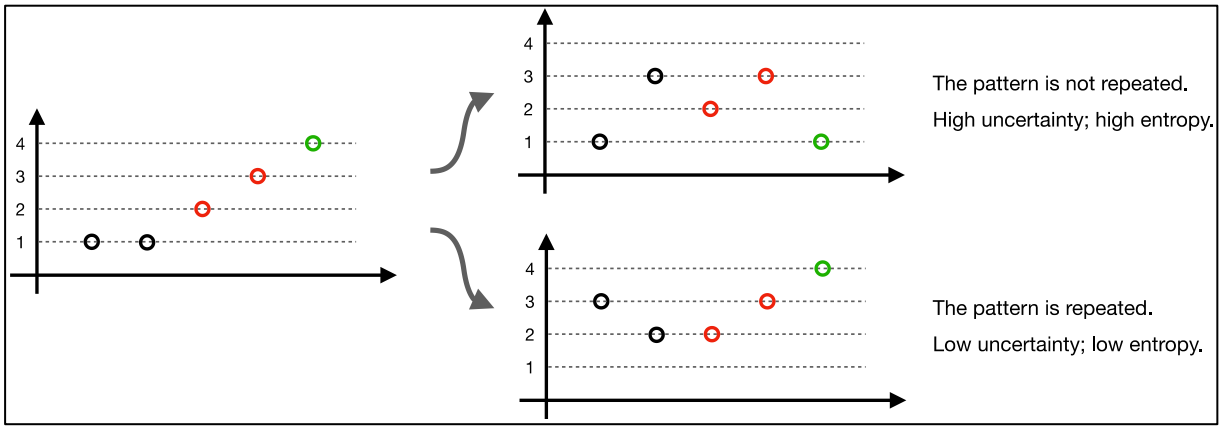

Ilustración 4. Representación gráfica de la noción de entropía muestral.

En la anterior figura, el panel de la izquierda presenta un segmento de la serie temporal analizada, en el que se observa la presencia de un patrón, en concreto, que a los valores 2 y 3 (círculos rojos) les sigue un 4 (círculo verde). Existen entonces dos posibilidades: o bien aparece el mismo patrón con frecuencia, como en el panel inferior derecho; o bien no aparece, es decir, se encuentra un valor diferente después de un 2 y un 3 (panel superior derecho). En el primer caso, la serie temporal es muy predecible y, por tanto, se obtendrá una entropía baja; mientras que en el segundo caso ocurre lo contrario.

2.2. Conjuntos de datos fijos

2.2.1. Trayectorias

Como se ha descrito en la introducción, la industria del automóvil es uno de los sectores en los que se genera una gran cantidad de datos en lo que al ámbito IoT respecta. Uno de los tipos importantes de datos que se miden en esta industria son las coordenadas GPS a lo largo del tiempo, producidas por los sensores en los vehículos en movimiento. Los datos generados son series temporales de coordenadas GPS que pueden utilizarse para representar las trayectorias que han recorrido estos los vehículos que los portan. Es este contexto en torno al que se plantea el análisis de trayectorias como componente del "Toolbox" desarrollado en este proyecto. Así, el módulo de análisis asociado a este apartado proporciona un conjunto de funciones para explotar la información intrínseca a este formato de información. A continuación, se plantean los conceptos fundamentales asociados a la noción de trayectoria con respecto a la cual se ha trabajado, así como los fundamentos correspondientes al análisis de trayectorias.

2.2.1.1. Conceptos Básicos

- **GPS**, o sistema de posicionamiento global, es un sistema de geo navegación basado en satélites. En lo que respecta al desarrollo del presente trabajo de fin de máster, la mención de coordenadas GPS se referirá a valores de latitud y longitud.
- **Trayectoria**, entendida como una serie temporal formada por coordenadas GPS, cada una de las cuales ha sido generada en un momento específico del tiempo. Así, se trata de una sucesión de pares latitud-longitud a través de la dimensión temporal.

Es importante mencionar la necesaria relación del análisis de trayectorias con las series temporales. Como serie de puntos de datos indexados en orden temporal, se corresponden con el formato de datos ya detallado en el apartado “2.1.1”, aunque se han separado de dicho módulo dadas las particularidades intrínsecas al tratamiento de datos GPS.

2.2.1.2. Funcionalidades relevantes

El campo del análisis de trayectorias ha sido sujeto de investigación durante mucho tiempo, de modo que la información disponible es abundante y actualizada. A continuación, se listan los campos principales de esta disciplina en torno a los que se circunscribe el desarrollo del “Toolbox”, junto con las referencias correspondientes cuando sean pertinentes.

- **Estadísticas simples**, como el cálculo de las velocidades y aceleraciones.
- **Interpolación de trayectorias**, es decir, la creación de una nueva trayectoria con una mayor (y constante) resolución de tiempo, utilizando los datos disponibles para derivar la posición cuando no se conoce [11].
- **Detección de anomalías**, es decir, la detección de las posiciones comunicadas que no cumplen con las limitaciones físicas (por ejemplo, la velocidad máxima) o con la dinámica pasada del CPP [12].
- **Clustering de trayectorias**, es decir, la identificación de subgrupos de trayectorias con alta similitud [13].

2.2.1.3. Algoritmos empleados

Esta sección cubrirá los algoritmos existentes que se utilizaron para implementar las funcionalidades del módulo de análisis de trayectorias del “Toolbox”.

- **Distancia de Haversine**: definida a partir de un caso especial de la ley de Haversines [14], una ley de la trigonometría esférica. Esta fórmula determina la distancia entre dos puntos de una esfera. Por lo tanto, es utilizable en el análisis de trayectorias para determinar la distancia entre dos puntos del globo o, en el contexto que ocupa a este trabajo de fin de máster, entre dos coordenadas GPS.
- **QuickBundles Clustering** (QB) [15]: se trata de un algoritmo de agrupación con un rendimiento de tiempo lineal con respecto al número de trayectorias.

El tiempo de cálculo de QB depende linealmente del número de muestras porque (i) utiliza MDF con complejidad lineal y (ii) no hay reasignación de la agrupación de trayectorias. Esto significa que las trayectorias se evalúan una sola vez al realizar el clustering y, a diferencia de K-means, donde hay una fase de amalgama. Los clusters se definen mediante un triplete c = (I, h, n), donde I es una lista de índices de las trayectorias en el cluster, n es el número de trayectorias en el cluster, y h es la suma de trayectorias (h es una matriz Kx2 que puede ser actualizada para las nuevas trayectorias).

$$h = \sum s_i$$

De este modo, el centroide del cluster queda definido por v:

$$v = \frac{h}{n}$$

Entre los puntos fuertes de este algoritmo de clustering, tenemos su mencionada complejidad lineal complejidad lineal, la facilidad para interpretar los clusters y la versatilidad que proporciona el cambio del número de puntos que definen cada trayectoria y el valor del umbral para considerar que las trayectorias son similares. Todo ello resulta muy útil en la exploración de un conjunto de datos de trayectorias.

Sin embargo, QB fue desarrollado para superar la complejidad de la segmentación de las fibras. Para aplicarlo a las trayectorias de cluster, se requiere lo siguiente:

- Todas las trayectorias tienen que estar definidas por el mismo número de puntos.
- Un umbral de distancia para especificar el valor máximo a partir del cual que dos trayectorias se agrupan.

2.2.2. Redes complejas

Una gran variedad de sistemas de la naturaleza, la sociedad y la tecnología pueden modelarse como grafos de vértices conectados por aristas [6].

De acuerdo con el planteamiento de Steven H. Strogatz en su artículo “Exploring Complex Networks” [16], los sistemas existentes en nuestro mundo que pueden representarse y estudiarse como una red pueden incluir: las redes de energía eléctrica, los grafos de llamadas telefónicas, las redes de coautoría y citación de investigadores, la World-Wide Web, etc. Y como los ordenadores potentes son cada vez más comunes y están más extendidos, es posible representarlos y realizar cálculos complejos con redes de millones de nodos. En cualquier caso, el desafío principal reside en la decisión acerca de cómo implementar estas representaciones [16]. Las redes y sus nociones asociadas entrañan una alta complejidad para su comprensión, de modo que el autor enumera una serie de puntos clave para tener en cuenta durante el diseño de una red:

- **Complejidad estructural**: ilegibilidad del diagrama asociado a la red por ser una complicada maraña de aristas entrecruzadas.
- **Evolución de la red**: el diagrama de cableado podría cambiar con el tiempo. Como ejemplo, la World-Wide Web está sujeta a la creación y eliminación de páginas y enlaces a cada momento.
- **Diversidad de conexiones**: los enlaces entre nodos pueden tener diferentes pesos y sentidos.
- **Diversidad de nodos**: puede haber muchos tipos diferentes de nodos.

El desarrollo del “Toolbox” planteado para este proyecto de fin de máster ha tenido en cuenta los anteriores desafíos para generar un módulo que ofrezca funcionalidades concretas para el uso de redes complejas en el ámbito de las aplicaciones IoT. Para ello, primero hay que definir ciertos conceptos.

Como medios de inspección de los conceptos relevantes asociados al análisis de redes complejas, se han tomado como referencia las publicaciones “Analyzing and modelling Real-World Phenomena with Complex Networks: A Survey of Applications” [17], “Combining complex networks and data mining: Why and how” [18] y ”Temporal Networks” [6]. A continuación, se presentan las definiciones de los conceptos básicos tomadas de dicho artículo.

2.2.2.1. Conceptos básicos

- Representación de redes:
    - **Grafo** representativo de la red (G): es un par ordenado $G = (N, E)$,

formado por un conjunto $N = n_1, n_2, \dots n_n$ de vértices (o nodos, o puntos) y un conjunto $E = e_1, e_2, \dots e_n$ de aristas o enlaces $e_k = n_i, n_j$ que conectan los vértices.

En un grafo no dirigido las aristas e_{ik} y e_{ji} que conectan los nodos i y j, son lo mismo. En un grafo dirigido, que tiene en cuenta el sentido de las aristas que conectan dos grafos, este no es el caso. En el desarrollo de este proyecto se utilizan grafos no dirigidos.

- **Recorrido** ("walk") de longitud k: secuencia alterna no vacía $n_0 e_0 \dots e_{k1} n_k$ de vértices y aristas de G tal que $e_i = n_i, n_{i+1}$para todo $i < k$. Si $n_0 = n_k$, el recorrido es cerrado.
- **Camino** ("path") entre dos nodos es un recorrido por los nodos de la red en el que cada nodo se visita una sola vez. Si un camino lleva de n_i a n_j, entonces se dice que n_j es un sucesor de n_i, y n_i es un predecesor de n_j.
- **Matriz de adyacencia**: representación alternativa de un grafo en la que los elementos de la matriz A a_{ij} son iguales a 1 siempre que haya una arista que conecte los nodos i y j, e iguales a 0 en caso contrario.
- **Intensidad de las conexiones**: representadas en el grafo mediante la asociación de valores numéricos denominados pesos a las aristas.

- Medidas asociadas a una red:
  - **Longitud** de un camino en un grafo ponderado es la suma de los pesos de las aristas a lo largo del camino.
  - **Camino más corto**, también llamado distancia geodésica, se define como la menor suma de las longitudes de las aristas a lo largo de todos los caminos posibles desde i hasta j.
  - **Excentricidad** del nodo es la longitud del más largo de todos los caminos más cortos que parten del nodo dado.

    Para toda la red, es posible representar las distancias geodésicas mediante una matriz de distancia D, en la que la entrada d_{ij} representa la longitud de los caminos más cortos entre los nodos i y j.

