



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Análisis de algoritmos de clusterización para la detección de
rutas marítimas habituales a partir de datos AIS*

Grado en Ingeniería Mecánica

ALUMNO: Aurelio Ramón Fernández Espinosa

DIRECTORES: Miguel Rodelgo Lacruz

CURSO ACADÉMICO: 2020-2021

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Análisis de algoritmos de clusterización para la detección de
rutas marítimas habituales a partir de datos AIS*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

En el año 2019 se define el Plan de Transformación Digital de la Armada para promover entre sus miembros la propuesta de nuevos programas de I+D+i que permitan identificar posibles aplicaciones de nuevas tecnologías en el ámbito de sus competencias. Este mismo año, el Almirante Jefe del Estado Mayor de la Armada (AJEMA) consideró oportuno la apertura de un nuevo proyecto de investigación que delegaría de forma conjunta en el Centro Universitario de la Defensa (CUD) y el COVAM (Centro de Operaciones y Vigilancia de Acción Marítima) para el desarrollo de un sistema de Inteligencia Artificial que permitiera exprimir al máximo los potenciales beneficios que pueden proveer las enormes cantidades de datos que emiten los barcos en la mar de forma continua, principalmente en forma de mensajes AIS.

Alineado con este proyecto, se concibe este Trabajo de Fin de Grado como una propuesta de investigación para el análisis del desempeño de distintos algoritmos de clusterización en el entorno marítimo, realizando un análisis offline de una base de datos históricos compuesta de mensajes de identificación de buques en la mar. Para el éxito del proyecto se realiza una revisión del estado del arte de la inteligencia artificial en el ámbito marítimo y, en concreto, de los algoritmos de clusterización más conocidos. Tras estudiar las diferentes opciones, se considera que los algoritmos de clusterización basados en densidad serían los más apropiados para esta aplicación, por lo que se lleva a cabo la implementación de los más representativos, de manera que se puedan visualizar los resultados con el objetivo de optimizar sus parámetros y sacar información de valor. Los resultados muestran que es posible detectar automáticamente las zonas de parada o fondeo, lo que resulta de gran utilidad para particionar y realizar un análisis posterior de las rutas.

PALABRAS CLAVE

Inteligencia Artificial, COVAM, AIS, Algoritmos de clusterización, *Maritime Domain Awareness*

AGRADECIMIENTOS

Quiero que mis primeras palabras de agradecimiento estén dirigidas a mis compañeros de la promoción 421-151 por navegar a mi lado estos últimos cinco años, especialmente a aquellos con los que he establecido un vínculo más estrecho y que han hecho de mi vida en la Escuela un baúl de mil recuerdos y experiencias.

También debo mostrar mi agradecimiento a Miguel Rodelgo Lacruz, tutor y director de este proyecto, por orientarme de forma cercana durante el desarrollo del mismo y despertar en mí la curiosidad por el tema que se ha trabajado.

Por último, agradecer de forma especial y cariñosa a mi familia todo el apoyo que muestran cada día, animándome siempre a seguir adelante y a dar lo mejor de mí mismo.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	6
1 Introducción y objetivos	7
1.1 Contexto y motivación	7
1.2 Objetivos	10
1.3 Estructura de la memoria	10
2 Estado del arte	12
2.1 Big Data	12
2.1.1 Concepto de Big Data	12
2.1.2 Big Data Analytics.....	15
2.1.3 Inteligencia Artificial (IA) en el entorno del Big Data	18
2.2 Maritime Domain Awareness (MDA)	21
2.2.1 Introducción al MDA.....	21
2.2.2 COVAM	22
2.2.3 Automatic Identification System (AIS)	24
2.2.3.1 Descripción del sistema.....	24
2.2.3.2 Funcionamiento básico.....	25
2.3 Trabajos relacionados	26
2.4 Análisis de Clusterización.....	28
2.4.1 Introducción al Análisis de Agrupamiento	28
2.4.2 Etapas del análisis de clusterización.....	30
2.4.3 Medidas de asociación.....	30
2.4.4 Clasificación de los tipos de algoritmos	32
2.4.4.1 Algoritmos particionales vs. Algoritmos jerárquicos	32
2.4.4.2 Algoritmos de pertenencia exclusiva vs. no exclusiva.....	33
2.4.4.3 Otras clasificaciones.....	33
2.5 Revisión y comparativa de algoritmos de los clusterización más destacados	34
2.5.1 Descripción de los algoritmos de mayor interés para el proyecto	34
2.5.2 Comparativa de los algoritmos presentados	44
2.6 Comparativa y selección del lenguaje.....	46
2.6.1 Python vs R.....	46
3 Desarrollo del TFG.....	48

3.1 Instalación del software	48
3.1.1 Anaconda®	48
3.1.1.1 Bibliotecas	50
3.2 Datos AIS: Campos del mensaje.....	51
3.3 Evaluación de los algoritmos	53
3.3.1 DBSCAN	54
3.3.2 HDBSCAN	72
3.3.3 Consideraciones y modificaciones al del dataset.....	80
3.3.3.1 DBSCAN – Dataset de mayor tamaño	83
3.3.3.2 HDBSCAN - Dataset de mayor tamaño	88
4 Conclusiones y líneas futuras	90
4.1 Conclusiones	90
4.2 Líneas futuras	90
5 Bibliografía.....	92
Anexo I: Código DBSCAN	98
Anexo II: Código HDBSCAN.....	102
Anexo III: Cálculo de la velocidad real de los buques	106
Anexo IV: Filtro para extraer buques parados	108

ÍNDICE DE FIGURAS

Figura 1-1 DESI de España para el año 2020 [2]	8
Figura 1-2 Clasificación de España entre los Estados miembros 2020 [3]	8
Figura 2-1 Las 10 V del Big Data	13
Figura 2-2 <i>Five V of Big Time Maritime Data</i> [7]	14
Figura 2-3 Tipos de estructuras de datos [8]	15
Figura 2-4 Esquema del proceso de <i>Data Analytics</i> [12]	17
Figura 2-5 Gráfica de complejidad del análisis vs valor que aporta [13]	18
Figura 2-6 <i>Data virtuous cycle</i> [16]	19
Figura 2-7 Inteligencia Artificial y sus subconjuntos [17]	20
Figura 2-8 Sala del COVAM [20]	23
Figura 2-9 Mapa elaborado por MarineTraffic a partir de información transmitida en tiempo real por diferentes buques [24]	24
Figura 2-10 <i>Plotter</i> Radar-AIS [25]	25
Figura 2-11 <i>Maritime Traffic Knowledge Discovery and Representation System</i> [32]	27
Figura 2-12 Maritime vessel anomaly as defined in the H2020 BigDataOcean project [37]	28
Figura 2-13 Ejemplos de grupos en R^2	29
Figura 2-14 Distancias Minkowski [39]	31
Figura 2-15 Algoritmos de clusterización no jerárquicos o particionales	32
Figura 2-16 Algoritmos de clusterización jerárquica	33
Figura 2-17 Tipos de algoritmos	34
Figura 2-18 Gráfica explicativa algoritmo <i>Chain-map</i>	35
Figura 2-19 Algoritmo K-means [41]	37
Figura 2-20 Diagrama de Voronoi con 12 clústeres [45]	37
Figura 2-21 K-medoids <i>clustering output</i> [47]	38
Figura 2-22 Funcionamiento del algoritmo DBSCAN [48]	39
Figura 2-23 K-Means (izquierda) no es capaz de separar grupos no linealmente separables, mientras DBSCAN (derecha) si es lo es [49]	40
Figura 2-24 Distancias para ordenación en OPTICS [50]	41
Figura 2-25 DENCLUE <i>hill-climbing</i> (izquierda) frente a DENCLUE 2.0 [52]	42
Figura 2-26 Algoritmo STING: división de la rejilla [54]	42
Figura 2-27 Ejemplo de algoritmo CLIQUE [55]	44
Figura 2-28 Logotipo R [57]	46
Figura 2-29 Logotipo Python [59]	46
Figura 3-1 Distribución Anaconda [61]	49

Figura 3-2 Anaconda Navigator	50
Figura 3-3 Suministro de combustible en alta mar de un barco a otro (<i>bunkering</i>) [64]	53
Figura 3-4 Gráfica k-distance para MinPts=20	55
Figura 3-5 Zoom en la zona de interés de la Figura 3-4.....	56
Figura 3-6 Plot para MinPts = 20 y $\varepsilon=0.1$	56
Figura 3-7 Zoom del plot (MinPts = 20 y $\varepsilon = 0.1$) en la zona del Estrecho de Gibraltar	58
Figura 3-8 Estrecho de Gibraltar en OpenCPN.....	58
Figura 3-9 Plot para MinPts = 20 y $\varepsilon=0.1$, centrado en Algeciras y el Mar de Alborán.....	59
Figura 3-10 Plot centrado en Algeciras para $\varepsilon = 0.05$	60
Figura 3-11 Plot centrado en Algeciras para $\varepsilon=0.03$	60
Figura 3-12 Mapa de Algeciras - Gibraltar extraído de OpenCPN.....	61
Figura 3-13 Plot centrado en Algeciras para $\varepsilon=0.02$	62
Figura 3-14 Plot centrado en Algeciras para $\varepsilon=0.014$	62
Figura 3-15 Plot para $\varepsilon=0.1$, escala inicial	63
Figura 3-16 Plot para $\varepsilon=0.014$, escala inicial	63
Figura 3-17 Ampliación del plot sobre zonas destacadas	64
Figura 3-18 Vista de Google Maps del Guadalquivir a su paso por Sevilla	65
Figura 3-19 Vista de Google Maps de la Bahía de Cádiz	65
Figura 3-20 Distancia euclídea vs Distancia Haversine (basado en [67]).....	66
Figura 3-21 Plot para $\varepsilon = 3$ km utilizando distancia Haversine	67
Figura 3-22 Plot de Algeciras para $\varepsilon = 3$ km, distancia Haversine	67
Figura 3-23 Plot de Algeciras para $\varepsilon = 2$ km, distancia Haversine	68
Figura 3-24 Plot de Algeciras para $\varepsilon = 1.5$ km, distancia Haversine	68
Figura 3-25 Mapa de Algeciras-Gibraltar extraído de Google Maps.....	69
Figura 3-26 Zoom de Algeciras para $\varepsilon = 1$ km y $\text{min_samples}=10$, distancia Haversine.....	70
Figura 3-27 Plot general $\varepsilon = 1$ km y $\text{min_samples}=10$, distancia Haversine	70
Figura 3-28 Outliers resultantes para $\varepsilon = 1$ km y $\text{min_samples}=10$, distancia Haversine.....	71
Figura 3-29 Zoom sobre los outliers del Mar de Alborán	71
Figura 3-30 Distribución de la probabilidad de un conjunto de datos X [68].....	73
Figura 3-31 Representación del umbral global para un conjunto de datos arbitrario [68].....	73
Figura 3-32 Representación del umbral secundario para dos conjuntos de datos arbitrarios [68]...74	
Figura 3-33 Árbol de jerarquías para un conjunto de datos arbitrario [68].....	74
Figura 3-34 Comparativa árboles jerárquicos para distintos parámetros de <i>min_cluster_size</i>	75
Figura 3-35 Plot para <i>min_cluster_size</i> = 120.....	76
Figura 3-36 Plot de Algeciras, <i>min_cluster_size</i> = 120.....	77
Figura 3-37 Plot de la Bahía de Cádiz, <i>min_cluster_size</i> = 120.....	77

Figura 3-38 Plot para <i>min_cluster_size</i> = 120 y <i>min_samples</i> = 5.....	78
Figura 3-39 Algeciras - <i>min_cluster_size</i> = 120 y <i>min_samples</i> = 5	79
Figura 3-40 Cádiz - <i>min_cluster_size</i> = 120 y <i>min_samples</i> = 5	79
Figura 3-41 Plot <i>min_cluster_size</i> = 120, <i>min_samples</i> = 5 y <i>cluster_selection_epsilon</i> = 1km	80
Figura 3-42 Box plot del campo “ <i>sog</i> ”	81
Figura 3-43 <i>Dataframe</i> incluyendo los valores de velocidad calculados.....	82
Figura 3-44 Diagrama de dispersión “ <i>sog</i> ” vs “ <i>speed</i> ”	82
Figura 3-45 Zonas de parada usando el algoritmo DBSCAN - MinPts = 50 y ϵ = 1km.....	84
Figura 3-46 Zonas de parada, algoritmo DBSCAN (Sin outliers) - MinPts = 50 y ϵ = 1km	84
Figura 3-47 Outliers resultantes de la clusterización con DBSCAN – MinPts = 50 y ϵ = 1km.....	85
Figura 3-48 Rutas marítimas habituales en el Estrecho de Gibraltar, en Google Maps.....	85
Figura 3-49 Recorte extraído de una carta náutica digital en la zona del Estrecho de Gibraltar	86
Figura 3-50 Discriminación de los muelles de puertos importantes en el Estrecho de Gibraltar	87
Figura 3-51 Zonas de parada utilizando el algoritmo DBSCAN (Sin outliers) - MinPts = 50 y ϵ = 2km.....	87
Figura 3-52 Zonas de parada utilizando el algoritmo HDBSCAN - <i>min_cluster_size</i> = 110 y <i>min_samples</i> = 105.....	88
Figura 3-53 Zonas de parada utilizando el algoritmo HDBSCAN (Sin outliers) - <i>min_cluster_size</i> = 110 y <i>min_samples</i> = 105.....	88
Figura 3-54 Outliers resultantes de la clusterización con DBSCAN – <i>min_cluster_size</i> = 110 y <i>min_samples</i> = 105.....	89

ÍNDICE DE TABLAS

Tabla 2-1 Test de rendimiento para CLARANS; DBSCAN y STING (Basado en [53])	43
Tabla 2-2 Tabla comparativa de los algoritmos más destacados	45
Tabla 2-3 Python vs R	47
Tabla 3-1 Campos de un mensaje AIS	52
Tabla 3-2 Parámetros de validación para diferentes valores de MinPts	57
Tabla 3-3 Parámetros de validación para diferentes valores de <i>min_cluster_size</i>	76
Tabla 3-4 Parámetros de validación para diferentes valores de <i>min_samples</i>	78

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto y motivación

Haciéndolo a una velocidad que crece de forma exponencial, nuestra sociedad tiene la obligación de adaptarse y evolucionar al ritmo que marca el desarrollo de la tecnología. Cada vez es más común escuchar palabras, muchas veces prestadas por el inglés, que hacen referencia a nuevos conceptos que revolucionan los métodos tradicionales y exigen flexibilidad y estar abierto a nuevos cambios.

Un término al que se hace referencia de forma habitual es la digitalización. Según lo define la Cámara de Comercio de España [1], la digitalización significa adaptación y transformación con la tecnología, oportunidad, competitividad, flexibilidad, crecimiento, innovación, liderazgo, personalización y adaptación a los clientes y a sus nuevas demandas y expectativas. Los expertos aseguran que el mundo va a cambiar más en los próximos 20 años que en toda la historia de la humanidad, una transformación vertiginosa que va a estar protagonizada por las nuevas tecnologías. La hiperconectividad, el internet de las cosas (IoT), la inteligencia artificial o la impresión 3D son algunas de las innovaciones tecnológicas que constituyen la punta de lanza de esta cuarta revolución industrial a la que, por primera vez, la economía española y, por ende, las empresas españolas, están a tiempo de sumarse.

Cada año, la Comisión Europea supervisa los avances digitales de los Estados miembros a través de los informes del Índice de la Economía y la Sociedad Digitales (DESI), los cuales combinan para cada país datos cuantitativos procedentes de los indicadores del DESI para sus cinco dimensiones (conectividad, capital humano, uso de internet, integración de la tecnología digital y servicios públicos digitales), con análisis sobre las políticas y mejores prácticas del país.

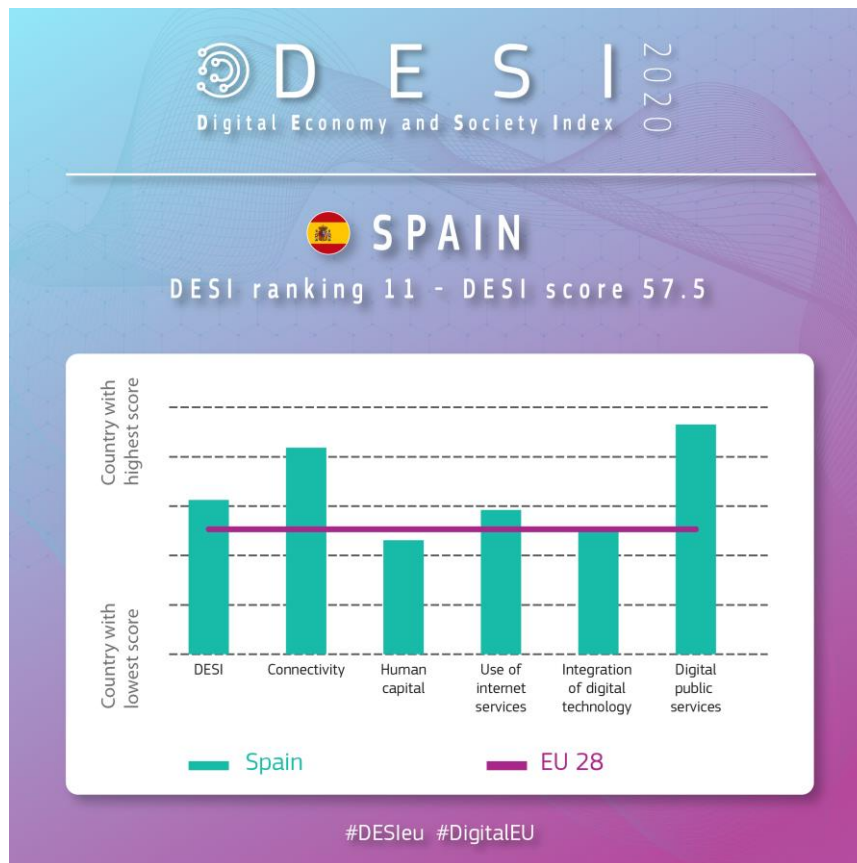


Figura 1-1 DESI de España para el año 2020 [2]

Como se puede ver en la Figura 1-1, el informe del DESI incluye un análisis de la conectividad de banda ancha, las competencias digitales, el uso de internet, la digitalización de las empresas, los servicios públicos digitales, las tecnologías emergentes, la ciberseguridad, el sector de las TIC y su gasto en I+D+i, y el uso de los fondos de Horizonte 2020 por parte de los Estados miembros. España ocupa el puesto número 11 entre los 28 Estados miembros de la UE en la edición 2020 del índice del DESI (Ver Figura 1-2), sobre la base de datos previos a la pandemia.

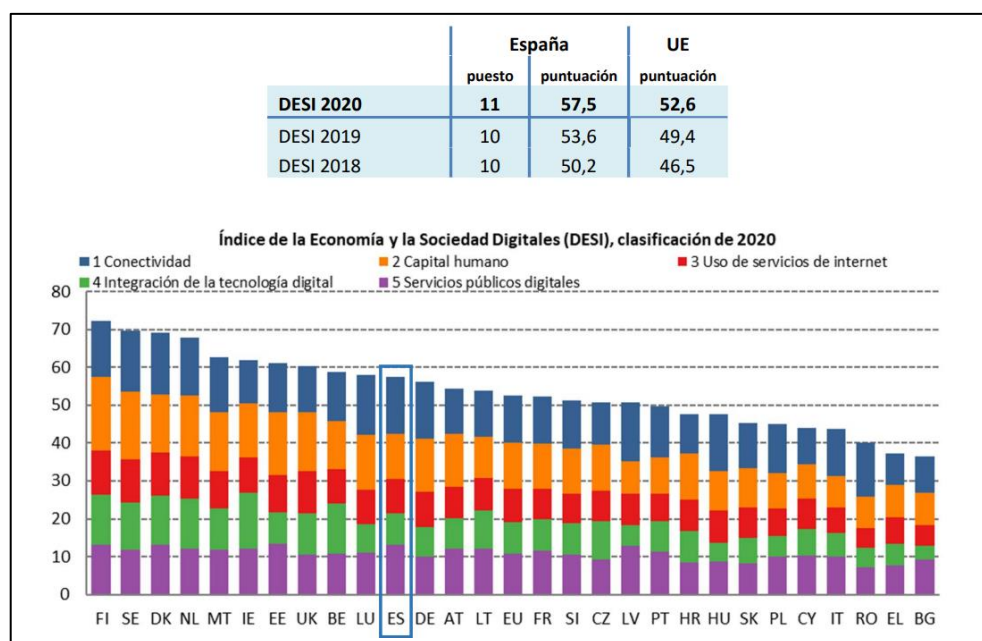


Figura 1-2 Clasificación de España entre los Estados miembros 2020 [3]

La actual crisis del COVID-19 está teniendo un impacto significativo en indicadores sociales clave, relacionados con el uso de los servicios de Internet por parte de los ciudadanos. La pandemia ha puesto de manifiesto la importancia de los activos digitales, y la forma en que las redes y la conectividad, los datos, la inteligencia artificial y la supercomputación sustentan nuestras economías permitiendo que los trabajos no se vean interrumpidos, llevando a cabo un seguimiento de la propagación del virus y acelerando la búsqueda de medicamentos y vacunas. En el caso de nuestro país, se adoptaron varias medidas en el ámbito digital para hacer frente a la crisis relacionada con la COVID-19 y, aunque se realizaron esfuerzos importantes para combatir los efectos de la misma, España suele encontrarse por debajo de la media de la UE en los indicadores de capacidades digitales y tiene un rendimiento relativamente débil en la digitalización de las empresas, especialmente de las PYMEs, que siguen sin sacar provecho del potencial del comercio electrónico. Consciente de ello, España planea movilizar en los dos próximos años más de 200 millones de euros para la recuperación [3].

Contando a partir de la primera revolución industrial, iniciada a mitad del siglo XVIII, e inmersos aún en la tercera revolución industrial, las industrias se encuentran a las puertas de la ya conocida cuarta revolución industrial, o *Industria 4.0* (I4.0), impulsada y sustentada en la digitalización. Como cualquier empresa, la Armada española se ha sumado al cambio y, aprovechando el apoyo del CESTIC (Centro de Sistemas y Tecnologías de Información y Comunicaciones de la Secretaría de Estado de Defensa), ha iniciado un proceso de transformación digital (*Armada 4.0*) que le permitirá adaptarse a un entorno en constante evolución, con el objetivo de maximizar su eficacia, y la eficiencia de sus sistemas logísticos, financieros y humanos. Como elemento diferenciador respecto a las estrategias adoptadas por otras organizaciones, el Almirante Jefe del Estado Mayor de la Armada (AJEMA) ha señalado en numerosas ocasiones que el personal debe situarse en el centro de reflexión, ya que es necesario que la búsqueda apunte a la utilización óptima del recurso humano como motor principal de este proceso y porque, al fin y al cabo, cualquier nueva tecnología va a estar siempre dirigida a facilitar su trabajo.

Con este propósito, se define el Plan de Transformación Digital de la Armada en el año 2019, en el cual se establece, entre otras propuestas, la conveniencia de “Proponer aquellos programas de I+D+i que permitan identificar las posibles aplicaciones de las nuevas tecnologías a la actividad de la Armada”. A raíz de esto nace la Oficina de Innovación de la Armada (OFINAR) para fomentar, evaluar y apoyar el desarrollo de cualquier propuesta viable realizada por el personal de la Armada, tanto civil como militar, de cara a mejorar la capacidad de respuesta ante las necesidades específicas de la Armada. Como es el caso de cualquier empresa, resulta de especial interés para la Armada el tratamiento de grandes volúmenes de datos (“*Big Data*”), el análisis de la información (“*Data Analytics*”) y la explotación de la Inteligencia Artificial en todas sus formas. Lo que se busca conocer en profundidad, en su caso, no es un cliente al que satisfacer, sino el entorno marítimo. El instrumento del que se vale la Armada para fusionar y analizar la información que se recibe mediante un flujo continuo de datos procedente de numerosas fuentes, principalmente a través del AIS (*Automatic Identification System*), es el Centro de Operaciones y Vigilancia de Acción Marítima (COVAM), con base en Cartagena. La explotación de estas nuevas tecnologías en una actividad operativa tan significativa para la Armada como es el “Conocimiento del Entorno Marítimo” (CEM) resultaría francamente beneficioso para aumentar la seguridad en el mar, atendiendo a la vigilancia del tráfico marítimo y la gestión de anomalías respecto a patrones de comportamiento habituales.

Es por ello que, el 20 de octubre de 2019, el AJEMA consideró oportuno la apertura de un Proyecto de Investigación (I+D+i) a desarrollar por el Centro Universitario de la Defensa (CUD) de la Escuela Naval Militar (ENM) para la aplicación de Inteligencia Artificial en el COVAM. En este escrito se expone, entre otros, el beneficio que puede aportar la capacidad de la ENM en materia de investigación, y la implicación de los futuros Oficiales en el desarrollo de soluciones tecnológicas, parte en la cual se encuadra este trabajo.

1.2 Objetivos

El objetivo principal del trabajo que a continuación se desarrolla es llevar a cabo el análisis del funcionamiento de diferentes sistemas de aprendizaje automático no supervisado para la detección de rutas habituales y anomalías dentro del entorno marítimo. Concretamente, se tratará de localizar zonas de estacionamiento y fondeo que puedan identificarse como puntos de inicio o final de las rutas, de manera que se puedan particionar para análisis posteriores. Se analizará el comportamiento de los algoritmos de agrupamiento seleccionados frente a los datos que se le proporcionarán, con el propósito de alcanzar de forma heurística la mejor combinación de parámetros para conseguir resultados de valor.