- **Diámetro** de la red es la longitud del más largo de todos los caminos más cortos calculados en una red, es decir, el valor máximo de la matriz de distancia.
- **Eficiencia** de la red es una medida que describe la eficiencia con la que la información puede fluir a través de la red. Representa el valor medio de la inversa de la distancia geodésica entre pares de nodos, calculada mediante la fórmula

$$\frac{1}{N * (N-1)} * \sum \frac{1}{d_{ij}}$$

También hay otras medidas de la red, pero éstas son las relevantes para los fines de este trabajo.

2.2.2.2. Aplicaciones en Minería de Datos

Como ya se ha dicho, el análisis de redes complejas es relevante y puede aplicarse a distintos escenarios de la vida real. Pero el propósito de este proyecto es el análisis de datos en evolución, por lo que se centrará en la utilización de redes complejas con respecto a la minería de datos. Tanto el análisis de redes como la minería de datos se han aplicado para abordar el problema de la extracción de información significativa a partir de conjuntos de datos cada vez mayores, pero sorprendentemente pocos trabajos los han utilizado simultáneamente. Se puede suponer que se hacen obsoletos unos a otros, pero no es así. Un sencillo experimento numérico demostrado en la página 18 de [18] muestra que los algoritmos de minería de datos no suelen ser suficientes para tratar la estructura de un sistema suficientemente complejo. Algunas de las razones son las siguientes:

- Los algoritmos de minería de datos están optimizados para extraer las relaciones entre un pequeño conjunto de características, pero no para analizar las interacciones a escala global. Por ejemplo, el cálculo de la eficiencia de una red requiere la estimación de cada uno de los caminos más cortos dentro de ella; una tarea que no puede ser realizada únicamente por un algoritmo de minería de datos.
- La mayoría de los algoritmos de minería de datos no manejan bien un enfoque estadístico. En su lugar, están optimizados para detectar patrones entre conjuntos fijos de características.

Así pues, el uso del análisis de redes en relación con el análisis de datos en evolución es un área deseada y poco desarrollada. Siguiendo esta premisa, el desarrollo de herramientas de análisis de redes es un paso necesario y útil para el análisis de datos evolutivos.

2.2.2.3. Algoritmos empleados

Esta sección se centrará en los algoritmos existentes que se utilizaron para la implementación de los servicios. Dado que el servicio implementado para este módulo del proyecto está asociado a la mayoría de las métricas generales y básicas, sólo fue necesario implementar un algoritmo conocido. Los demás cálculos se realizaron con cálculos rudimentarios, sin seguir ningún algoritmo específico.

El algoritmo utilizado es el algoritmo de Dijikstra, también conocido como algoritmo del camino más corto de Dijkstra o algoritmo SPF.

El objetivo del algoritmo de Dijikstra es encontrar el camino más corto entre los nodos de un grafo ponderado [19]. A continuación, se muestra una descripción del pseudocódigo del algoritmo de Dijkstra.

Algorithm 1 Algorirmo de Dijkstra

```
procedure DIJKSTRA(Grafo, fuente)
    Crear conjunto de vértices Q
    for all vértice v en Grafo do
        dist[v] ← INFINITO
        prev[v] ← N/A
        añadir v a Q
    end for
    dist[fuente] ← 0
    while Q no está vacío do
        u ← vértice de Q con mínimo dist[u]
        eliminar u de Q
        for all vecino v de u do
            alt ← dist[u] + length(u, v)
            if alt < dist[v] then
                dist[v] ← alt
                prev[v] ← u
            end if
        end for
    end while
    return dist[], prev[]
end procedure
```

Ilustración 5. Pseudocódigo del algoritmo de Disjkstra.

3. Descripción de la solución

La presente sección recopila los detalles de implementación de la aplicación desarrollada.

Para satisfacer los objetivos planteados para el proyecto, se ha generado un conjunto de servicios ("Toolbox") agrupados en 3 APIs separadas, cada una de las cuáles corresponde con cada uno de los ámbitos de análisis definidos en la introducción: series temporales, análisis de trayectorias y análisis de redes complejas. A continuación, se muestra

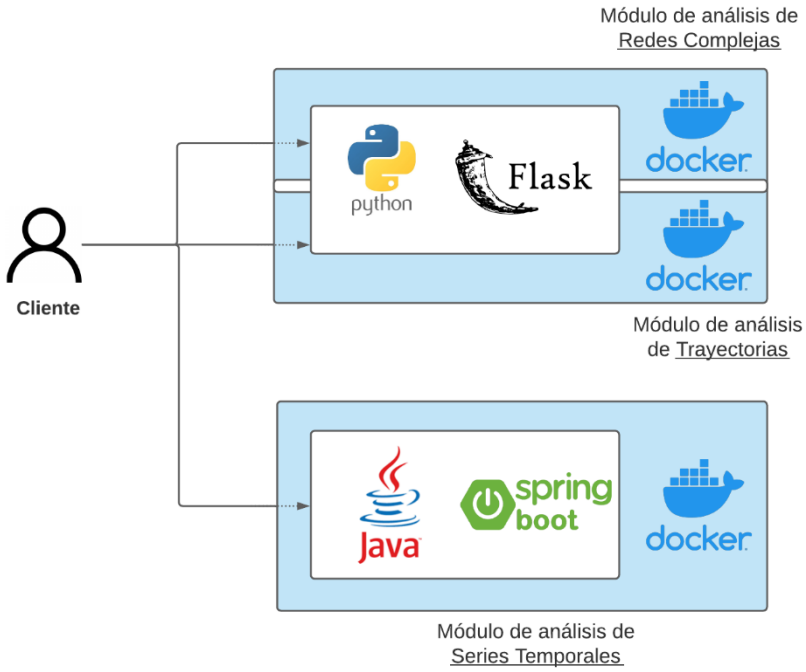

Ilustración 6. Detalle del conjunto de tecnologías asociadas a los servicios desarrollados.

La anterior figura detalla el conjunto de tecnologías empleado para cada módulo:

- APIs de análisis de trayectorias y redes complejas: emplean Python para la codificación del flujo de datos y manipulación de datos y Flask como framework para la definición de los endpoints de cada APIs.
- API de análisis de series temporales: se ha desplegado como una aplicación de Java haciendo uso del framework de Spring Boot para la definición de los endpoints de su API.

La entidad “cliente” presente en el gráfico se refiere a cualquier consumidor de servicios que desee operar con las APIs del “Toolbox”. Puede tratarse de un servicio web que genere visualizaciones con las salidas recibidas, un usuario individual que envíe peticiones a las APIs… Su uso no está restringido a ningún tipo de entorno definido.

Con respecto al despliegue de la solución, se puede observar que se ha optado por un enfoque de “containerization” haciendo uso de Docker. Esta decisión de diseño se soporta en los siguientes puntos:

- **Consistencia**: dado su carácter aislado del sistema que lo hospeda, el despliegue de aplicaciones “dockerizadas” es independiente del resto de procesos en ejecución. De este modo, se evitan problemas de dependencias solapadas o incompatibilidades, además de acotar la depuración de problemas al contenedor en el que se ejecuta la aplicación.
- **Replicabilidad**: el mantenimiento de este tipo de infraestructuras es fácilmente controlable desde el fichero de configuración del despliegue. Esto facilita la migración del servicio entre entornos (por ejemplo, de desarrollo a producción).
- **Adaptación al paradigma de “Continuous Integration”:** la configuración del despliegue de un contenedor puede estar ligada a un repositorio de código como Github, de modo que la integración de cualquier modificación en la fuente es directa.

3.1. Módulo de análisis de series temporales

Este módulo habilita una serie de funciones orientadas a la inspección y predicción de series temporales, esto es, datos estructurados la largo del tiempo (sean unidimensionales o bidimensionales). A continuación, se puede ver un detalle de los endpoints diseñados para la API asociada a este módulo:

Endpoint	URL	request type	parameters	result
Process Data	crosscpp/networks/create/	POST	A data model (as defined in the documentation) and, if performed for the first time, an AEON Subscription URL.	Output of the required analysis as specified in the data model received. If performed for the first time, the following operation is performed in the server: subscription to the AEON Subscription Url provided in the request will take place (through which upcoming data models will be processed).

Ilustración 7. Especificación de la API del módulo de análisis de series temporales.

La principal particularidad de la API este módulo es que, como se indicó en las funciones introductorias, está asociado a datos en evolución o “streaming data”. Como tales, su funcionamiento requiere de la conexión a un canal de “streaming” en el que los datos son publicados en tiempo real por los proveedores de servicios IoT. De ahí que el único endpoint disponible sea de subscripción a un canal de “streaming data”. Se detallan con mayor especificidad los parámetros que deben especificarse en la petición:

- **Model**: keyword asociada al tipo de análisis a ejecutar sobre el canal de “streaming data”. Es específico de cada funcionalidad.
- **variables:** colección de parámetros a ajustar del modelo seleccionado, si aplica. Es específico de cada funcionalidad.
- **data:** general, aplica a todas las funcionalidades.
    - Porporcionado para cada trayectoria especificada en el cuerpo JSON de la petición.
    - Array de tuplas de acuerdo con el siguiente formato:
        - **{"value":** <numeric>, **"timestamp":"**<date>**"}**
        - **Formato de fecha -** YYYY-mm-ddTHH:MM:SS.Z
- **id:** identificador textual del canal de “streaming data”. General, aplica a todas las funcionalidades.
- **subscriptionURL:** URL del canal de “streaming data”. General, aplica a todas las funcionalidades.

A continuación, se muestra un ejemplo de cuerpo JSON requerido para establecer la subscripción a una canal:

Ilustración 8. Ejemplo de entrada para la generación de un proceso de análisis de series temporales.

El ejemplo de la imagen anterior pretende tener un carácter generalista. A continuación, se mostrarán los detalles específicos de cada una de las peticiones asociadas a cada funcionalidad que provee este módulo.

3.1.1. Entropía muestral

Proporciona una medida probabilística de la complejidad de una serie temporal.

Dada una dimensión específica e y una tolerancia t, el algoritmo devuelve la probabilidad (como logaritmo negativo) de que si dos conjuntos de e puntos de datos simultáneos tienen una distancia t, dos conjuntos de (e+1) puntos de datos simultáneos también tendrán una distancia t.

Su keyword para el campo “model” es “SampleEntropy” y los parámetros ajustables de esta función bajo el identificador “variables” son:

- **embeddingDimension:** número de puntos a considerar en cada conjunto.
- **tolerance:** distancia estimada entre cada conjunto de datos contiguo.
- **normalized:** normalizar o no la serie temporal.

3.1.1.1. Salida

El resultado es un valor numérico asociado a la métrica de entropía muestral que corresponda a las trayectorias provistas, de acuerdo a los parámetros ajustados.

3.1.2. Entropía de permutación

Medida de la complejidad de una serie temporal.

Dado el patrón ordinal de conjuntos secuenciales de **e** puntos (cada uno espaciado por **d** puntos del otro) en una serie temporal, mide la variabilidad de su comportamiento según todos los posibles cambios en el patrón ordinal de cada conjunto.

Su keyword para el campo "model" es "PermutationEntropy" y los parámetros ajustables de esta función bajo el identificador "variables" son:

- **embeddingDimension:** número de puntos a considerar en cada conjunto.
- **delay:** número de puntos que deben separar un conjunto de otro.
-

3.1.2.1. Salida

El resultado es un valor numérico asociado a la métrica de entropía de permutación que corresponda a las trayectorias provistas, de acuerdo a los parámetros ajustados.