Relacionados con el mismo, se proponen los siguientes objetivos específicos:

- Revisión del estado del arte de las técnicas de análisis de clusterización existentes y la importancia de su empleo en el ámbito del Big Data marítimo como herramienta de seguridad en la navegación.
- Análisis de trabajos publicados relacionados directamente con el propósito del trabajo.
- Selección de algoritmos de agrupamiento para su implementación, basándose en criterios de aplicabilidad en el ámbito marítimo.
- Implementación de los algoritmos para su comparación cualitativa y de rendimiento.
- Análisis offline a partir de una fracción de una base de datos AIS proporcionada por el COVAM para la optimización de los parámetros de funcionamiento.

Además, otros objetivos complementarios que se plantean son los siguientes:

- Familiarización con herramientas de Machine Learning e inteligencia artificial para el trabajo con Big Data.
- Ampliar conocimientos de programación trabajando con el lenguaje Python.
- Exponer conocimientos adquiridos durante la carrera en asignaturas como Informática para la ingeniería.

1.3 Estructura de la memoria

Habiendo contextualizado este trabajo, y una vez establecidos los objetivos, se describe a continuación como se va organiza la estructura de esta memoria en sus diferentes apartados.

- En el Capítulo 1 se introduce al lector al contexto bajo el cual se desarrolla este trabajo, justificando los diferentes hechos que motivan el proyecto, y se establecen los objetivos a alcanzar con el mismo.
- En el Capítulo 2 se lleva a cabo una revisión del estado del arte, haciendo un repaso de los conceptos teóricos de relevancia para entender este proyecto, así como la comparación de las distintas posibilidades que podemos plantear a la hora de abordar nuestra investigación. Para ello, se introducen, en primer lugar, los conceptos de *Big Data*, el análisis de datos masivos y la inteligencia artificial aplicada a este ámbito. A continuación, se realiza una introducción al concepto de *Maritime Domain Awareness* (MDA) y se profundiza en el papel que juega el

COVAM como entidad competente de la Armada en este campo. Por último, se ofrece una introducción al análisis de clusterización y se lleva a cabo una revisión comparativa de los potenciales algoritmos a utilizar, para cerrar el capítulo con unas pinceladas sobre otros proyectos de investigación de corte similar.

- En el Capítulo 3 se aplican los algoritmos seleccionados atendiendo a la revisión del capítulo anterior, y se presentan y analizan los resultados que presentan en sus distintas configuraciones.
- En el Capítulo 4 se realiza una reflexión sobre el desarrollo del TFG para sacar conclusiones, y se presentan posibles líneas futuras.
- Para cerrar este trabajo, se incluyen la bibliografía y cuatro anexos.

2 ESTADO DEL ARTE

2.1 Big Data

2.1.1 Concepto de Big Data

En los últimos años, el desarrollo exponencial de las tecnologías y los sistemas de comunicación ha dado pie a la aparición de volúmenes de datos de tales dimensiones (hablamos del paso de terabytes a zetabytes), que las herramientas y procedimientos tradicionales para su gestión y manipulación (como bases de datos relacionales y estadísticas convencionales, o paquetes de visualización) se han visto sobrepasadas. Es por ello que aparece el nuevo término *Big Data* en todos los ámbitos para referirse a esta inmensa cantidad de datos que empresas y organizaciones tratan de aprovechar en el proceso de toma de decisiones, para adoptar así nuevas estrategias que les permitan posicionarse mejor en el escenario competitivo.

Debido a la complejidad de su naturaleza, diferentes autores han intentado plantear una definición para este término, aportando nuevas perspectivas. La primera aproximación moderna a este concepto aparece en el año 2001, cuando Doug Laney define el desafío del crecimiento de los datos como algo tridimensional. Aquí nacen las conocidas V del *Big Data*, siendo tres las que conforman el planteamiento inicial de Laney: Volumen, Velocidad y Variedad. Más adelante se han añadido nuevas magnitudes, dando lugar a otros modelos más específicos como los modelos de las 5 V o de las 7 V. Para entender en profundidad este concepto y el desafío que supone su utilización eficiente, se detalla a continuación un modelo reciente que plantea 10 V (ver Figura 2-1) del *Big Data* [4]:

1. Volumen: es la cantidad masiva de datos que se generan y almacenan, considerándose *Big Data* cuando se alcanzan números de gran magnitud.
2. Velocidad: se refiere a los datos en movimiento por las interconexiones que realizamos, es decir, la rapidez con la que los datos son creados, almacenados y procesados. En muchos casos a la empresa le interesa que este parámetro sea en tiempo real (*streaming*).
3. Variedad: en *Big Data* se manejan archivos de muchos tipos, desde archivos de audio, imagen y vídeo, hasta archivos de registro, captados por sensores, etc.
4. Variabilidad: entendida en este caso como la velocidad inconsistente a la que se cargan y procesan bases de datos, fruto de los múltiples tipos y fuentes de datos. Se trata de un proceso dinámico.

5. Veracidad: es el grado de fiabilidad de la información recibida, tanto en su procedencia como en su contenido.
6. Validez: se refiere a la calidad de los datos recibidos, su limpieza y su adecuación al uso que se les pretende dar. Se deben adoptar buenas prácticas de gobernanza de datos para garantizar coherencia y calidad. Los científicos de datos pasan entre un 50% y un 80% de su tiempo limpiando datos antes de considerarlos para su análisis [5].
7. Vulnerabilidad: es uno de los campos que más interesa reforzar en la actualidad, pues el robo y secuestro de datos supone una de las mayores preocupaciones para las empresas. Las estimaciones indican que sólo un 20% del universo digital cuenta con protecciones frente al robo, medidas de privacidad o cumplimiento de reglamentaciones [6].
8. Volatilidad: después de un tiempo, ciertos datos comienzan a estar obsoletos y, por tanto, pierden utilidad. Esto podría causar una pérdida de rendimiento, por lo que hay que establecer reglas para la disponibilidad y vigencia de los datos, así como para garantizar una rápida recuperación de los mismos cuando sea necesario.
9. Visualización: es necesario usar gráficos sencillos para la ver e interpretar una cantidad de datos que sería imposible comprender con gráficos tradicionales. Para ellos se utilizan distintas técnicas y herramientas como la agrupación, el uso de mapas, etc.
10. Valor: posiblemente la característica más perseguida de todas las presentadas. Las demás cobran sentido cuando se consiguen datos de valor, entendido como factores clave y de interés, particulares de cada empresa.

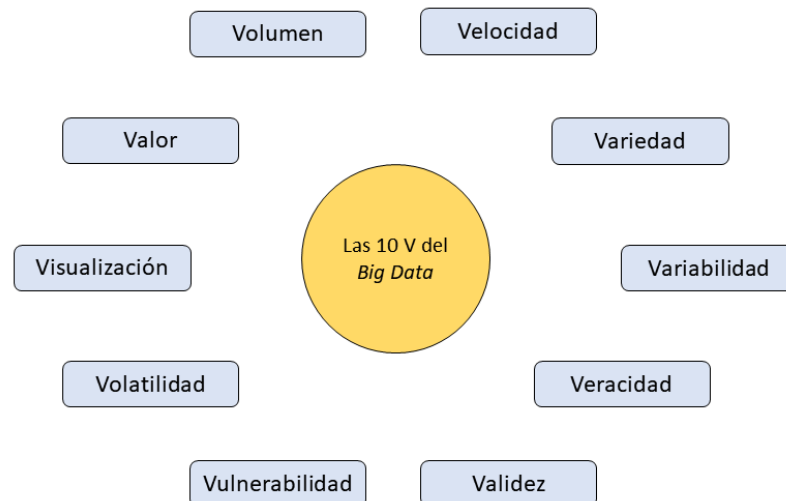


Figura 2-1 Las 10 V del Big Data

Atendiendo a lo establecido en [7] y que se apoya en la Figura 2-2, el *Big Data* Marítimo que conforma el esqueleto de este proyecto (datos AIS) se caracteriza principalmente por cinco de estas 10V que se han presentado:

- Los datos se recogen a distinta *Velocidad*, como las observaciones en tiempo real de las trayectorias de los buques.
- Existe una amplia *Variedad* en el tipo de datos publicados, recogidos y compartidos en conjunto, como es el caso de los campos AIS.
- Evidentemente, por su naturaleza los datos marítimos incluyen grandes *Volúmenes* de datos recogidos por investigadores y científicos.
- Las tareas que procesan y analizan los datos les añaden un *Valor* importante por la utilidad que proporcionan.
- Como ocurre en los datos AIS, los datos son enviados muchas veces duplicados o con información redundante, y esta información a veces no proviene de sensores, sino que es un operador quien la distribuye. Por ello, es necesario prestar atención a su *Veracidad*.

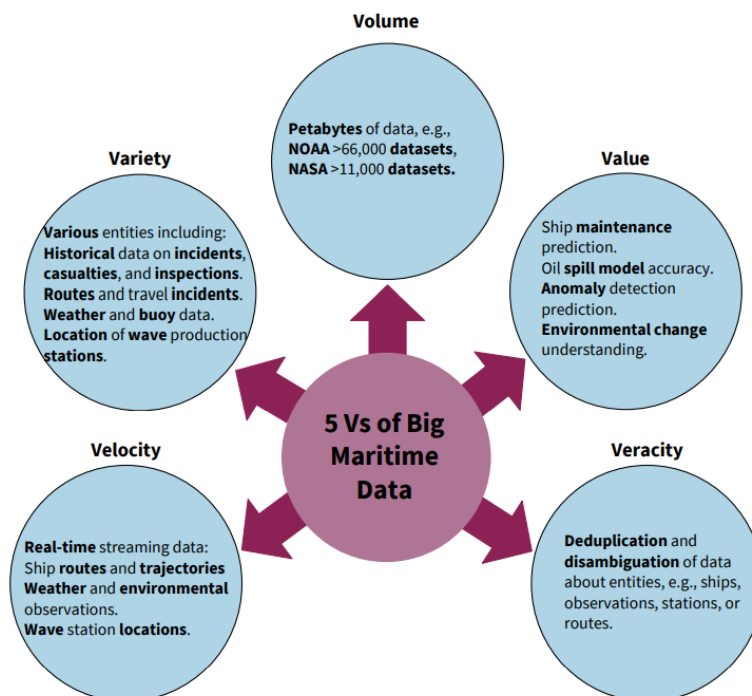


Figura 2-2 Five V of Big Time Maritime Data [7]

Es conveniente también tener en cuenta los distintos tipos de estructura (ver Figura 2-3) que pueden presentar los datos, a fin de entender la cuarta V que ha sido mencionada previamente:

- Datos estructurados: datos con un formato, tamaño y longitud definidas, lo que permite que su procesamiento sea más sencillo. Las bases de datos, como las que utilizaremos para desarrollar en el trabajo, son un ejemplo de datos estructurados.
- Datos no estructurados: Su estructura interna no está bien definida y los datos se encuentran desorganizados. Estos valores no tienen valor si no son ordenados y correctamente identificados y almacenados. Ejemplos de este tipo de estructura son los archivos de vídeo, audios o documentos PDF.

- Datos semiestructurados: En este caso encontramos información no regular con una organización definida, como es el caso de los formatos HTML, XML o JSON.

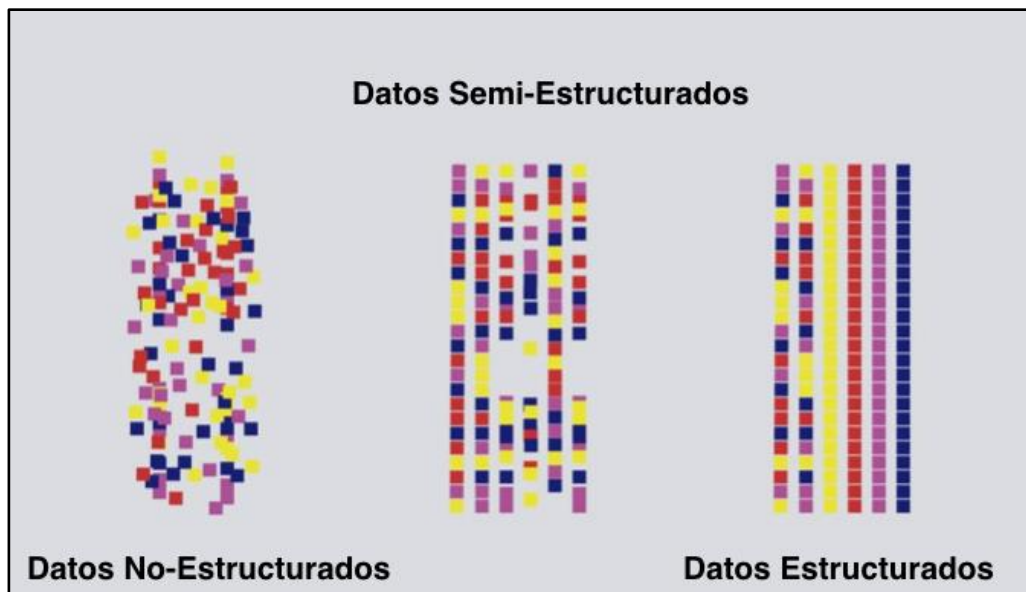


Figura 2-3 Tipos de estructuras de datos [8]

La naturaleza complicada y variable del *Big Data* radica principalmente en el formato no estructurado de la mayor parte de los datos que se generan en la actualidad. Esto se debe a la evolución del Internet de las cosas (IoT) y a que no solo humanos, sino máquinas y sensores también participan de forma activa en la producción masiva y creciente de los datos. Estos datos se procesan, se les da formato y se almacenan, pero en lugar de cargarse en una sola fuente, lo hacen de manera distribuida. De este modo, grandes volúmenes de datos pueden ser divididos y distribuidos entre varios procesadores. Una vez estructurados, se logra la búsqueda y análisis de los datos a una mayor velocidad.