3.1.3. Irreversibility

Medida cuantitativa del comportamiento de una serie temporal cuando sus "timestamps" son invertidos. Se computa sobre un subconjunto de datos de la serie temporal completa.

Su keyword para el campo "model" es "Irreversibility" y los parámetros ajustables de esta función bajo el identificador "variables" son:

- **embeddingDimension:** número de puntos a considerar.

3.1.3.1. Salida

El resultado es un valor numérico asociado a la métrica de irreversibilidad que corresponda a las trayectorias provistas, de acuerdo a los parámetros ajustados.

3.1.4. PearsonCorrelation

Esta funcionalidad, cuyas particularidades ya han sido descritas en la sección introductoria, no tiene parámetros a ajustar bajo el identificador “variables”. Su keyword para el campo “model” es “PearsonCorrelation”.

3.1.4.1. Salida

El resultado es un valor numérico asociado a la medida de correlación de Pearson de las dos series temporales provistas como entrada.

3.1.5. SpearmanCorrelation

Esta funcionalidad, cuyas particularidades ya han sido descritas en la sección introductoria, no tiene parámetros a ajustar bajo el identificador “variables”. Su keyword para el campo “model” es “SpearmanCorrelation”.

3.1.5.1. Salida

El resultado es un valor numérico asociado a la medida de correlación de Spearman de las dos series temporales provistas como entrada.

3.1.6. Árbol de regresión

Las particularidades de esta funcionalidad han sido discutidas en suficiente detalle en la sección introductoria.

Su keyword para el campo “model” es “RegressionTree” y los parámetros ajustables de esta función bajo el identificador “variables” son:

- **forecast:** número de puntos a predecir.
- **division:** índice dentro de la serie temporal a partir del cual realizar la divisón entre datos de entrenamiento y de test.

3.1.6.1. Salida

El resultado es un valor array con tantos valores numéricos como datos se hayan requerido predecir en el parámetro “forecast”.

3.1.7. NeuralNetwork

Las particularidades de esta funcionalidad (basada en un Perceptrón Multicapa) han sido discutidas en suficiente detalle en la sección introductoria.

Su keyword para el campo “model” es “NeuralNetwork” y los parámetros ajustables de esta función bajo el identificador “variables” son:

- **forecast:** número de puntos a predecir.
- **division:** índice dentro de la serie temporal a partir del cual realizar la divisón entre datos de entrenamiento y de test.

3.1.7.1. Salida

El resultado es un valor array con tantos valores numéricos como datos se hayan requerido predecir en el parámetro “forecast”.

3.1.8. Arima

Las particularidades de esta funcionalidad han sido discutidas en suficiente detalle en la sección introductoria.

Su keyword para el campo “model” es “Arima” y los parámetros ajustables de esta función bajo el identificador “variables” son:

- **forecast:** número de puntos a predecir.
- **arimaOrder:**
  - **p**: asociado a la parte regular autoregresiva. Número de observaciones de retardo a incluir en el modelo.
  - **d**: asociado a la parte de integración del modelo. Número de veces que las observaciones originales son diferenciadas.
  - **q**: tamaño de la ventana de “moving average”.

3.1.8.1. Salida

El resultado es un valor array con tantos valores numéricos como datos se hayan requerido predecir en el parámetro “forecast”.

3.1.9. Detalles de implementación

Este módulo ha sido desarrollado en Java, versión 1.8.0. Se han utilizado las siguientes bibliotecas estándar:

- MOA, versión 2019.04.1: librería utilizada en la construcción del modelo de propiedades estadísticas, y en el modelo que evaluará la estructura de la relación entre múltiples series temporales.
- Signaflo, versión 0.4: librería utilizada en la construcción del modelo de aprendizaje automático predictivo, y para la implementación de los algoritmos AR/MA/ARIMA.
- Spring Boot: librería utilizada en el desarrollo del módulo.

3.2. Módulo de análisis de trayectorias

Este módulo pretende ofrecer a los proveedores de servicios un conjunto de herramientas para el tratamiento de los datos de las trayectorias, es decir, datos asociados a coordenadas geoespaciales (longitud-latitud) que codifique una evolución espacio-temporal, más concretamente los desplazamientos de los coches. A continuación, se puede ver un detalle de los endpoints diseñados para la API asociada a este módulo.

Endpoint	URL	request type	parameters	result
Simple Statistics	/crosscpp/trajectories/statistics/	POST	a json array with trajectory objects. Each trajectory object should have 'data' as required key.	a json object. Key "result" with value of "success" or "failure". in case of success, json array 'response', each entry of which consists of "distance", "duration" and "velocity"
Clustering	/crosscpp/trajectories/clustering/	POST	a json array with trajectory objects. Each trajectory object should have 'data' as required key.	a json object. Key "result" with value of "success" or "failure". in case of success, a list with key "clusterLabels" containing cluster labels for each trajectory from the request, corresponding to their index in the list.
Anomaly Detection	/crosscpp/trajectories/anomaly_detection/	POST	a json array with trajectory objects. Each trajectory object should have 'data' as required key. An optional key 'threshold-type' can be provided to specify the method with which to identify outliers ('iqr' for interquarle range and 'std' for standard deviation - default).	a json object. Key "result" with value of "success" or "failure". in case of success, json array 'response', each of which entries contain the values contained in 'data' and an additional 'abnormality-rate' key as a quantitative measure of the detected outliers' incidence.
Interpolation	/crosscpp/trajectories/interpolation/	POST	a json array with trajectory objects. Each trajectory object should have 'data' and 'time-resolution' as required keys. 'time-resolution' must be an integer greater than 0.	a json object. Key "result" with value of "success" or "failure". in case of success, json array 'response', each of which entries contain the interpolated coordinates.

Ilustración 9. Especificación de la API del módulo de análisis de trayectorias.

3.2.1. Estadísticas generales

La funcionalidad de estadísticas proporciona una serie de propiedades de resumen para una trayectoria determinada, a saber, la distancia total recorrida en metros, su duración en segundos y su velocidad media en metros por segundo.

El usuario sólo tiene que proporcionar las trayectorias para las que desea obtener estas métricas, sin campos adicionales ajustables.

3.2.1.1. Entrada

Como se ha indicado en la descripción anterior, sólo hay un requisito para el funcionamiento de este proceso:

1) Las coordenadas de la trayectoria a remuestrear como pares de coordenadas (x, y).
   a) Cada uno de estos puntos debe tener asociada una marca de tiempo que indique la fecha y hora de ocurrencia.

A modo de consideración adicional, es posible proporcionar más de una trayectoria como entrada para este punto final. Para ello, la información mencionada debe proporcionarse dentro de una estructura de array con tantas posiciones como trayectorias a re-muestrear.

Cada trayectoria proporcionada debe cumplir con el formato y las condiciones mencionadas anteriormente y ser identificada por un paquete-ID proporcionado como un campo adicional.

El siguiente ejemplo de cuerpo JSON para la petición a la API se refiere al caso en el que se establecen dos trayectorias como entrada para el punto final de la interpolación.

Ilustración 10. Ejemplo de uso de la funcionalidad de estadísticas del módulo de análisis de trayectorias.

Para el caso de una sola trayectoria, la entrada JSON debe contener una matriz de "trayectorias" con una sola posición (para esa trayectoria).

3.2.1.2. Salida

Este endpoint devuelve un objeto JSON que asigna las métricas de distancia, duración y velocidad a cada uno de los paquetes-ID que identifican la trayectoria de entrada.

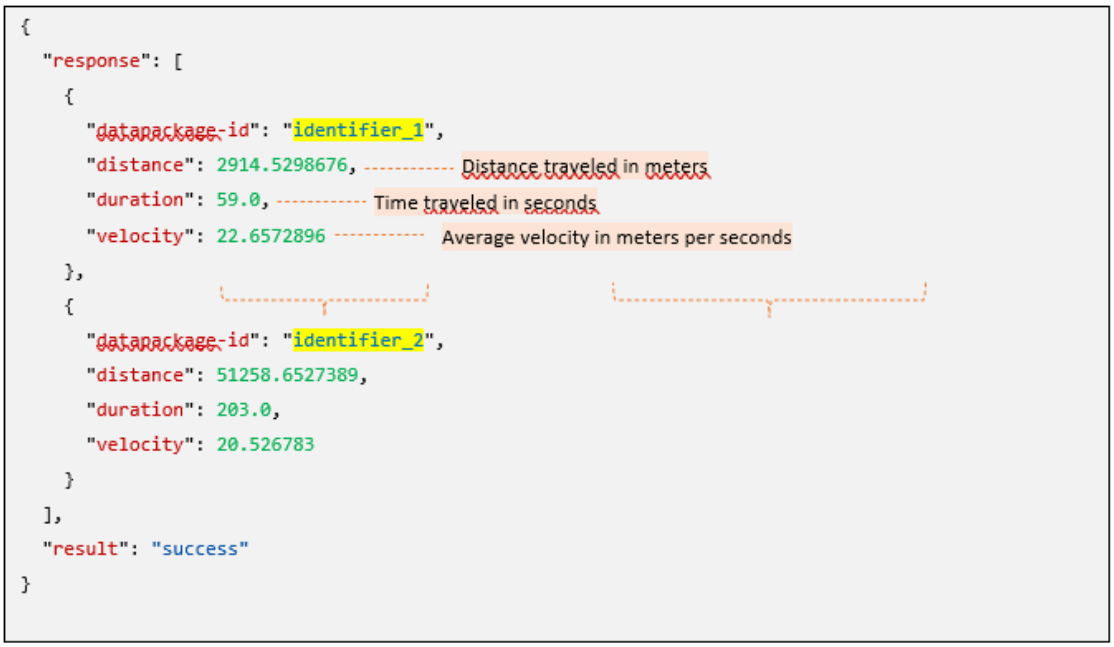

Ilustración 11. Ejemplo de la salida generada tras el uso de la funcionalidad de estadísticas del módulo de análisis de trayectorias.

3.2.2. Clustering de trayectorias

La función de clustering permite agrupar trayectorias similares. Las trayectorias vecinas se etiquetan como pertenecientes al mismo clúster basándose en una métrica de distancia por pares entre coordenadas.

Dado que el ajuste de los parámetros intrínsecos al algoritmo se ajusta automáticamente, el usuario sólo tiene que proporcionar las trayectorias que se van a agrupar, sin campos adicionales ajustables.

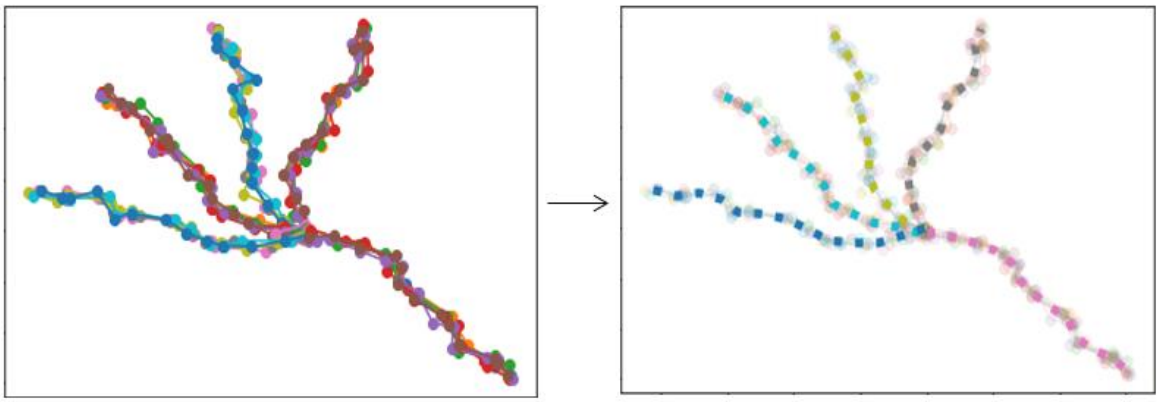

Ilustración 12. Ejemplo de varias trayectorias agrupadas en 5 clusters. En la imagen de la derecha, cada cuadrado representa la referencia del clúster.

3.2.2.1. Entrada

Como se ha indicado en la descripción anterior, sólo hay un requisito para el funcionamiento de este proceso:

1) Las coordenadas de las trayectorias a agrupar como pares de coordenadas (x, y).
   a) Cada uno de estos puntos debe tener asociada una marca de tiempo que indique la fecha y hora de ocurrencia. Nótese que la provisión de múltiples trayectorias debe estructurarse en un array con tantas posiciones como trayectorias a agrupar.

Cada trayectoria proporcionada debe cumplir con el formato y las condiciones mencionadas anteriormente y estar identificada por un paquete-ID proporcionado como campo adicional.

A continuación, se muestra un ejemplo de cuerpo JSON a proveer para la entrada y petición a este endpoint de la API.

Ilustración 13. Ejemplo de uso de la funcionalidad de clustering del módulo de análisis de trayectorias.

3.2.2.2. Salida

Este endpoint devuelve un objeto JSON que asigna una etiqueta de grupo a cada uno de los ID de paquete que identifican la trayectoria de entrada. Las trayectorias que pertenezcan al mismo grupo mostrarán la misma etiqueta de grupo.

```
{
  "ClusterLabels": [
    {
      "datapackage-id": "identifier_1",
      "label": 0
    },
    {
      "datapackage-id": "identifier_2",
      "label": 1
    },
    {
      "datapackage-id": "identifier_3",
      "label": 1
    }
  ],
  "result": "success"
}
```


Group-label assigned to the trajectory

Identifier

Ilustración 14. Ejemplo de la salida generada tras el uso de la funcionalidad de clustering del módulo de análisis de trayectorias.

3.2.3. Interpolación lineal de coordenadas

La funcionalidad de interpolación proporciona un medio de re-muestreo de una trayectoria determinada. El tipo de interpolación que se realiza es lineal.

Proporcionando una resolución temporal deseada (en segundos), el usuario puede establecer la frecuencia de aparición de cada punto en la nueva trayectoria remuestreada.

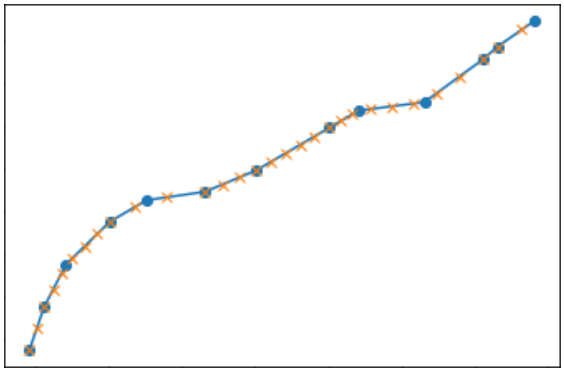

Ilustración 15. Ejemplo de una trayectoria remuestreada. Los círculos azules sólidos representan las coordenadas originales, mientras que las cruces naranjas son los puntos de la trayectoria remuestreada (para una resolución temporal determinada).

3.2.3.1. Entrada

Los parámetros necesarios para el funcionamiento de este proceso son:

1) Las coordenadas de la trayectoria a remuestrear como pares de coordenadas (x, y).
   a) Cada uno de estos puntos debe tener asociada una marca de tiempo que indique la fecha y hora de ocurrencia.
2) La resolución temporal (en segundos) a la que se pretende remuestrear la trayectoria como valor numérico. Este valor debe cumplir las siguientes condiciones:
   a) Ser un número entero.
   b) Ser mayor que 0.
   c) Ser menor que la diferencia de tiempo entre las marcas de tiempo de las coordenadas iniciales y las últimas.

Se puede proporcionar más de una trayectoria como entrada para este punto final. Para ello, la información mencionada debe proporcionarse dentro de una estructura de array con tantas posiciones como trayectorias a remuestrear.

Cada trayectoria proporcionada debe cumplir con el formato y las condiciones anteriormente mencionadas, incluyendo un valor de resolución temporal específico para cada una de ellas. Además, cada trayectoria debe estar identificada por un "package-ID" proporcionado como campo adicional.

El siguiente ejemplo se refiere al caso en el que se pasan dos trayectorias como entrada para el punto final de interpolación.

Ilustración 16. Ejemplo de uso de la funcionalidad de interpolación del módulo de análisis de trayectorias.

3.2.3.2. Salida

Este endpoint devuelve un objeto json que contiene las coordenadas de la trayectoria re-muestreada como pares de coordenadas (x, y). Cada uno de los nuevos puntos va acompañado de una marca de tiempo que contiene su hora de aparición remuestreada, con intervalos de tiempo entre cada punto de la trayectoria que obedecen a la resolución temporal especificada en la entrada.

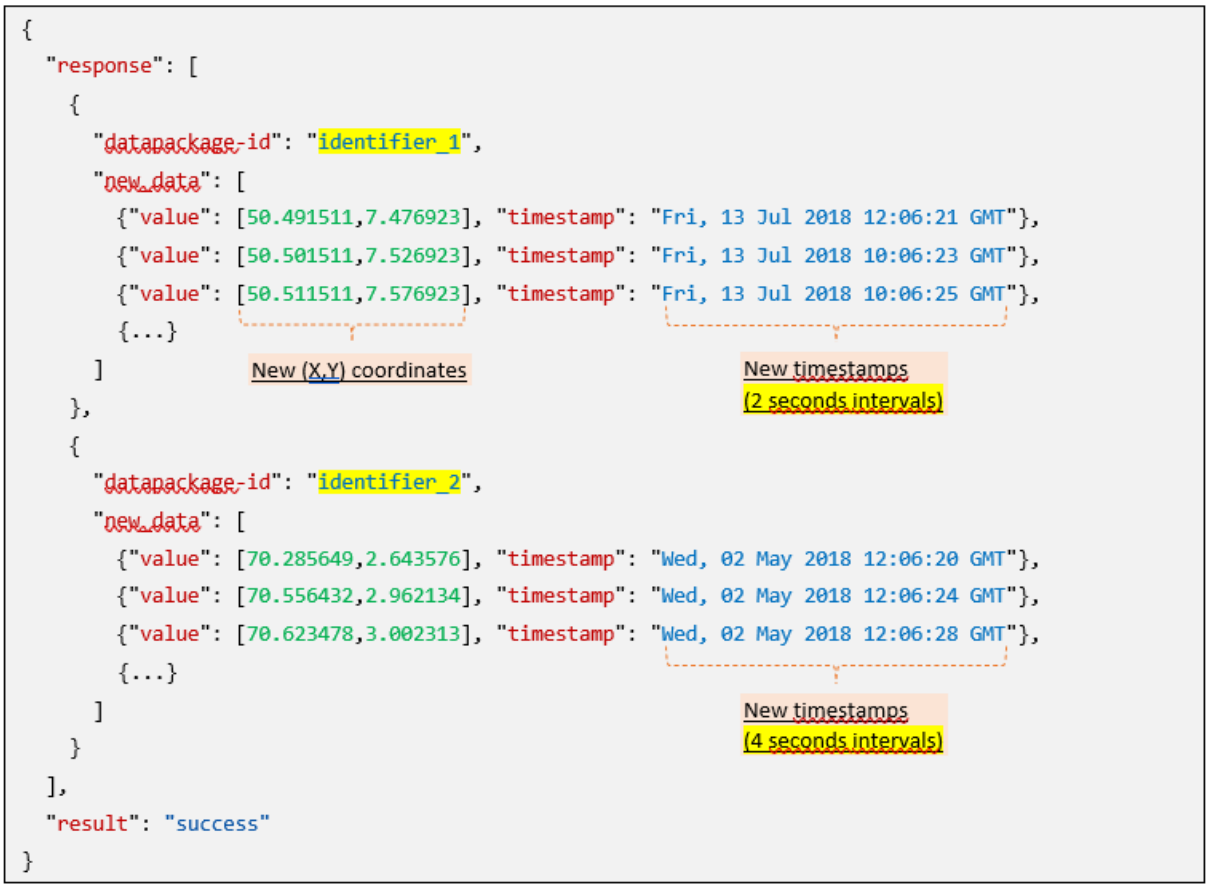

Ilustración 17. Ejemplo de la salida generada tras el uso de la funcionalidad de interpolación del módulo de análisis de trayectorias.

3.2.4. Anomaly detection

La funcionalidad de detección de anomalías permite identificar puntos anómalos en una trayectoria determinada. Tras asociar la trayectoria de entrada a un cluster a partir de la información almacenada en el sistema, compara la distancia por pares entre cada una de sus coordenadas y la del centroide del grupo. Si esta distancia supera un determinado umbral, las coordenadas asociadas se marcan como periféricas.

El usuario tiene la opción de cambiar el tipo de umbral a aplicar en este proceso entre un protocolo de desviación estándar y uno de rango intercuartil.

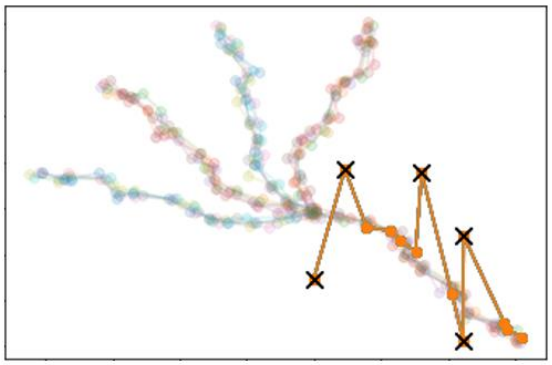

Ilustración 18. Ejemplo de anomalías detectadas en una trayectoria (marcadas con cruces negras).

3.2.4.1. Entrada

Los parámetros necesarios para el funcionamiento de este proceso son:

1) Las coordenadas de la trayectoria a evaluar como pares de coordenadas (x, y).
   a) Cada uno de estos puntos debe tener asociada una marca de tiempo que indique la fecha y hora de ocurrencia.
2) Opcionalmente, el método de umbral a aplicar para la identificación de los valores atípicos. Si el usuario no proporciona este parámetro, se aplica por defecto el protocolo de desviación estándar. Así, los únicos valores disponibles para este campo son
   a) 'std' para la desviación estándar.
   b) 'iqr' para el rango intercuartil.

Se puede proporcionar más de una trayectoria como entrada para este punto final. Para ello, dicha información debe proporcionarse dentro de una estructura de array con tantas posiciones como trayectorias se quieran evaluar.

Cada trayectoria proporcionada debe cumplir con el formato y las condiciones anteriormente mencionadas, incluyendo opcionalmente un campo de metodología de umbral específico para cada una de ellas. Además, cada trayectoria debe estar identificada por un package-ID proporcionado como campo adicional.

El siguiente ejemplo de cuerpo JSON se refiere al caso en el que se pasan dos trayectorias como entrada para el punto final de interpolación.