Un estudio realizado por la consultora IDC Digital Universe calculó que el tamaño del universo digital alcanzó los 2,8 zetabytes (ZB) en 2012, y estimaba que para este año 2020 se alcanzarían los 40 ZB [6]. Sin embargo, estos cálculos tuvieron que verse rectificadas a mitad de año debido a la situación que provocó la propagación del COVID-19 por todo el mundo, llevando a las personas a permanecer más tiempo en sus hogares, trabajando de forma telemática o simplemente realizando un consumo ocioso del mundo virtual. De esta manera, la siguiente apuesta de IDC fue que más de 59 ZB de datos serían creados, capturados, copiados y consumidos en el mundo. Aunque a priori podamos pensar que este incremento se trata de nuevo contenido, lo cierto es que la pandemia ha potenciado de forma notable el consumo de contenido replicado. Los expertos de IDC estiman que para el año 2024 este índice sea de 1:10 [9].

Es por ello que estas magnitudes sugieren la necesidad de un análisis y filtrado meticuloso de los datos, pues el potencial de este concepto se encuentra en el valor que se puede encontrar en su conjunto. Así, *Big Data* no es una tecnología en sí misma, sino un planteamiento de trabajo para la obtención de valor a partir de los grandes volúmenes de datos que se generan hoy en día.

2.1.2 *Big Data Analytics*

Identificar el valor del *Big Data* no es únicamente analizar los datos, sino que se trata de un proceso de descubrimiento que requiere que los analistas, empresarios y ejecutivos se planteen las preguntas

correctas, identifiquen patrones, tomen decisiones y predigan comportamientos. Este nuevo paradigma bajo el cual trabajan multitud de administraciones públicas y empresas privadas para ofrecer experiencias mucho más personalizadas se conoce como *Big Data Analytics*.

La analítica inteligente de datos masivos combina diferentes sistemas de alta tecnología y de matemáticas que analizan de forma conjunta la información, y le dan un significado muy valioso para empresas y gobiernos. Como se cita en [10], del documento técnico “El big data conoce la analítica del big data”:

“La analítica de alto desempeño le permite hacer cosas en las que nunca pensó porque los volúmenes de datos eran demasiado altos. Por ejemplo, puede obtener insights puntuales acerca de oportunidades efímeras, obtener respuestas precisas a problemas difíciles de resolver y descubrir nuevas oportunidades de crecimiento – todo mientras utiliza recursos de TI con mayor eficiencia.”

Big Data Analytics trae consigo un abanico de conceptos digitales nuevos [11] que las empresas deben manejar, como el de *data lakes* (el repositorio donde la información en bruto espera a ser analizada), el aprendizaje automático, la minería de datos (*data mining*), o el *clustering*, sobre el que estará enfocado este trabajo y el cual detallaremos más adelante.

Podemos resumir este proceso de análisis en cinco estaciones (ver Figura 2-4):

- Primero se recolectan los datos de las distintas fuentes.
- Después se procesan estos datos para estructurarlos de manera que nuestro sistema sea capaz de comprenderlos.
- A continuación, se almacenarán en un *data lake* o *data warehouse* a la espera de ser analizados.
- Posteriormente se analizan los datos con herramientas de análisis.
- Finalmente, haremos uso de inteligencia artificial y *Machine Learning* para obtener tendencias, patrones, predicciones y anomalías.

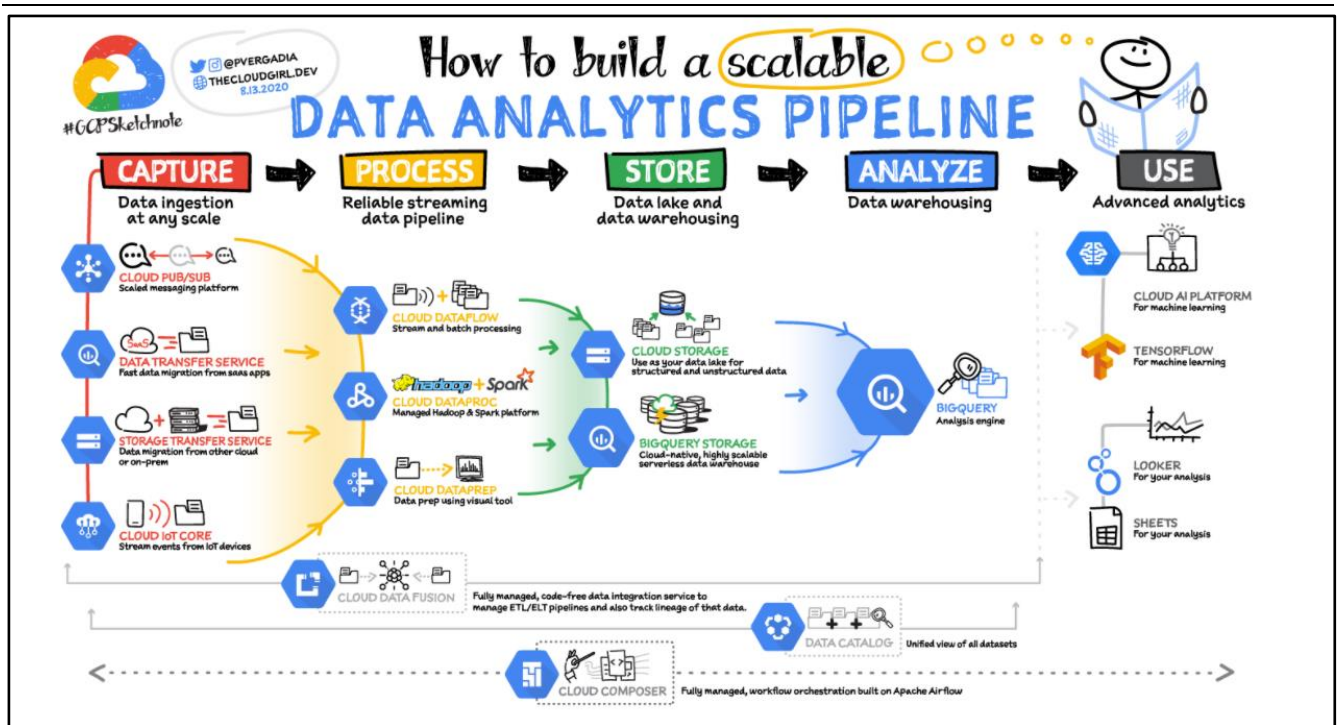


Figura 2-4 Esquema del proceso de *Data Analytics* [12]

El análisis del *Big Data* es un campo muy amplio que abarca todo tipo de datos y objetivos. Es por eso que existen cuatro tipos principales de *Big Data Analytics*, cada uno con una meta distinta y un ámbito de aplicación diferente (Véase Figura 2-5)

- **Descriptivo:** explica con la ayuda de los datos y mediante gráficos o informes lo que sucedió en el pasado, pero no lo que pasará en el futuro. Desarrollando indicadores clave de rendimiento (KPIs, *key performance indicators*), estas estrategias pueden ayudar a realizar un seguimiento de los éxitos y los fracasos.
- **De diagnóstico:** similar al anterior, pero trata de buscar una respuesta al por qué sucedió.
- **Predictivo:** analiza los datos para predecir lo que podría suceder. Las técnicas predictivas usan datos históricos para identificar tendencias y determinar posibles resultados. Entre otras, las redes neuronales o los árboles de decisión entran dentro de esta disciplina.
- **Prescriptivo:** Es una evolución del anterior basado en procesos de automatización, que recomiendan como proceder a partir de los datos que tenemos. Es verdaderamente útil para las empresas, sobre todo en situaciones de incertidumbre en las que es necesario tomar una decisión.

Estos tipos de análisis deben usarse de manera combinada para dar una visión completa de la situación de una empresa y las oportunidades y necesidades que tiene, pudiendo así tomar decisiones efectivas que permitan conseguir ventaja frente a los competidores.

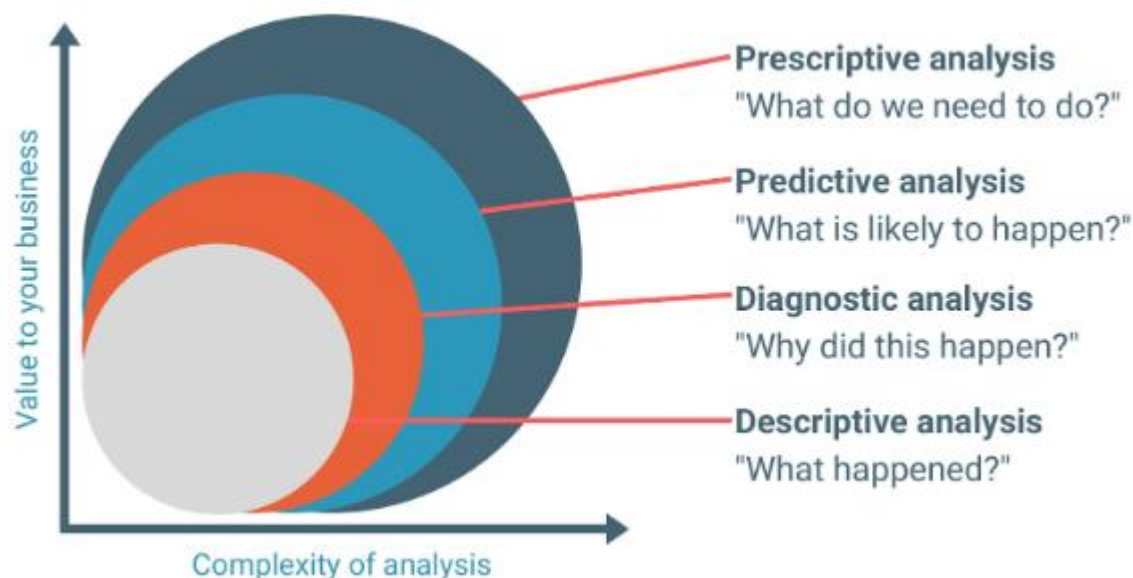


Figura 2-5 Gráfica de complejidad del análisis vs valor que aporta [13]

El correcto empleo de las técnicas de *Big Data Analytics* permitirá a las administraciones y empresas conocer mejor a sus usuarios, encontrar oportunidades antes no detectables, ofrecer un servicio mejor, o incluso mitigar el fraude. Es por esto que el entorno del *Big Data* se puede consagrar como una nueva revolución que no ha hecho más que empezar, y que está obligando al mundo a cambiar el estilo de los negocios, todo de la mano del desarrollo de la inteligencia artificial.

2.1.3 Inteligencia Artificial (IA) en el entorno del Big Data

Una vez introducido el concepto de *Big Data* y su análisis, y con objeto de ir encaminando la información aquí presentada hacia el desarrollo de este proyecto, es necesario hablar de la inteligencia artificial en el entorno de los datos masivos.

Una definición simple de este término sería aquella que la describe como la inteligencia llevada a cabo por máquinas. Aunque hace años pudiera parecer imposible dotar a una máquina de una capacidad similar al razonamiento humano, hoy es una realidad y está presente en multitud de campos de los más variados. La clave para hacer esto posible son las matemáticas, y es que, tal como se define en [14], la IA es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano.