3.2.4.2. Salida

Este endpoint proporciona un objeto JSON que contiene cada par de coordenadas (X,Y) de la trayectoria de entrada asociada a un campo "índice de anormalidad", que contiene un valor numérico que indica cuántas veces mayor que el umbral es la distancia del punto periférico a su análogo del centro de la agrupación (este valor es 0 para los puntos regulares).


```
{
  "response": [
    {
      "datapackage-id": "identifier_1",
      "analyzed-data": [
        {
         "abnormality-rate": 0, --------- Abnormality rate for this coordinate
         "value": [50.491511,7.476923], "timestamp": "Fri, 13 Jul 2018 12:06:21 GMT"},
        {
         "abnormality-rate": 2.29348236785,
         "value": [53.501511,7.526923], "timestamp": "Fri, 13 Jul 2018 10:06:23 GMT"},
        {
         "abnormality-rate": 0,
         "value": [50.511511,7.576923], "timestamp": "Fri, 13 Jul 2018 10:06:25 GMT"},
        {...}
      ]
    },
    {
      "datapackage-id": "identifier_2",
      "analyzed-data": [
        {
         "abnormality-rate": 0,
         "value": [70.285649,2.643576], "timestamp": "Wed, 02 May 2018 12:06:20 GMT"},
        {
         "abnormality-rate": 0,
         "value": [70.556432,2.962134], "timestamp": "Wed, 02 May 2018 12:06:24 GMT"},
        {
         "abnormality-rate": 3.5693745270,
         "value": [70.623478,4.002313], "timestamp": "Wed, 02 May 2018 12:06:28 GMT"},
        {...}
      ]
    }
  ],
  "result": "success"
}
```


Ilustración 19. Ejemplo de la salida generada tras el uso de la funcionalidad de detección de anomalías del módulo de análisis de trayectorias.

3.2.5. Detalles de implementación

El módulo de análisis de trayectorias fue desarrollado como una clase de Python, para la versión 3.6.6 de Python. Una API de Python Flask, que encapsula la clase y expone sus funciones principales.

La clase de análisis de datos hace uso de las siguientes librerías:

- Scikit-learn, version 0.20.3.
- Numpy, version 1.16.3.
- Scipy, version 1.2.1.
- Click, version 7.0.
- Flask, version 1.0.2.
- Flask-restplus 0.13.0
- Dipy 1.1.1
- Nibabel 3.0.1
- Geopy 1.21.0
- Itsdangerous, version 1.1.0.
- Jinja2, version 2.10.1.
- MarkupSafe, version 1.1.1.
- Python-dateutil, version 2.8.0.
- Six, version 1.12.0.
- Werkzeug, version 0.15.2.

Para proveer detalle sobre la implementación de la API, su estructura implica dos dos partes principales:

- La API de Flask en un archivo app.py, que maneja la comunicación entre el sistema de Cross CPP y el Analizador de Trayectorias. Funciona como una capa donde se reciben los datos entrantes, se procesan para cumplir con los requisitos del módulo, y se opera con la funcionalidad correspondiente.

- Analizador de trayectorias: Es la pieza de software que incluye los algoritmos subyacentes de cada funcionalidad. Recibe la información de la API de Flask, ejecuta los algoritmos y emite una respuesta.

3.3. Módulo de análisis de redes complejas

Este módulo tiene como objetivo ofrecer información relevante con respecto a las asociaciones presentes entre el comportamiento de las señales tomadas por un conjunto de dispositivos IoT. Estos se representan como nodos en una red, conectados por aristas (ponderadas o no) que indican su similitud. A continuación, se puede ver un detalle de los endpoints diseñados para la API asociada a este módulo.

Endpoint	URL	request type	parameters	result
Create Network	crosscpp/networks /create/	POST	A network ID, measurement-channel-id, an AEON Subscription Url and a set of nodes.	None. Operation performed in the server: A new network will be created, initializing it with the data included in the petition. Also, subscription to the AEON Subscription Url provided in the request will take place.
Information	/crosscpp/networks /info/<networkId>	GET	None.	A series of metrics summarizing the status of the network. This include network diameter, network efficiency, link density and node eccentricity.

Ilustración 20. Sumario de los endpoints asociados a la la API del módulo de análisis de redes complejas.

3.3.1. Creación de una red

A través de este endpoint de la API se podrá crear una red a través de la especificación de las señales implicadas (nodos) y las conexiones entre cada una de ellas, si procede (ponderadas, o no).

3.3.1.1. Entrada

Se plantea, a continuación, un ejemplo del cuerpo JSON requerido para el funcionamiento de este proceso:

```
{
"networkId": "test",
"nodes": [{
        "datapackage-id": "identifier-1"
        "data": [
            {"value":7.4434,"timestamp":"2018-07-13T10:06:21.622970Z"},
            {"value":7.76923,"timestamp":"2018-07-13T10:06:25.622970Z"},
            {"value":7.923,"timestamp":"2018-07-13T10:06:30.622970Z"},
[…]
         ]}
],
"links": [
      [0, 5],
      [1, 2],
      [3, 7, 3]
]}
```


Ilustración 21. Ejemplo de uso de la funcionalidad de creación de una red.

- **networkId:** identificador textual de la red.
- **nodes:** lista de señales de los dispositivos con los que se va a crear la red, del mismo modo que se pudo apreciar en el módulo de análisis de trayectorias.
- **links:** lista de tuplas indicando en las dos primeras posiciones el identificador numérico de los nodos a conectar (por orden de aparición en la lista "nodes"), y en la tercara, de forma opcional, el peso asignado a dicho enlace.

3.3.1.2. Salida

Este endpoint no retorna ningún tipo de información, salvo un indicador del éxito de la operación en caso de lograrse la creación de la red.

3.3.2. Obtención de información de una red

Como se adelantó en las secciones introductorias, se proporciona información de sumario de las redes generadas, incluyendo:

- Diámetro de la red
- Eficiencia de la red
- Excentricidad de cada nodo de la red

3.3.2.1. Entrada

Este endpoint no requiere que se le provea más información que el identificador de la red del que se quiere consultar la información (campo “networkId” establecido en el proceso de creación de la red).

3.3.2.2. Salida

Este endpoint retorna un diccionario con información de cada una de las métricas de análisis de la red consideradas. A continuación, se muestra un ejemplo:

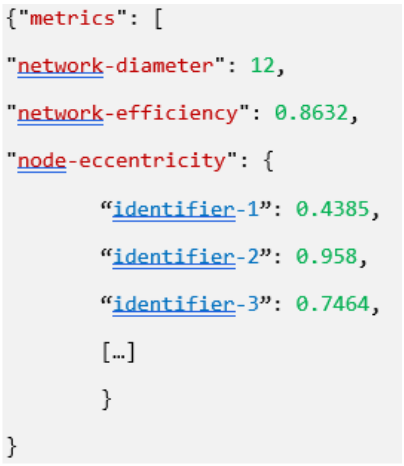

Ilustración 22. Ejemplo de la salida del uso de la funcionalidad de obtención de métricas de una red.

Cada métrica está identificada con claridad, seguida de su valor numérico asociado. En el caso de la excentricidad de los nodos, se devuelve un diccionario embebido en el que, para cada identificador de dispositivo establecido en el proceso de creación de la red (“datapackage-id”), se ofrece el valor numérico asociado.

3.3.3. Detalles de implementación

El módulo de análisis de redes se ha desarrollado en la versión 3.6.6 de Python. Como en el caso del módulo de análisis de trayectorias, se compone de dos partes: una clase que realiza todos los cálculos propiamente dichos, y una API que encapsula la anterior clase de Python y expone sus funciones principales.

La clase de análisis de datos hace uso de las siguientes bibliotecas estándar

- scikit-learn, versión 0.20.0.
- scipy, versión 1.1.0.

- numpy, versión 1.15.2.
- pandas, versión 0.23.4.

Para optimizar el cálculo de las métricas de red estándar, todos los cálculos básicos se apoyan en la biblioteca “networkx”, versión 2.2. Además, se necesitan varias bibliotecas adicionales para proporcionar la funcionalidad de la API, entre ellas

- cycler, versión 0.10.0.
- decorator, versión 4.3.0.
- kiwisolver, versión 1.0.1.
- lxml, versión 4.2.5.
- matplotlib, versión 3.0.0.
- pydot, versión 1.2.4.
- pyparsing, versión 2.2.2.
- python-dateutil, versión 2.7.3.
- pytz, versión 2018.5.
- six, versión 1.11.0.

Como se ha mencionado, la estructura de la API se puede dividir en dos partes:

- La API de **Flask** en un archivo app.py, que se encarga de la comunicación entre el sistema Cross CPP y el analizador de trayectorias. Funciona como una capa en la que se reciben los datos entrantes, se procesan para cumplir los requisitos del Analizador de Trayectorias y se envían al Analizador de Trayectorias. Por último, en esta capa se recibe la respuesta de los algoritmos, también se procesa y se envía al Cross CPP.
- **NetworkDX**: Es la pieza de software que incluye los algoritmos subyacentes de cada funcionalidad. Recibe la información de la API de Flask, ejecuta los algoritmos y emite una respuesta.

4. Ejemplos de uso de la herramienta

A fin de ofrecer detalles en cuanto la forma en que se pueden utilizar los servicios proporcionados por el "Toolbox" de análisis de datos, se ha desarrollado un conjunto de casos de uso. Proceden de posibles escenarios proporcionados por algunos de los socios del proyecto Cross-CPP en el que se enmarca este Trabajo de fin de Máster. De acuerdo a este planteamiento, se presenta la estructura que seguirán las siguientes secciones del documento:

- Introducción de los perfiles y necesidades de los socios colaboradores.
- Detalle de cada uno de los casos de uso identificados, de acuerdo a los requerimientos de los socios:
    - Garantía de la calidad de los datos.
    - Activación de alertas y avisos informativos.
    - Fiabilidad de las mediciones del CPP.
    - Agrupación de datos.

4.1. Requerimientos identificados

Por un lado, Meteologix [20] es una empresa de TI e I+D encargada de producir contenidos de valor asociados al comportamiento meteorológico en todo el mundo. En el contexto de este proyecto, colabora con Volkswagen [21], una empresa automovilística que pretende enriquecer el conjunto de soluciones informáticas que ofrece a sus clientes de vehículos.

El objetivo conjunto de estos socios consiste en informar con exactitud de las condiciones ambientales y proporcionar información precisa y valiosa con respecto a las condiciones meteorológicas que rodean a un vehículo concreto y su posible impacto en la seguridad de su conducción. Para ello, es conveniente poder hacer un seguimiento de dichas mediciones en cualquier lugar desde el que cualquier cliente lo solicite.

La cooperación de Volkswagen con las funcionalidades de Meteologix (en concreto, los servicios de información y previsión meteorológica) debe garantizar el suministro de los datos adecuados correspondientes a cada escenario específico, lo que redunda en los siguientes requisitos:

1. Los datos recogidos para su uso posterior deben ser
    a. **Fiables**: libres de anomalías y de cualquier otra condición que pueda perjudicar los modelos y las estimaciones.
    b. **Precisos**: ajustados a cualquier tipo de variabilidad que pueda haber en una señal de confianza (por ejemplo, cambios severos de temperatura según la hora del día).

    Estas características permitirían mejorar los modelos y las estimaciones en lugares con condiciones climáticas especiales o variables, como montañas, valles, paisajes tropicales o desérticos, etc.

    Como se discutirá en las siguientes secciones, también implican la posibilidad de métodos para filtrar los datos de acuerdo con las características de interés, ayudando a cualquier informe posterior o proceso de predicción.

2. Generar avisos o notificaciones según escenarios que puedan poner en peligro la seguridad de los vehículos:
    a. Condiciones meteorológicas extremas.
    b. Condiciones peligrosas de la carretera.

    Las funcionalidades asociadas implican el reconocimiento de patrones entre los datos de entidades IoT conocidas (es decir, edificios y/o otros coches) para ayudar a la identificación del entorno del cliente y desencadenar las acciones a realizar con la suficiente confianza en su corrección.

3. Evaluar la fiabilidad de las mediciones de los coches atendiendo a las condiciones ambientales de su entorno.
4. Disponer de capacidades de clustering para identificar grupos de trayectorias similares, permitiendo preparar cualquier proceso de análisis posterior atendiendo a las particularidades de cada cluster.

4.2. Calidad de los datos

Una vez realizada la recogida de datos, la verificación de la fiabilidad de un sensor es crítica, ya que cualquier comportamiento anómalo compromete directamente la posterior toma de decisiones, e incluso la seguridad del cliente. Bajo esta premisa, se plantea un escenario en el que un sensor está produciendo un comportamiento anómalo en las señales enviadas al CPP Marketplace.

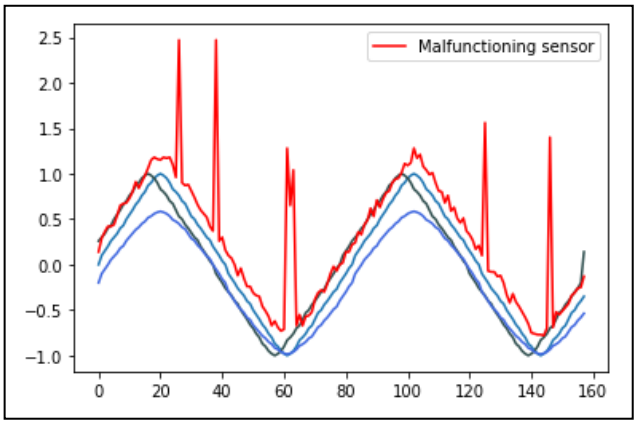

Ilustración 23. Ejemplo de diferentes señales recibidas de un sensor. Las señales coloreadas en diferentes tonos de azul muestran un comportamiento regular, mientras que la señal en rojo es claramente anómalaEjemplo de diferentes señales recibidas de un sensor.

.

Para detectar estos comportamientos indeseables en los datos recibidos del mercado, la funcionalidad de "Drift Detection" del “Toolbox” es de gran interés.

Al cuantificar la agresividad con la que fluctúa la señal de un sensor, proporciona una medida muy interpretable de la cantidad de ruido que contiene. A continuación, se muestra

un ejemplo de la salida de la métrica de Entropía de Muestra cuando se alimenta con las señales presentadas anteriormente:

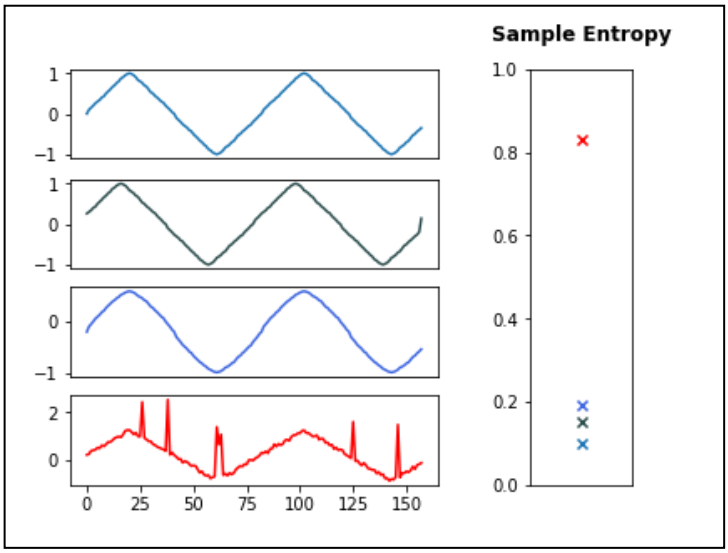

Ilustración 24. Desglose del resultado de la métrica de la Entropía de la Muestra para cada señal. La señal anómala lleva asociado un valor mucho más alto que el de las señales de forma regular.

Una salida como la presentada en la imagen anterior permite separar claramente las señales fiables de las anómalas, lo que ayuda a identificar los dispositivos que funcionan mal.

Además, este proceso puede acoplarse a la función de agrupación de trayectorias del "Toolbox", ya que la identificación de grupos de trayectorias puede ayudar a desarrollar modelos de referencia del comportamiento global de los sensores de los coches en lugares específicos. La aplicación de este procedimiento en el flujo de trabajo propuesto redundaría en un enfoque más detallado de la detección de anomalías.

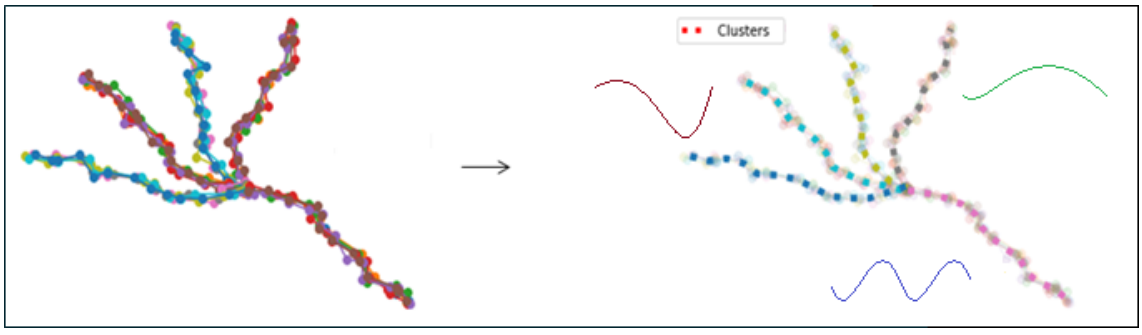

Ilustración 25. Ejemplo de la salida del algoritmo de agrupación de trayectorias de la caja de herramientas. Las diferentes localizaciones identificadas por cada cluster pueden estar asociadas a diferentes comportamientos de los sensores.

Además, de acuerdo con los diferentes valores que se pueden asignar a cada una de las señales alimentadas, el usuario tiene la decisión final de filtrar aquellas con un mayor grado de deriva detectado (mayor Entropía de Muestra en el ejemplo anterior).

Los coches que trazan trayectorias anómalas pueden producir mediciones atípicas. Este caso también incluye la posibilidad de un mal funcionamiento del dispositivo GPS del coche.

Ilustración 26. Ejemplo de detección de posiciones anómalas comunicadas por un dispositivo GPS. Los puntos marcados con una "X" representan las anomalías identificadas por el algoritmo de detección de la Toolbox.

La idea que subyace a la identificación de estos casos es prácticamente equivalente a la del escenario anterior. Para este caso, sin embargo, el “Toolbox” cuenta con un conjunto de procedimientos específicamente diseñados para el tratamiento de datos GPS.

Proveyendo un conjunto de trayectorias a la funcionalidad de detección de anomalías, el usuario obtendrá información sobre las coordenadas de una trayectoria que se alejan de la normalidad. Con esta información, el usuario puede decidir si incluir o no un coche determinado en cualquier análisis posterior de las mediciones de sus sensores.

4.3. Generación de alertas

Los escenarios analizados en esta sección se refieren a la activación y generación de avisos informativos a los clientes. La información se presenta en dos categorías, atendiendo a si la información que se evalúa está asociada a la meteorología o a la carretera por la que circula un vehículo.

4.3.1. API para servicios meteorológicos

La capacidad de notificar al usuario en caso de condiciones meteorológicas anómalas o potencialmente peligrosas en la ruta planificada requiere el reconocimiento preciso de los patrones climáticos de una zona específica.

Con este enfoque en mente, dada una señal específica de interés (por ejemplo, la temperatura, la intensidad de la lluvia...) para la que se han recogido datos, la funcionalidad de Redes de la caja de herramientas de análisis es una opción adecuada para distinguir los patrones anómalos que se producen en un área específica.

Como ejemplo, proporcionando la señal de temperatura del coche del usuario junto con la de otras recogidas en el CPP Marketplace (de la misma ubicación), este procedimiento generaría una red en la que las señales altamente relacionadas se representarían como un vecindario conectado. En este caso, cada grupo representaría los diferentes niveles a los que puede desplazarse la temperatura.

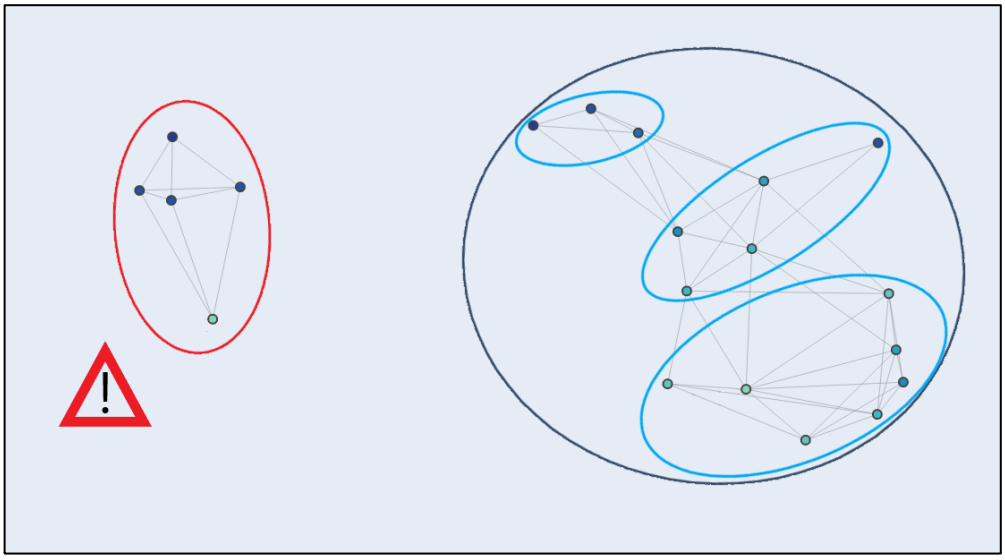

Ilustración 27. Ejemplo de una red generada para diferentes señales. Las circunferencias coloreadas representan los diferentes grupos que pueden identificarse en ella. El grupo rojo representa un conjunto de señales anómalas.

Como se representa en la imagen anterior, los niveles de temperatura anómalos se representarían como un barrio poblado más alejado de los demás (y sin conexiones con ellos). Este resultado sería significativamente valioso para determinar si la ruta planificada por un cliente comprende zonas con condiciones meteorológicas que necesitan que se emita un

aviso, ya que la señal del coche estaría relacionada con un "grupo anómalo" en la red resultante.

Además, cabe destacar la utilidad de la funcionalidad de las Redes para detectar valores atípicos en un conjunto de señales, lo que ayuda a garantizar la fiabilidad de los datos y a orientar cualquier proceso posterior de eliminación de señales no utilizables. En la salida proporcionada por la funcionalidad de Redes, se pueden identificar como puntos aislados en el espacio. Los valores atípicos se distinguen de las anomalías plausibles (como las comentadas en el ejemplo anterior) en que tienen pocos o ningún vecino, en lugar de formar parte de un grupo poblado y separado de los demás.

4.3.2. Diagnóstico de estado de la carretera

Un escenario crítico se define por la capacidad de detectar con precisión las condiciones peligrosas de la carretera y actuar en consecuencia.