Existe una relación recíproca entre el *Big Data* y la IA. Estas dos tecnologías se retroalimentan, pues la Inteligencia Artificial se nutre de datos, con los que aprende y potencia su capacidad, y el *Big Data* encuentra su máxima explotación mediante el uso de la IA. Atendiendo a las afirmaciones del experto estratega digital Glenn Gruber [15]:

“Hoy en día queremos tantos datos que podamos conseguir, no sólo para obtener una mejor visión de los problemas empresariales que estamos tratando de resolver, sino porque cuantos más datos proporcionamos a los modelos de aprendizaje automático, mejores se vuelven” Dice Gruber. *“Es un círculo vicioso”* (Véase Figura 2-6)

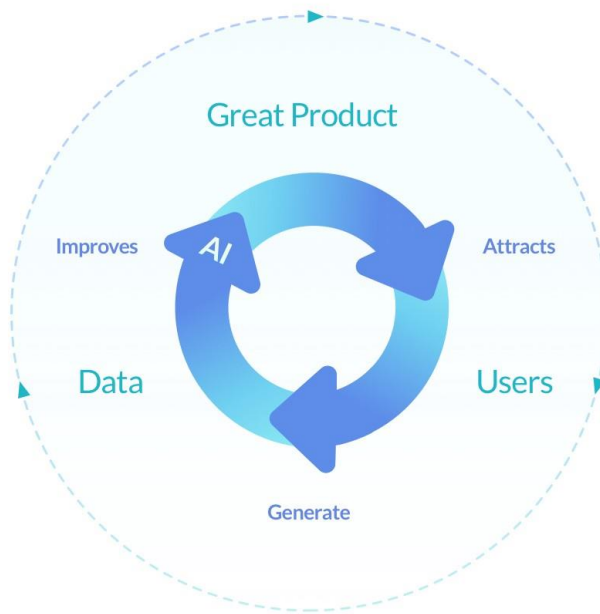


Figura 2-6 Data virtuous cycle [16]

La Inteligencia Artificial aplicada al *Big Data* proporciona principalmente los siguientes beneficios:

- Detección de anomalías: la IA es capaz de analizar los datos, proporcionados por el *Big Data*, incluso en tiempo real, para detectar desviaciones en la tendencia de los datos.
- Predicción de resultados: basándose en una condición conocida que tiene cierta influencia en el resultado futuro.
- Reconocimiento de patrones: detectando patrones que los seres humanos no serían capaces de detectar, incluyendo el reconocimiento en gráficos.

El resultado de utilizar herramientas de IA como *Machine Learning* (ML) o *Deep Learning* (definidas en la Figura 2-7) dan como resultado una mayor eficiencia en las operaciones empresariales y, en consecuencia, una mayor productividad.



Figura 2-7 Inteligencia Artificial y sus subconjuntos [17]

En el contexto de este trabajo resulta interesante adentrarse un poco más en el concepto de *Machine Learning*. En esta esta rama de la inteligencia artificial la cual empezó a cobrar importancia a partir de los años 80, el ordenador ya no depende de unas reglas y un programador, sino que él mismo puede establecer sus propias reglas y aprender por sí solo. Este aprendizaje se sustenta en el uso de algoritmos, una serie de pasos ordenados que se aplican para realizar una tarea y hacer predicciones. Los tipos de implementación de ML pueden clasificarse en tres categorías diferentes:

- **Aprendizaje supervisado**: los algoritmos trabajan con datos etiquetados (*labeled data*), con intención de encontrar una función que les asigne la etiqueta de salida correcta. Es decir, a partir de un histórico de datos, el algoritmo entrena y aprende a asignar la etiqueta adecuada a los nuevos valores que reciba con un alto grado de precisión.
- **Aprendizaje no supervisado**: se diferencia del aprendizaje supervisado en que, en este caso, las etiquetas de los datos son desconocidas, por lo que el proceso de entrenamiento no se puede realizar. Por tanto, solamente se puede describir la estructura de los datos para intentar encontrar algún tipo de patrón que simplifique el análisis. El carácter de este tipo de aprendizaje es puramente exploratorio, lo que hace fácil comprender que el presente proyecto se posicione bajo esta rama del *Machine Learning*. A partir de una serie de mensajes no etiquetados (mensajes AIS) se buscarán patrones de comportamiento en la mar.
- **Aprendizaje de refuerzo** según la naturaleza de los datos que recibe. En este tipo de aprendizaje se engloban aquellos procedimientos que no pueden clasificarse como uno de los tipos anteriores. El aprendizaje por refuerzo se basa en mejorar la respuesta utilizando un modelo de retroalimentación, en el cual el sistema aprende por ensayo-error a partir de las respuestas que obtiene del mundo exterior como respuestas.

El interés en el mundo empresarial en lo que a Inteligencia Artificial respecta, y debido al amplio abanico de ventajas que supone su implementación, los profesionales en *Big Data* e IA estarán cada vez más solicitados en la Industria 4.0 en los próximos años, a medida que las compañías se encuentren frente a la necesidad de incrementar sus capacidades en el campo del análisis de datos para adaptarse al nuevo formato de negocio.

Tanto es el potencial de estas nuevas tecnologías, que voces como la del filósofo sueco de la Universidad de Oxford, Nick Bostrom, anticipan que existen probabilidades del orden del 90% de que entre 2075 y 2090 haya máquinas tan inteligentes como los humanos. Stephen Hawking se atrevió a dar un paso más, afirmando que en menos de 100 años las máquinas superarán por completo a los humanos [14], y su desarrollo nos permitirá realizar tareas que nunca antes hubiéramos podido realizar por motivo de su complejidad.

2.2 Maritime Domain Awareness (MDA)

En el apartado anterior hemos podido ver la importancia del *Big Data* en la actualidad y el potencial de uso que tiene en todos los ámbitos, demostrando ser una de las tendencias tecnológicas clave en los próximos años, como se afirma en [18]. Sin duda, uno de los campos en los que menos se ha desarrollado y que, sin embargo, es de especial interés por su naturaleza es en seguridad y defensa.

El imparable crecimiento del sector marítimo lleva aparejado un incremento en el tráfico marítimo global y otras actividades de explotación oceánica. Este crecimiento ha contribuido intrínsecamente al desarrollo de una serie de innovaciones tecnológicas que garantizan la protección y la seguridad en la navegación. La fusión de todas estas fuentes de información y tecnologías permite la creación de sistemas automáticos de monitorización que ayudan a crear una imagen en tiempo real de la situación marítima que, en consecuencia, favorece las operaciones marítimas y la creación de nuevas políticas. Es por ello que a continuación se presenta el concepto de *Maritime Domain Awareness*,

2.2.1 Introducción al MDA

El concepto de *Maritime Domain Awareness (MDA)* fue desarrollado a mediados de la primera década de este siglo a raíz de la Nueva Estrategia de Seguridad Marítima de los EEUU, originada a su vez a partir de la Estrategia de Seguridad Nacional posterior a los atentados del 11 de septiembre de 2001 [19].

La *International Maritime Organization (IMO)* define este concepto como el conocimiento efectivo de cualquier cosa relacionada con el dominio marítimo que pueda tener un impacto en materia de seguridad, economía o medioambiente. También existe otro concepto, el cual puede generar confusión pero que, en realidad, se trata de una parte integral del MDA. Este es el *Global Maritime Situational Awareness (GMSA)*, que hace referencia concretamente a la acción de unificar de los datos de todas las agencias y naciones para mejorar el conocimiento del dominio marítimo dentro del concepto general de MDA. Se trata de un término más específico que nace a partir del hecho de que ninguna nación posee todas las capacidades ni ejerce toda la autoridad en el mar por sí mismo.

De acuerdo con el *National Plan to Achieve Maritime Domain Awareness* de los EEUU, el MDA sirve para simplificar el entorno complejo y ambiguo en el que se enmarca la seguridad marítima, persiguiendo principalmente tres metas estratégicas:

- Mejorar la transparencia del dominio marítimo detectando, disuadiendo y enfrentando las amenazas con prontitud y margen hacia los intereses de los EEUU (lo que podemos extrapolar a otras naciones).
- Facilitar la toma de decisiones y respuestas precisas, dinámicas y decididas a todo el espectro de amenazas marítimas

- Mantener la completa aplicación de la ley para asegurar la libertad de la navegación y el discurso eficiente del comercio.

Lograr el MDA depende de la capacidad que se logre para diferenciar las anomalías, pues los datos por sí solos son insuficientes. Estos datos deben ser analizados y tratados con ayuda de algoritmos de análisis por ordenador, de modo que se logre anticipación a la hora de actuar respecto a amenazas y actividades al margen de la ley.

2.2.2 COVAM

Por estar este trabajo bajo su marco, y por su relación directa con el concepto de *Maritime Domain Awareness*, no podemos dejar de hablar uno de los principales órganos en esta materia en nuestro país. Debido a su situación geográfica, España es un punto estratégico en el mundo marítimo, pues podría decirse que es la puerta de entrada a Europa desde el Atlántico. Es por ello que nuestro país tiene grandes intereses en mantener la seguridad en el mar y las rutas comerciales para el desarrollo de su economía.

A lo largo de la historia España ha sido uno de los precursores en la navegación y el transporte marítimo, y pionero en el establecimiento de rutas comerciales trasatlánticas tras el descubrimiento de América en el año 1492, con la creación de la denominada Carrera de Indias. Para asegurar el transporte de las riquezas y evitar el asalto de piratas y mercenarios a lo largo de estas rutas, España formó la llamada Flota de Indias o Flota del Tesoro Español, por lo que no cabe duda que la seguridad en el mar no es un concepto nuevo. Sin embargo, este término ha estado sometido al tiempo y, por tanto, ha ido adaptándose y modernizándose hasta llegar a los sistemas de seguridad y vigilancia actuales.

Actualmente, la Zona Económica Exclusiva (ZEE) de España es vital para el país, por lo que no es de extrañar que la Fuerza de Acción Marítima de la Armada (FAM) haya creado un centro de operaciones desde el cual se pueda controlar el entorno marítimo en cualquier lugar. España transporta por vía marítima el 80 por ciento de sus importaciones y posee una importante flota pesquera a nivel mundial y otras embarcaciones dedicadas al turismo y las prácticas deportivas [20].

Tras la creación de la FAM, se pone en marcha un ambicioso proyecto para centralizar todos los datos de la actividad marítima en aguas españolas y otras zonas de especial interés para la navegación. Es aquí cuando nace el Centro de Operaciones y Vigilancia de Acción Marítima (COVAM), como parte de la estructura de la Armada, con vocación de apoyo a la Comunidad Marítima y bajo el mando del Almirante de Acción Marítima (ALMART).

A lo largo de muchos años, la Armada ha sido depositaria de numerosas responsabilidades y competencias en materia de Seguridad Marítima, pero con el paso del tiempo y la reestructuración orgánica de nuestro país, ha surgido la necesidad de adaptarse a un escenario nuevo. Dadas las características geográficas de nuestro país, rodeado casi en su totalidad por el océano teniendo que vigilar superficies muy extensas, y por el hecho de que, en el caso de ocurrir algún incidente, este normalmente concierne a numerosos departamentos y empresas, se debe realizar un esfuerzo conjunto entre todas las agencias y organizaciones para velar por la seguridad naval. Mientras que otros centros de operaciones enmarcados dentro del organigrama militar atienden de forma exclusiva a esta rama, el COVAM se configura como un elemento de enlace que ofrece sus esfuerzos y su capacidad de vigilancia y conducción de operaciones de a la Comunidad Marítima. Tanto es así, que la definición de Seguridad Marítima aparece en el “Concepto de Seguridad Marítima en la Armada” (CONSEGMAR) como “una actividad cívico-militar de prevención de riesgos y de lucha contra las amenazas del entorno marítimo” [21].

El COVAM se encuentra ubicado en Cartagena, desde donde fusiona y analiza toda la información relativa a los espacios marítimos de interés, lo que se denomina Conocimiento del Entorno Marítimo (CEM) o *Recognized Maritime Picture* (RMP).



Figura 2-8 Sala del COVAM [20]

El COVAM tiene una plantilla compuesta por 30 personas (en su mayoría suboficiales de la Armada), pero la propia estructura y diseño del COVAM permite la participación física de personal de otros organismos de la Administración Marítima debido a la arquitectura de sus sistemas de Mando y Control y la compartimentación interna de sus instalaciones [21].

Sus medios se encuentran distribuidos de forma permanente en tres capas diferenciadas, en función del nivel de clasificación de la información que se maneja:

1. La primera capa está diseñada para la obtención, fusión, análisis y distribución de toda la información de conocimiento general del entorno marítimo.
2. La segunda capa trata información de carácter más sensible, sin que necesariamente se le haya asignado un grado de clasificación militar (puede deberse a sus intereses industriales, comerciales o mediáticos). Se accede a ella mediante sistemas de control de acceso a los que podrían acceder organismos determinados en cada momento.
3. La tercera capa maneja información clasificada de uso y acceso exclusivamente militar.

Su trabajo se desarrolla en una amplia sala (Véase Figura 2-8) con la ayuda de los sistemas informáticos a través de los cuales recibe la información, provista principalmente uno de los elementos más empleados para el seguimiento del tráfico marítimo: el sistema automático de identificación AIS (*Automatic Identification System*). Este sistema es el alimento de muchos proyectos para la extracción de rutas marítimas, predicción de rutas y detección de anomalías, y que utilizaremos como base de datos en el desarrollo de este proyecto.

2.2.3 Automatic Identification System (AIS)

Todo buque en navegación debe cumplir con lo establecido en el Capítulo 5 del Convenio Internacional SOLAS (*Safety Of Life At Sea*) [22] en lo que respecta a la recolección de datos que describan las condiciones en las que se encuentra durante su travesía como, por ejemplo, su rumbo, su posición o su velocidad.