Profundizando en este caso, se puede programar una solicitud de datos en el CPP Marketplace para recuperar información sobre un conjunto de señales de velocidad de los coches en una ubicación específica (la carretera evaluada).

A este respecto, la caja de herramientas de análisis proporciona una serie de capacidades de "detección de desviaciones", a saber, la Entropía de Muestra, la Entropía de Permutación y la Irreversibilidad. Estas permiten cuantificar la cantidad de deriva presente en una serie temporal (una señal de velocidad en este caso).

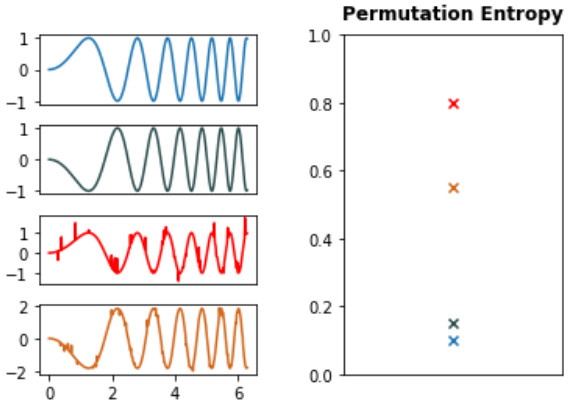

Ilustración 28. Desglose del resultado de la métrica de la Entropía de la Muestra para cada señal de velocidad. Las señales que presentan zonas muy oscilantes se asocian a un valor mucho mayor que las de forma regular.

A partir de una salida como la que se presenta en esta imagen, el usuario podría obtener un sentido cuantificado de la brusquedad con la que cambia la velocidad de un coche.

Además, en lo que respecta a la necesidad especial de precisión en la identificación de los escenarios de activación del ABS, este proceso podría ir acompañado de un paso de preprocesamiento relativo al filtrado de los datos de la colección evaluada. Es probable que los coches que muestren comportamientos anómalos en sus trayectorias tengan valores de señal incoherentes en sus coordenadas GPS periféricas, perjudicando así cualquier proceso de análisis en el que estén implicados.

Para hacer frente a este problema, la caja de herramientas de análisis ofrece una funcionalidad de detección de valores atípicos con la que se pueden identificar las coordenadas anómalas en un conjunto de trayectorias de coches.

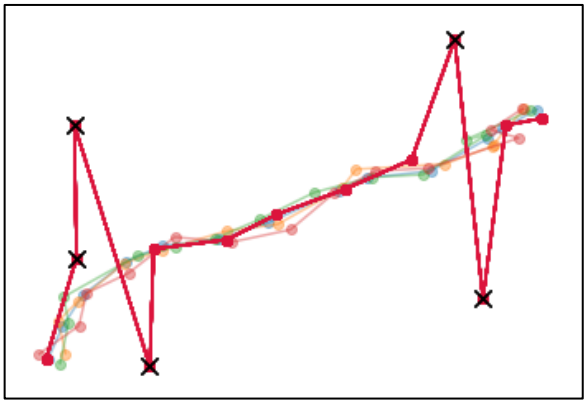

Ilustración 29. Ejemplo de detección de posiciones anómalas comunicadas por un dispositivo GPS. Los puntos marcados con una "X" representan las anomalías identificadas por el algoritmo de detección de la caja de herramientas.

Con esta salida, el usuario puede identificar y filtrar cualquier coordenada anómala en los datos utilizados. Además, en caso de que no haya un reemplazo claro para los valores atípicos detectados y se desee inspeccionar más a fondo la trayectoria correspondiente, la Caja de Herramientas de Análisis también proporciona una función de interpolación con la que se pueden volver a muestrear las coordenadas de una trayectoria determinada.

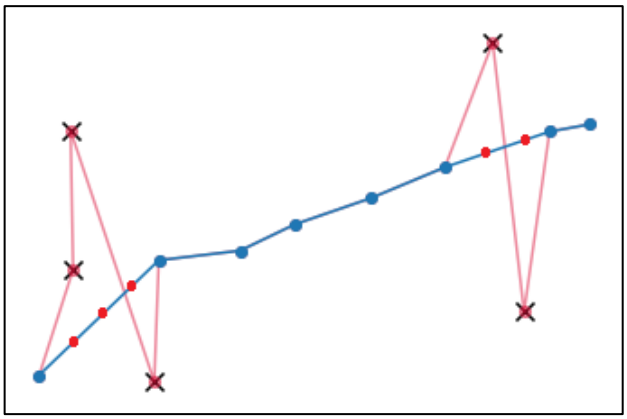

Ilustración 30. Trayectoria remuestreada basada en una trayectoria de la que se han eliminado las coordenadas anómalas.

Estas funcionalidades permiten preprocesar y manipular los datos de forma que se pierda la mínima información posible, haciendo que la entrada a cualquier proceso de análisis posterior sea lo más precisa y fiable posible.

Recordando el escenario del ABS que estamos discutiendo, si hubiera trayectorias anómalas asociadas a las señales de velocidad requeridas desde el CPP Marketplace, el usuario podría filtrar esas coordenadas y re-muestrearlas. Esto ayudaría significativamente a las estimaciones posteriores de lo que las mediciones de los sensores habrían sido en una ubicación no anómala.

4.4. Fiabilidad de las mediciones de un vehículo

Para garantizar la fiabilidad de las mediciones de un coche y de las indicaciones de información asociadas que se ofrecen al usuario, es crucial tener en cuenta cualquier efecto potencial del entorno del vehículo sobre los diferentes sensores que lleva.

Por ejemplo, supongamos que un cliente necesita un informe de temperatura para una ruta planificada, lo que requiere que el sistema estime la temperatura en la zona asociada. Sin embargo, se sabe que el color de un coche puede afectar en gran medida a la temperatura medida por sus sensores, ya que los tonos oscuros absorben el calor y los brillantes lo reflejan. Si no se tiene en cuenta este hecho, se pueden ofrecer al cliente mediciones poco precisas.

En este escenario, la funcionalidad de Redes de la Caja de Herramientas de Análisis es una opción adecuada para identificar claramente grupos de mediciones de temperatura en un área específica para las señales recogidas de una Solicitud de Datos del Mercado CPP.

Al proporcionarlas como entrada, este procedimiento generaría una red en la que las señales altamente relacionadas se representarían como una vecindad conectada. En este caso, cada grupo representaría los diferentes niveles de temperatura medidos por diferentes vehículos en la misma zona.

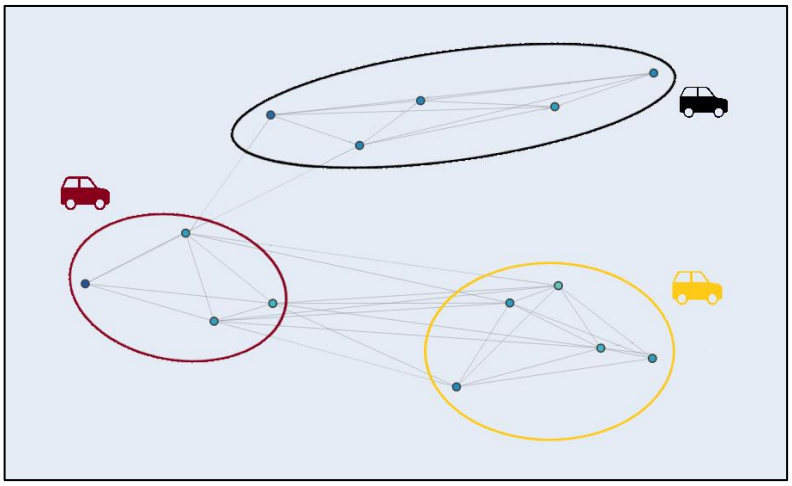

Ilustración 31. Ejemplo de una red generada para diferentes señales. Las circunferencias coloreadas representan los diferentes grupos que pueden identificarse en ella.

Conociendo de antemano el color de cada uno de los coches implicados (una información que también está disponible en el CPP Marketplace), una salida como la presentada en esta imagen revelaría patrones claros sobre las temperaturas medidas por cada tipo, permitiendo así una estimación más precisa de cómo sería el valor real (por ejemplo, calculando una media o cualquier otra estadística de resumen).

4.5. Clustering de datos

Para garantizar la precisión en las estimaciones posteriores que realice el usuario, es clave identificar cómo fluctúa el comportamiento de una determinada señal en relación con otra variable (por ejemplo, la hora, la ubicación...).

Para ejemplificar, supongamos que el cliente pretende estimar la temperatura en una zona en la que se sabe que fluctúa mucho a lo largo del día. Para ello, primero se han recogido los datos de ese lugar en diferentes momentos del día.

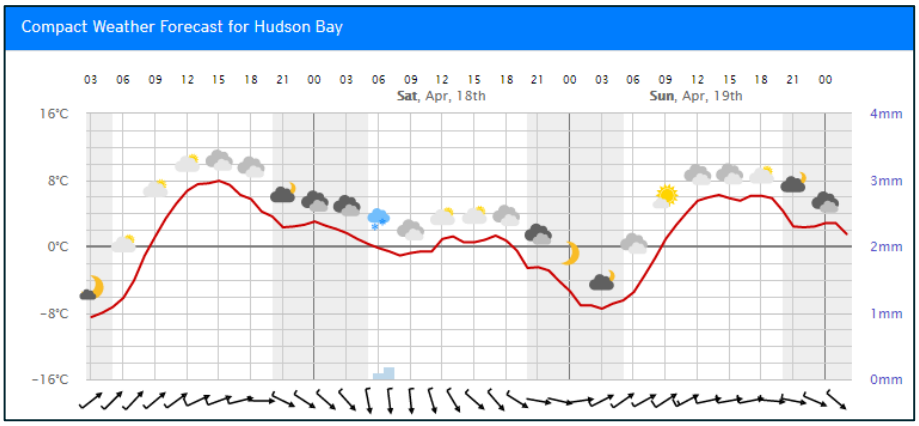

Ilustración 32. Ejemplo de un caso en el que el tiempo varía significativamente, tanto a lo largo del día como entre días. Imagen tomada de https://meteologix.com/es.

La funcionalidad de Redes de la Caja de Herramientas de Análisis es una opción adecuada para identificar claramente los patrones de temperatura en todas las señales recogidas.

Al proporcionar las señales mencionadas como entrada, este procedimiento generaría una red en la que las señales altamente relacionadas se representarían como una vecindad conectada. En este caso, cada grupo representaría los diferentes niveles a los que puede ascender la temperatura a lo largo del día.

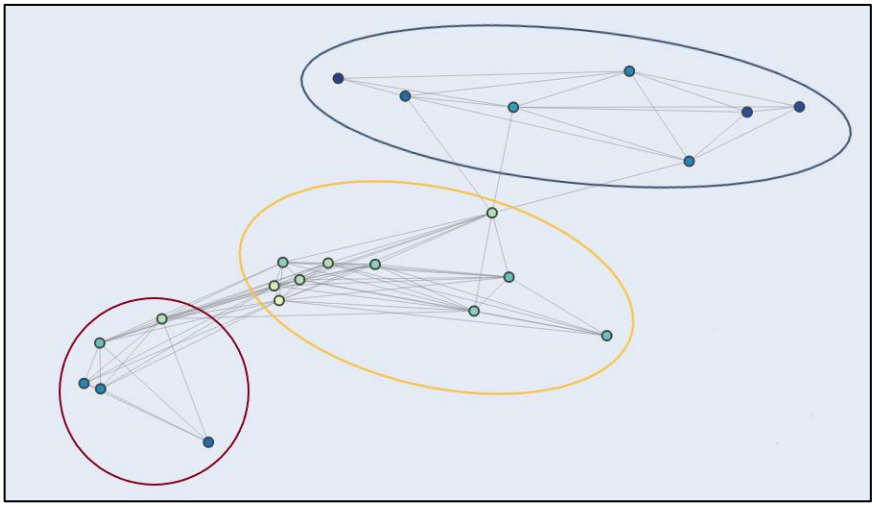

Ilustración 33. Ejemplo de un caso en el que el tiempo varía significativamente, tanto a lo largo del día como entre días. Imagen tomada de https://meteologix.com/es.

Esta información ayudaría a una clara identificación de los patrones para su adecuada separación con el fin de proporcionar estimaciones precisas para cada marco temporal.

Junto a las funcionalidades anteriormente presentadas, la Analytics Toolbox cuenta también con una operación de clustering con un conjunto de procedimientos específicamente diseñados para el tratamiento de datos GPS.

Este proceso puede ser de interés para identificar grupos de trayectorias, que a su vez pueden estar relacionados con diferentes comportamientos en las mediciones realizadas por los sensores de los coches asociados. Un resultado consistente en grupos de trayectorias claramente distinguibles permitiría una evaluación más precisa de las condiciones asociadas a un grupo concreto.

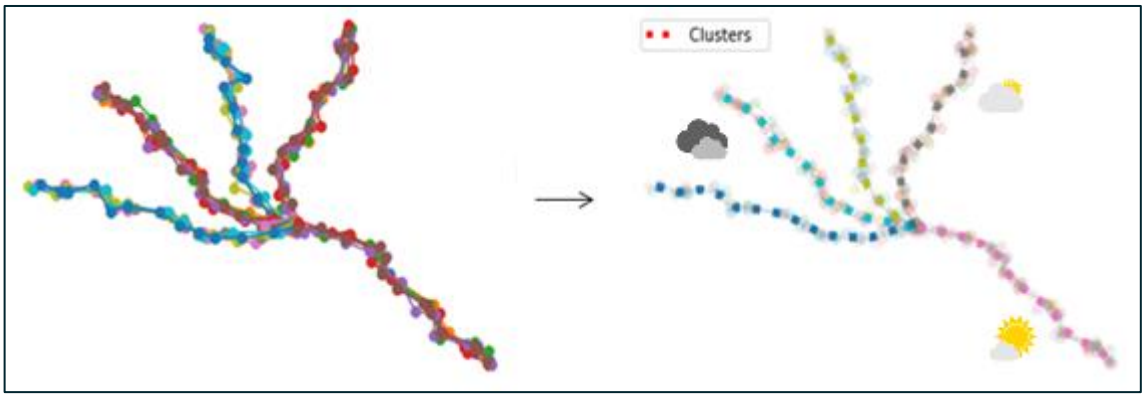

Ilustración 34. Ejemplo de la salida del algoritmo de agrupación de trayectorias de la caja de herramientas. Las diferentes localizaciones identificadas por cada clúster pueden estar asociadas a diferentes condiciones meteorológicas.

5. Conclusiones y trabajos futuros

El desarrollo de este proyecto de fin de máster ha concluido con el diseño, desarrollo e implementación exitosos de un "Toolbox" de análisis de datos generalista, aunque orientado al ámbito IoT por la circunscripción de este al proyecto europeo Cross CPP.

Con respecto al cumplimiento de los objetivos establecidos para este proyecto, puede establecerse que:

- El módulo de análisis de series temporales habilita funciones útiles para el análisis de este datos presentados en este formato. Abarca características asociadas al análisis descriptivo de una serie ("Drift Detection" y análisis de correlación), como al análisis predictivo (árboles de regresión, modelos ARIMA y modelos de redes neuronales).
- El módulo de minería de trayectorias ofrece funciones relacionadas con la estadística simple y la agrupación de datos de trayectorias, además de habilitar la detección de anomalías e interpolación lineal de coordenadas.
- El módulo de análisis de redes dispone de las funcionalidades necesarias para tratar la información estadística general de representaciones abstractas de redes.

Puede concluirse, por tanto, que se han alcanzado los objetivos fijados para el proyecto.

Merece mención especial la publicación en la revista MDPI para Ciencias Aplicadas (especial "Computing and Artificial Intelligence for Visual Data Analysis") del paper "Clustering Moving Object Trajectories: Integration in CROSS-CPP Analytic Toolbox" [22] a raíz del desarrollo de la funcionalidad de clustering de trayectorias.

Con respecto a las líneas de trabajo futuras, queda pendiente la exploración de vías de ampliación de las funcionalidades del "Toolbox" desarrollado, además de la evaluación de su carácter generalista aplicándolo a un conjunto de datos al margen del ámbito IoT.

6. Estimación y presupuesto

Para el desarrollo del trabajo, se plantea una metodología de desarrollo software en cascada. No obstante, el diseño e implementación de cada una de las funcionalidades lleva su propio ciclo de desarrollo completo. Si hubiera más de un desarrollador, eso permitiría paralelizar estas tareas. Como todo el trabajo ha sido desarrollado por el autor de este TFM, todas las tareas se han realizado de manera secuencial.

A continuación, se muestra la propuesta del desarrollo de trabajo con sus fases:

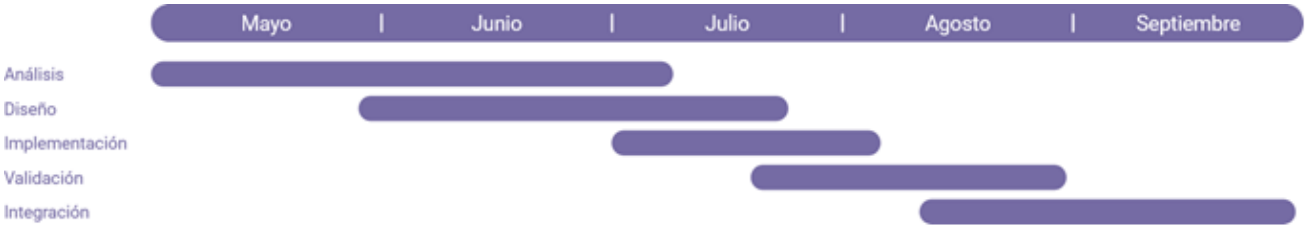

Ilustración 35. Estructura y organización de la distribución del esfuerzo en cada etapa del proyecto a lo largo del tiempo.

6.1. Medios empleados y esfuerzo dedicado

Los medios requeridos para el cumplimiento de los objetivos fijados han sido provistos por el laboratorio donde el alumno desarrolla su trabajo ("Medical Data Analytics Laboratory" [23], parte de la Universidad Politécnica de Madrid), e incluyen:

- Ordenador de sobremesa orientado al desarrollo técnico del software (fases de implementación y validación del proceso).
    - Procesador: Intel core i3-8100
    - Memoria RAM: 8 GB
    - Disco duro SSD: 512 GB
    - Tarjeta gráfica: Intel UHD Graphics 630
- Servidor dedicado para el hospedaje de los servicios para su puesta en fase de producción(fase de integración).
    - Procesador: Intel core i7-9700K
    - Memoria RAM: 16 GB
    - Disco duro SSD: 512 GB
    - Tarjeta gráfica: Nvidia GeForce RTX 2070 Super (8 GB)

Se estima una media de dedicación diaria para el desarrollo total del proyecto de 8 horas diarias (jornada laboral regular que se ha dedicado dentro del laboratorio MEDAL para el desarrollo del proyecto). Ello, ligado a los 5 meses establecidos para el desarrollo del presente trabajo de fin de máster ha dado lugar a una estimación presupuestal de 10000 €, a razón de 2000 € mensuales.

No se han considerado fungibles ni material adicional en el presupuesto, ya que es material provisto por el laboratorio en el que el alumno ha desarrollado el proyecto como parte de su carrera laboral.

7. Referencias

[1] A. D. Mauro, M. Greco y M. Grimaldi, «A formal definition of Big Data based on its essential features,» 2016, pp. 122-135.

[2] «AutoMat - Automotive Big Data Marketplace for Innovate Cross-sectorial Vehicle Data Services,» [En línea]. Available: https://www.automat-project-euu/.

[3] «CORDIS. About Cross-CPP,» 2018. [En línea]. Available: https://cordis.europa.eu/project/rcn/214254/factsheet/en.

[4] «Data stream mining,» [En línea]. Available: https://en.wikipedia.org/wiki/Data_stream_mining.

[5] J. Gama, «Knowledge Discovery from Data Streams,» p. 187.

[6] P. Holme y J. Saramäki, «Temporal networks,» *Physics Reports,* vol. 519.3, pp. 97-125, 2012.

[7] J. Brownlee, «How to Create an ARIMA Model for Time Series Forecasiting in Python,» *Time Series,* 2017.

[8] «Pearson correlation coefficient,» [En línea]. Available: https://en.wikipedia.org/wiki/Pearson_correlation_ coefficient#Definition.

[9] «Spearman's Correlation Coefficient,» [En línea]. Available: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient.

[10] M. Zanin, L. Zunino, O. Rosso y D. Papo, «Permutation Entropy and Its Main Biomedical and Econophysics Applications: A Review,» *Entropy,* vol. 14, 2012.

[11] D. Zheng, «A review of moving object trajectory clustering algorithms,» *Artificial Intelligence Review,* vol. 47.1, pp. 123-144.

[12] Z. Kalal, K. Mikolajczyk y J. Matas, «Forward-Backward Error: Automatic Detection of Tracking Failures,» de *20th International Conference on Pattern Recognition*, 2012.

[13] D. P. Mico Nanni, «Time-focused clustering of trajectories of moving objects,» *Journal of Intelligent Information Systems,* vol. 27.3, pp. 267-289, 2006.

[14] «Haversine formula,» [En línea]. Available: https://en.wikipedia.org/wiki/Haversine_formula.

[15] E. Garyfallidis, M. Brett, M. M. Correia, G. B. Williams y I. Nimmo-Smith, «QuickBundles, a method for tractographhy simplification,» *Frontiers in Neuroscience,* vol. 11, 2012.

[16] S. H. Strogatz, «Exploring Complex Networks,» *Nature,* pp. 268-276, 2001.

[17] L. d. F.Costa, O. N. O. Jr., G. Travieso, F. A. Rodrigues, P. R. V. Boas, L. Antiqueira, M. P. Viana y L. E. C. d. Rocha, «Analyzing and modelling Real-World Phenomena with Complex Networks: A Survey of Applications,» *Advances in Physics,* pp. 329-412, 2011.

[18] M.Zanin, D.Papo, P.A.Sousa, E.Menasalvas, A.Nicchi, E.Kubik y S.Boccalettif, «Combining complex networks and data mining: Why and how,» *Physics Reports,* vol. 635, pp. 1-44, 2016.

[19] E. W. Dijkstra, «A note on two problems in connection with graphs,» *Numerische Mathematik,* vol. 1.1, pp. 2269-271, 1959.

[20] «Meteologix,» [En línea]. Available: https://meteologix.com/es.

[21] «Volkswagen,» [En línea]. Available: https://www.volkswagen.es/.

[22] A. Blazquez-Herranz, J.-I. Caballero-Garzon, A. Zilverberg, C. Wolff, A. Rodríguez-Gonzalez y E. Menasalvas, «Clustering Moving Object Trajectories: Integration in CROSS-CPP Analytic Toolbox,» *MDPI, Applied Sciences,* vol. 11, 2021.

[23] «Medical Data Analytics Laboratory,» [En línea]. Available: https://medal.ctb.upm.es/.