Dado que utilizaremos una base de datos AIS proporcionada por el COVAM en capítulos posteriores, a continuación, se explicarán aspectos relacionados con este sistema de cara al desarrollo del proyecto.

2.2.3.1 Descripción del sistema

El sistema AIS [23] es el desarrollo tecnológico marino más importante en materia de seguridad en la navegación desde la introducción del radar. Este sistema nació como una herramienta para evitar abordajes en la mar y proveer a los buques comerciales de información sobre su entorno. El AIS consigue esto gracias a la transmisión constante de la identidad del buque, su posición, su velocidad, su rumbo y otro tipo de información relevante para las embarcaciones que se encuentran en el radio de alcance.

Combinado con una estación costera, este sistema también ofrece a las autoridades portuarias y cuerpos de seguridad marítima la posibilidad de controlar el tráfico marítimo y reducir los peligros en la navegación, como podemos ver en la Figura 2-9.

Debido a los grandes beneficios que ha propiciado el AIS, esta tecnología se hizo obligatoria en todo el mundo en 2002 para todos los buques de pasajeros y buques de peso bruto superior a 300 toneladas.

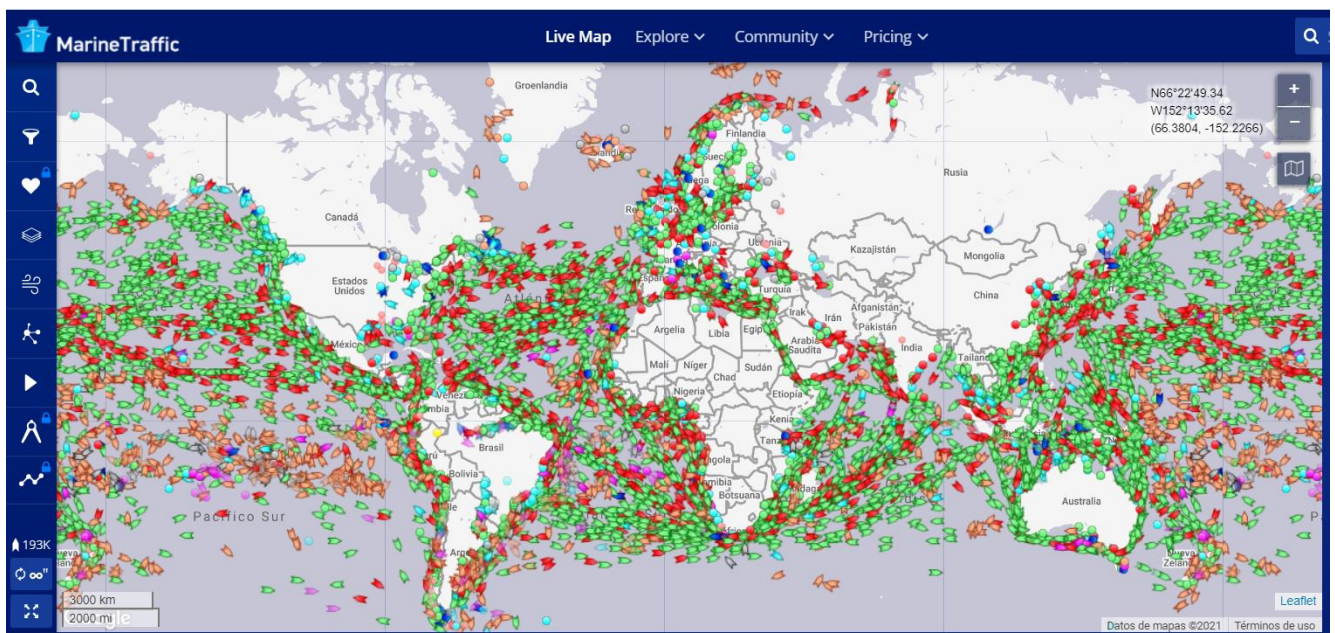


Figura 2-9 Mapa elaborado por MarineTraffic a partir de información transmitida en tiempo real por diferentes buques [24]

2.2.3.2 Funcionamiento básico

El Sistema de Identificación Automática AIS fue desarrollado para permitir que buques y estaciones costeras cercanas fueran capaces de localizarse e identificarse entre sí.

Un transpondedor AIS determina su posición, velocidad y rumbo haciendo uso de un receptor GPS integrado. Esta información se combina con otra información de la navegación importante y se comunica automáticamente con otros buques que cuenten con este sistema a bordo, sin interacción alguna del usuario.

Los transpondedores de otros buques y estaciones costeras reciben la información y la usan para representar gráficamente la situación del tráfico en la zona. El transpondedor puede ir conectado a diferentes tipos de *plotter* (Véase Figura 2-10) y cartas náuticas digitales para presentar las posiciones de los barcos en un formato similar al de un radar. El alcance de este sistema se corresponde al del tipo de emisión con el que funciona, VHF (*Very High Frequency*).

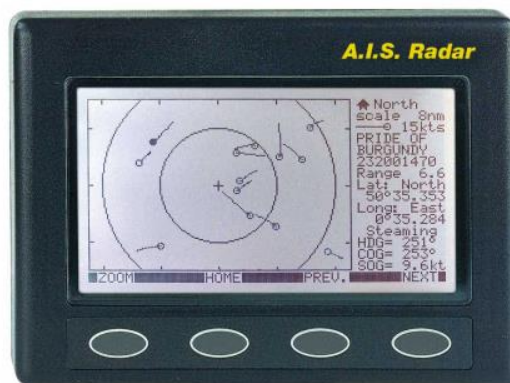


Figura 2-10 *Plotter Radar-AIS* [25]

Existen dos tipos de transpondedores:

- Clase A: para barcos recogidos dentro del Convenio SOLAS, ofrece más información que los de clase B.
- Clase B: diseñados para barcos pequeños no contemplados dentro del Convenio SOLAS, de manera que también puedan disponer de este elemento de seguridad en la navegación en un formato más simple y de menor coste.

Por otro lado, también existen receptores AIS que no transmiten su posición, pero reciben las difusiones de ambas clases de transpondedores.

En el modo normal de operación, un transmisor AIS difundirá su posición en un intervalo de 2 a 10 segundos, dependiendo de la velocidad que lleve el buque, y cada 3 minutos cuando el buque se encuentra fondeado y estacionario.

2.3 Trabajos relacionados

Desde hace años, los datos AIS han motivado numerosos estudios para la extracción de datos, a partir de grandes conjuntos, de aplicación en el entorno marítimo. En lo que respecta a extracción de rutas marítimas y detección de patrones de comportamiento y anomalías, los datos AIS son considerados como una fuente perfectamente válida pues, al fin y al cabo, representan el mayor porcentaje de datos transmitidos en la mar.

La mayoría de los trabajos publicados hasta la fecha pueden ser categorizados en función del tipo de algoritmo y método que utilizan, pudiendo diferenciar entre los que tratan el problema desde un enfoque de que divide el espacio en rejillas (*spatial grid techniques*) y los que usan métodos para representar el tráfico marítimo a partir de un conjunto de datos bruto (*spatio-temporal data mining techniques*). Cabe resaltar que el primer grupo se desenvuelve bien a la hora de analizar zonas marítimas de pequeño tamaño, pero cuando intenta trabajar a nivel global, encuentra dificultades a la hora de la computación. Además, no consigue diferenciar eficazmente aquellas anomalías estructuradas, como los puntos donde empieza/finaliza una navegación, por lo que su aplicación no sería interesante en este proyecto.

Ristic, La Scala, Morelande y Gordon son autores de un análisis estadístico de patrones de movimiento de los buques para extraer tendencias con los que construir detectores de anomalías con ayuda de métodos basados en densidades [26]. Pallota, Vespe y Bryan presentan una metodología denominada TREAD (*Traffic Route Extraction and Anomaly Detection*), que hace uso del algoritmo DBSCAN para detectar anomalías de formas automática y predecir futuros movimientos de las embarcaciones [27]. Zhang, Goerlandt, Kujala, Wang y Nikitakos hacen uso de algoritmos jerárquicos y otros métodos de agrupamiento para reconocer rutas de navegación en aguas de la Bahía de Xiamen y Chengsanjiao [28]. Por otro lado, Chang [29] investiga métodos para identificar lugares peligrosos por medio del análisis de datos AIS. El método que propone se fundamenta en algoritmos basados en densidad, e identifica cambios de rumbo para la determinación de rutas comparando los ángulos de caída con un umbral establecido. Este autor también es responsable de otras publicaciones y estudios, entre los que cabe mencionar el modelo que presenta para monitorizar barcos pesqueros haciendo uso de datos VMS, en vez de AIS [30]. En [27] se presenta un enfoque vectorial del problema, donde se suponen las trayectorias como un conjunto de nodos unidos por líneas, determinando rutas marítimas en función de su información espacio-temporal. Este trabajo dio pie a otros estudios en los que se proponen modificaciones al método, de cara a sintetizar y representar el tráfico marítimo. Entre otros, destaca el método que propone Chen [31] para extraer los canales principales en el oeste del estrecho de Taiwán, basado en esta metodología.

La ventaja principal de estos métodos es su capacidad para llevar a cabo el análisis del tráfico marítimo (ver Figura 2-11) a escala mundial y realizar predicciones. Sin embargo, aunque pueden detectar anomalías (Figura 2-12) en base a las rutas habituales de los barcos, la detección de anomalías debidas a la interacción entre barcos sigue siendo un reto.

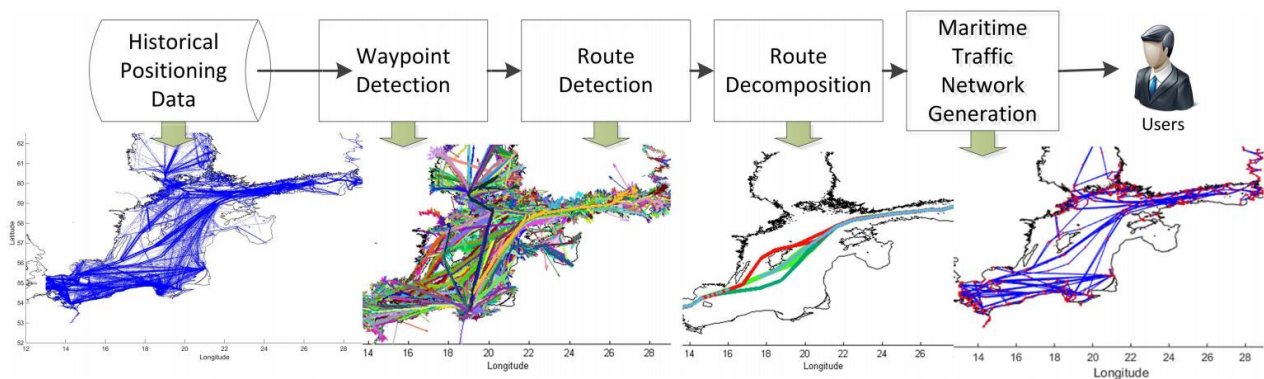


Figura 2-11 Maritime Traffic Knowledge Discovery and Representation System [32]

También cabe mencionar otras iniciativas internacionales con las que el desarrollo de este trabajo se alinea y guarda íntima relación. El *Centre for Maritime Research and Experimentation (CMRE)* de la *Science and Technology Organization (STO)* de la OTAN, en sus labores de investigación científica y desarrollo tecnológico centrados en el ámbito marítimo, trata de desarrollar herramientas que fusionan información heterogénea de numerosas fuentes para favorecer y atender las necesidades de la Alianza en materia de defensa y seguridad. En el documento publicado por la STO “*Highlights 2019*” [33], se destacan dos actividades para el análisis de datos masivos procedentes del sistema AIS, llevadas a cabo en el CMRE:

1. *Demonstration of maritime situational awareness pattern-of-life capabilities at the international maritime exercise 2019.*
2. *Multi-source reasoning with valuation networks for maritime anomaly detection – The Andromeda case*

Además, el CMRE organiza el “*Maritime Situational Awareness Workshop (MSAW)*” y el “*Maritime Big Data Workshop (MBDW)*”, convenciones que reúnen a investigadores, proveedores tecnológicos y miembros de la comunidad en aras de intercambiar su experiencia en materia de seguridad marítima y fomentar el esfuerzo común del dominio científico con el militar. Fruto de estas celebraciones es la publicación de diversos trabajos que proponen la utilización del *Big Data*, *Data Analytics* y *Machine Learning* para la detección de anomalías como los presentados por Mariano Alfonso Biscardi et al., en el que proponen una arquitectura de software en la nube para el manejo de grandes cantidades de datos en tiempo real [34], Johan Ali Mohamoud et al., enfocado a la mejora del MSA basado en el análisis de comportamientos [35], o Domonik Filipiak et al., el cual propone métodos para la detección de anomalías a partir de datos AIS [36].

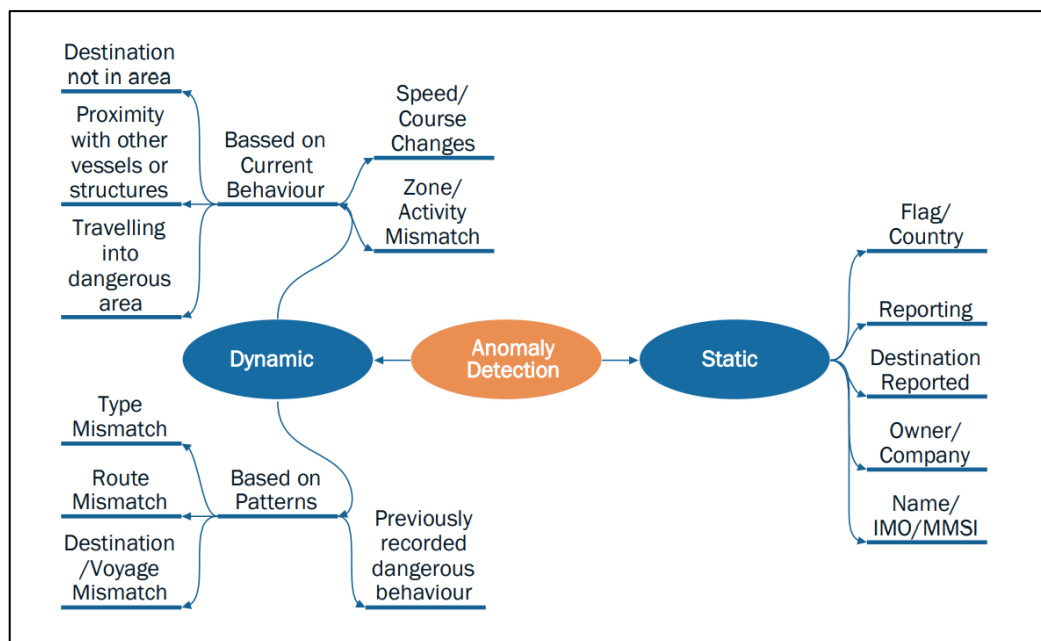


Figura 2-12 Maritime vessel anomaly as defined in the H2020 BigDataOcean project [37]

Como se puede comprobar a lo largo de esta sección, gran parte de estos trabajos, por no decir su totalidad, se basan en el uso de algoritmos de agrupamiento para procesar las cantidades ingentes de datos que se reciben del entorno marítimo en cada momento, con el objetivo de detectar patrones y anomalías en este medio. Es así como se justifica el interés que existe a nivel global en conseguir la capacidad de extraer de forma automática rutas habituales a partir un gran volumen de datos sin etiquetar que, de otra manera, no tendrían casi valor.

2.4 Análisis de Clusterización

A diferencia de las carreteras, las rutas marítimas de transporte no están perfectamente definidas, pues tanto su longitud como sus fronteras y contenido varía en espacio y tiempo por motivos diversos. A día de hoy, se tiene pequeño conocimiento sobre rutas las específicas que los buques siguen para llegar de un puerto a otro, lo cual supone información esencial para cálculos estadísticos e indicadores. Así como antiguamente no se poseía de mucha información de vigilancia marítima, hoy en día los sistemas de seguimiento automático generan sobreinformación. El principal reto que la seguridad marítima enfrenta hoy en día es la necesidad de encontrar patrones de comportamiento entre los buques que se encuentran navegando, de cara a detectar comportamientos anómalos que puedan estar relacionados con actividades ilícitas. Este tipo de desviaciones que se deben detectar se dividen principalmente en desviaciones estáticas (discordancias o inexistencia del número IMO o nombre del barco, por ejemplo) y desviaciones dinámicas, referidas a aquellas anomalías derivadas de una trayectoria inesperada.

En los próximos capítulos trataremos de utilizar algoritmos de agrupamiento para descubrir rutas marítimas habituales y comportamientos extraños.

2.4.1 Introducción al Análisis de Agrupamiento

El análisis de grupos o *cluster analysis* es el nombre genérico de una gran variedad de procedimientos que pueden ser utilizados para crear una clasificación. Es, concretamente, un procedimiento estadístico multivariante que parte de un conjunto de datos sobre una muestra de

entidades que trata de reorganizarlas en grupos relativamente homogéneos que se denominan clústeres. El objetivo de estos procedimientos es agrupar las muestras de manera que posean un alto grado de asociación entre aquellas del mismo grupo, pero bajo entre miembros de grupos diferentes.

Como se explica en [38], la mayor parte de la literatura sobre esta materia ha sido escrita durante los últimos 30 años. Si bien este término tuvo su primera aparición en el título de un artículo del año 1954 que trataba sobre datos antropológicos, el desarrollo de estos métodos estuvo motivado por el libro Principios de Taxonomía Numérica, publicado en 1963 por los biólogos Sokal y Sneath. Estos autores argumentaron que un método eficiente para realizar clasificaciones biológicas debía recoger todos los datos posibles sobre el conjunto de organismos de interés, estimar el grado en que estos eran similares y agrupar estos organismos en distintos grupos en función de su parecido. Es evidente que estos dos biólogos trataban de buscar un patrón para comprender el proceso evolutivo. De ahí en adelante, la literatura que versa sobre este concepto se desarrolla de forma considerable, principalmente debido al desarrollo de los ordenadores.

El agrupamiento tiene carácter exploratorio y constituye un proceso iterativo (si es automático) o interactivo (si interviene un usuario) basado en el ensayo y error. En la práctica, se ejecuta un algoritmo de agrupamiento y, a partir de los resultados que este devuelva, el operador modificará los parámetros y volverá a ejecutar el algoritmo con el objetivo de obtener resultados más satisfactorios. El comportamiento de la mayoría de los algoritmos de agrupamiento se ve altamente influenciado por la forma y la densidad de los grupos, así como por la distancia y la relación espacial entre ellas. Este aspecto geométrico puede ser muy variado, como se muestra en la Figura 2-13.

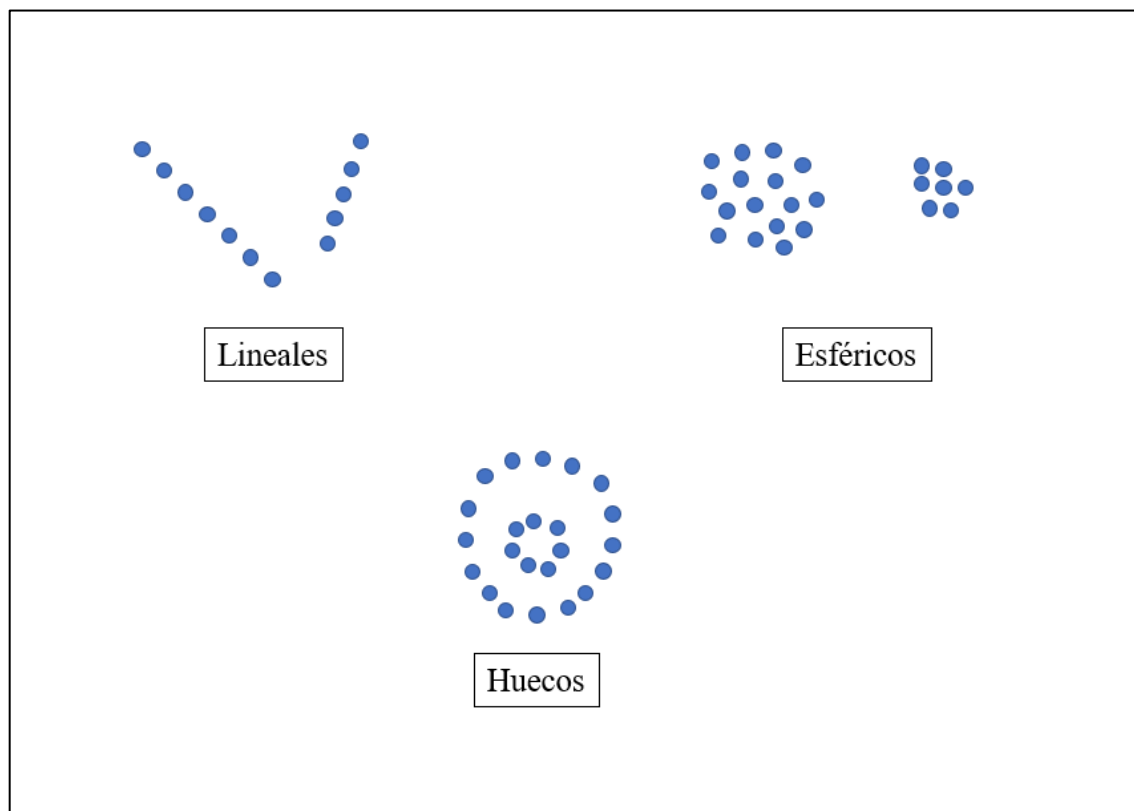


Figura 2-13 Ejemplos de grupos en R^2

2.4.2 Etapas del análisis de clusterización

En el empleo de una técnica de clusterización podemos diferenciar los siguientes puntos [38]:

1. Elección de variables

El investigador debe elegir, en base a sus objetivos, un conjunto concreto de características que quiere que se cumplan de cara a establecer un marco que sirva de referencia para la formación de las agrupaciones. En esta etapa se debe decidir no sólo el tipo de variable, sino también el número de variables; una mala elección puede suponer importantes repercusiones a nivel computacional y en la estructuración de los grupos.

2. Elección de la medida de asociación

Por lo general, los métodos de clusterización requieren que se establezca una medida de asociación para medir la proximidad entre los objetos. Cuando se realiza un análisis de grupo de individuos, esta proximidad se suele expresar en términos de distancias. Por otro lado, en el caso de análisis de variables, generalmente se utilizan coeficientes de correlación, de carácter más subjetivo, como los *coeficientes de Pearson* y de *Kendall*.

3. Elección de la técnica a emplear

Dentro de un mismo grupo se busca la máxima similitud entre los elementos (distancia intra-clúster pequeña), pero una distancia entre grupos lo más amplia posible (distancia inter-clúster grande). La naturaleza compleja y diversa del concepto de grupo y la forma de encontrarlos eficientemente ha dado lugar a multitud de algoritmos de agrupamiento diferentes, cuyo empleo vendrá determinado en función del problema a tratar y el tipo de datos que se vaya a utilizar en el estudio. En la práctica, es conveniente no cerrarse al empleo de un único algoritmo, sino abarcar un abanico de posibilidades y contrastar los resultados obtenidos, pues diferentes algoritmos ofrecen distintos resultados aplicados a un mismo conjunto de datos.

4. Validación de los resultados e interpretación de los mismos

Por último, como en todo estudio científico, es necesario valorar si el resultado se asume como válido o si, por otra parte, es necesario volver a empezar la secuencia (bien desde el principio o bien a partir de un punto). Si bien no debería seguirse de forma rigurosa, podría asumirse que, si algoritmos diferentes ofrecen resultados similares, estaremos ante un resultado probablemente válido. En esta etapa interviene con gran peso la valoración del que lleva a cabo el estudio a la hora de realizar una evaluación con criterio, para lo que existen procedimientos que puede emplear y técnicas que resultan útiles para tomar decisiones de forma más intuitiva.

2.4.3 Medidas de asociación

Para poder unir variables es necesario tener algunas medidas numéricas que caractericen las relaciones entre variables. Cada medida refleja asociación en un determinado sentido, por lo que es necesario escoger una medida apropiada para el problema que se vaya a tratar.

La medida de asociación puede ser una *distancia* o una *similitud*:

- Cuando se elige una *distancia* como medida de asociación, los grupos formados contendrán individuos parecidos de forma que la distancia entre ellos tiene que ser pequeña.
- Cuando se elige una medida de *similitud*, los grupos formados contendrán individuos entre los que exista una alta similitud.

Teniendo en cuenta el propósito de este trabajo, las medidas que nos interesan son aquellas empleadas para datos de escala intervállica:

- **Distancia de Minkowski**, se define como:

$$d(i, k) = \left(\sum_{j=1}^d |x_{ij} - x_{kj}|^r \right)^{\frac{1}{r}}, \text{ donde } r \geq 1$$

Asignando distintos valores a r , se obtienen las distancias Minkowski de uso más extendido (ver Figura 2-14):

- **Distancia Manhattan o “taxicab”**: cuando $r=1$

$$d(i, k) = \sum_{j=1}^d |x_{ij} - x_{kj}|$$

- **Distancia euclídea**: cuando $r=2$

$$d(i, k) = \left(\sum_{j=1}^d |x_{ij} - x_{kj}|^2 \right)^{\frac{1}{2}} = \sqrt{(x_i - x_k)^T (x_i - x_k)} = \|x_i - x_k\|$$

Es la distancia más corta entre dos objetos, y la más utilizada en técnicas de agrupamiento.

- **Distancia del supremo o distancia máxima**: cuando $r \rightarrow \infty$

$$d(i, k) = \max_{1 \leq j \leq d} |x_{ij} - x_{kj}|$$

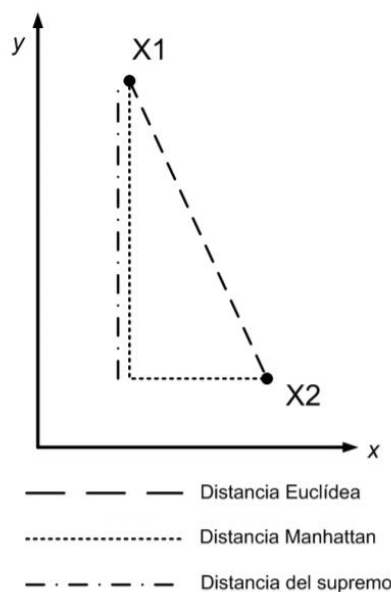


Figura 2-14 Distancias Minkowski [39]

2.4.4 Clasificación de los tipos de algoritmos

A partir de la información extraída fundamentalmente de [39], [40] y [41] sobre los distintos tipos de algoritmos de agrupamiento, se establecen dos tipos de clasificación principales para los algoritmos de agrupamiento: algoritmos particionales/jerárquicos y algoritmos de pertenencia exclusiva/no exclusiva, existen otras clases de algoritmos que también cabe reseñar (ver Figura 2-17).

2.4.4.1 Algoritmos particionales vs. Algoritmos jerárquicos

Los algoritmos particionales buscan dividir todo el espacio en K conjuntos o grupos de datos con el objetivo de optimizar un índice de coste, por lo que también se conocen como métodos de optimización. Esto implica que el usuario debe especificar el número de grupos que deben formarse, lo cual supone la principal diferencia con los métodos jerárquicos. Otra diferencia que existe es que estos algoritmos trabajan directamente con la matriz de datos original, por lo que no es preciso convertirla en una matriz de distancias o similitudes. Son de común aplicación en ingeniería, donde las particiones individuales son de interés. En la Figura 2-15 se exponen algunos ejemplos comunes.

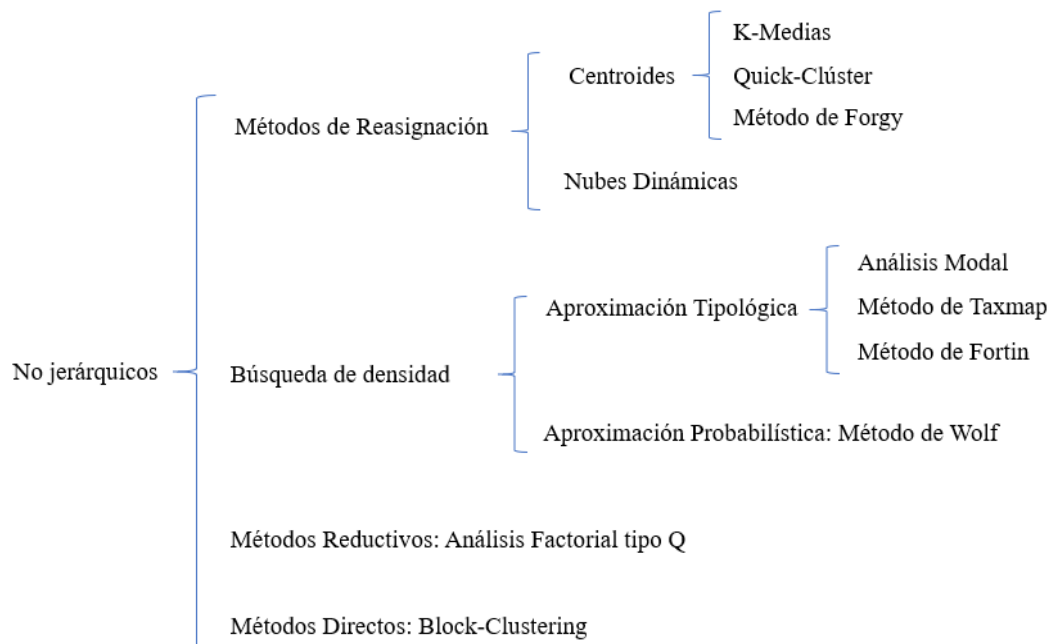


Figura 2-15 Algoritmos de clusterización no jerárquicos o particionales

Los algoritmos jerárquicos realizan una descomposición jerárquica de los objetos de forma que se minimice alguna función distancia o, en su defecto, que se maximice alguna medida de similitud. Existen dos tipos: los que parten de un único conjunto que engloba todas las observaciones, el cual se va ramificando en grupos más pequeños (llamados disociativos o *top-down*), y los que, considerando cada observación como un grupo independiente, los va fundiendo en clústeres más grandes en función de la distancia (aglomerativos o *bottom-up*). Estos algoritmos producen un gráfico de árbol o dendograma. En la Figura 2-16 podemos ver algunos de los algoritmos jerárquicos más conocidos. Este tipo de algoritmos suelen utilizarse en biología, ciencias sociales y ciencias de comportamiento por la necesidad de elaborar taxonomías.

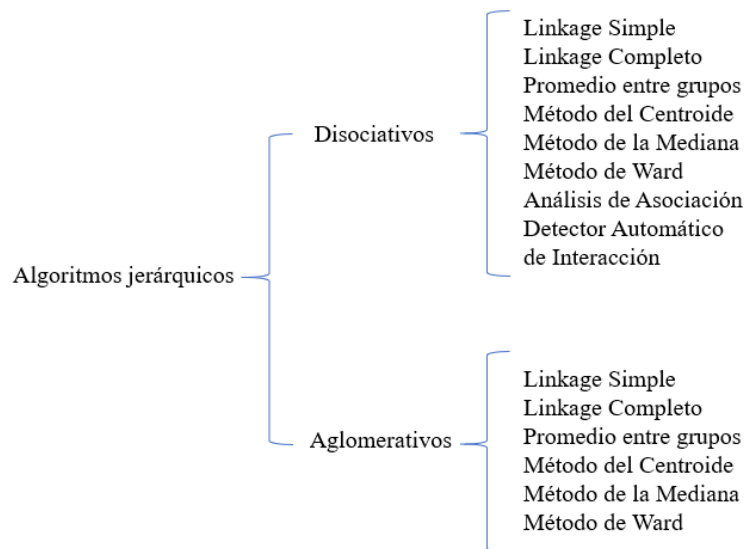


Figura 2-16 Algoritmos de clusterización jerárquica

2.4.4.2 Algoritmos de pertenencia exclusiva vs. no exclusiva

En los algoritmos de pertenencia exclusiva (*clustering* duro o *hard clustering*), cada objeto pertenece exactamente a un único grupo o clúster, al contrario que los de pertenencia no exclusiva (*clustering* blando o *soft clustering*), en los cuales los objetos pueden encontrarse de forma superpuesta en varios grupos. Los algoritmos de pertenencia no exclusiva más característicos son los algoritmos de agrupamiento borroso (*fuzzy clustering*), en donde cada objeto asignado tiene asignado un grado de pertenencia a todos los grupos detectados. Este grado de pertenencia es mayor cuanto mayor sea la similitud del objeto hacia el clúster.

Además de estos dos tipos de clasificaciones, cabe mencionar la existencia de otra clasificación que habla de clusterización parcial, cuya definición nos sirve para introducir el concepto de *outlier*: objetos que pueden aparecer sin pertenecer a ningún grupo diferenciado. Su complementario, la clusterización completa, se define como aquella en la que todos los objetos están asignados a un grupo particular.

2.4.4.3 Otras clasificaciones

Por la naturaleza del análisis de agrupamiento, sabemos que las distintas ramas de la ciencia e investigación han dado lugar a cientos de algoritmos. Es por ello que en la continuación se presentan otros algoritmos basados en técnicas alternativas a las que se han expuesto, los cuales también se valorarán para el posterior estudio de este trabajo.

Density-Based clustering. Agrupan objetos según un criterio de densidad, en vez de proximidad. Ejemplos de este tipo de algoritmos son DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), GDBSCAN (*Generalized Density-Based Spatial Clustering of Applications with Noise*) o DENCLUE (*DENSITY-Based CLUstEring*), OPTICS (*Ordering Points To Identify Clustering Structure*).

Grid-Based Clustering. Se divide el espacio en celdas y se buscan los objetos que pertenecen a cada celda. Existen dos tipos: STING (*STatistical Information Grid*) y CLIQUE (*CLUstEring in QUEst*).

Model-Based Clustering (Distribution-based clustering). Algoritmos que seleccionan una distribución estadística y se aplica para buscar los parámetros que mejor representan el comportamiento. Un ejemplo es el algoritmo de *Expectation-Maximization*.

Spatial Clustering. Se trata de algoritmos que agrupan los objetos por semejanza geográfica. Buscan agrupar los objetos en formas geométricas y detectar las relaciones espaciales entre ellas, lo que resulta en que algoritmos de otras clasificaciones también podrían considerarse dentro de ésta. Ejemplos son el GDBSCAN, GRAVClust o los métodos *grid-based*.

Distributed Data Clustering. Una de las soluciones a la extracción y obtención de datos en sistemas distribuidos es el *clustering* de datos, conocido como DDM (*Distributed Data Mining*). Los algoritmos de clusterización en sistemas distribuidos requieren de una alta sincronización, pues consiste en la agrupación de los datos para construir clústeres en áreas locales que envían sus resultados a una unidad central de proceso, donde se reestructura la información. Muchos de estos algoritmos incluyen técnicas de encriptación para transmitir la información de forma segura a través de redes donde la privacidad puede verse amenazada.

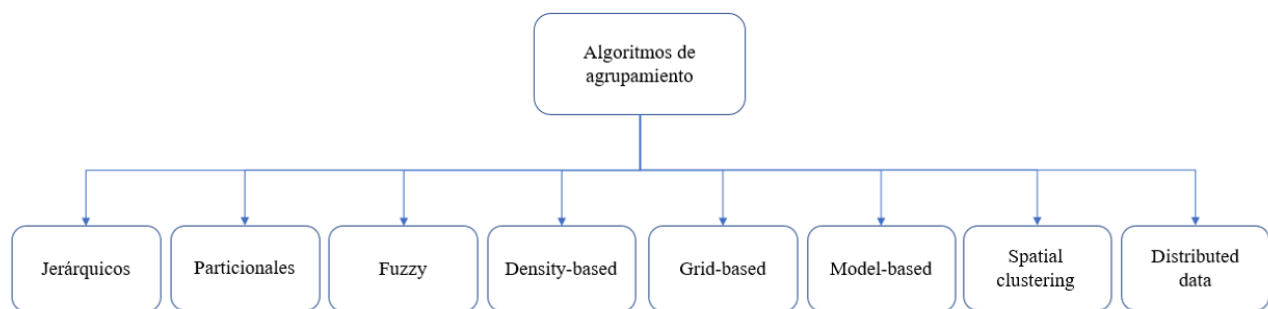


Figura 2-17 Tipos de algoritmos

2.5 Revisión y comparativa de algoritmos de los clusterización más destacados

Como se ha mencionado anteriormente, el algoritmo apropiado y sus parámetros dependen del conjunto de datos que se analiza y el uso que se le dará a los resultados. Es por ello que a continuación se describirán algoritmos que pueden resultar de interés para el propósito de este trabajo, y se compararán para su posterior aplicación en el desarrollo.

2.5.1 Descripción de los algoritmos de mayor interés para el proyecto

La base de datos con la que vamos a trabajar está focalizada principalmente en posiciones geográficas, si bien existen otros parámetros que se incluyen en los mensajes que la componen. Este factor motiva la descripción de varios de los algoritmos de clusterización particionales más destacados, así como de algunos algoritmos de agrupamiento alternativos, con la intención de hacer una selección y llevarlo a la práctica. La descripción de estos algoritmos está basada fundamentalmente en la información que se ofrece en [39], [42], [43] y [44].

Algoritmos de las distancias encadenadas o *chain-map*

Este algoritmo es uno de los algoritmos de agrupamiento más sencillos. Para un conjunto de objetos en forma de vectores, en donde cada valor del vector es una característica, el algoritmo selecciona al azar uno de estos objetos. A continuación, los objetos comienzan a relacionarse por proximidad, para lo que el algoritmo calcula todas las distancias euclídeas entre el objeto seleccionado y el anterior, ordenando así todos los objetos en un determinado orden.

Las distancias euclídeas pequeñas indican que los objetos deben pertenecer a un mismo clúster, mientras que las distancias grandes significarán saltos entre grupos. Para considerar una distancia como “grande” será necesario establecer un umbral que determine la distancia mínima (ver Figura 2-18).

A pesar de no resultar siempre óptimo, al proporcionar una agrupación en la que no es necesario establecer de forma previa el número de grupos deseado, este algoritmo sirve como paso previo para iniciar otros algoritmos de clusterización y para la estimación del número inicial de grupos a buscar.

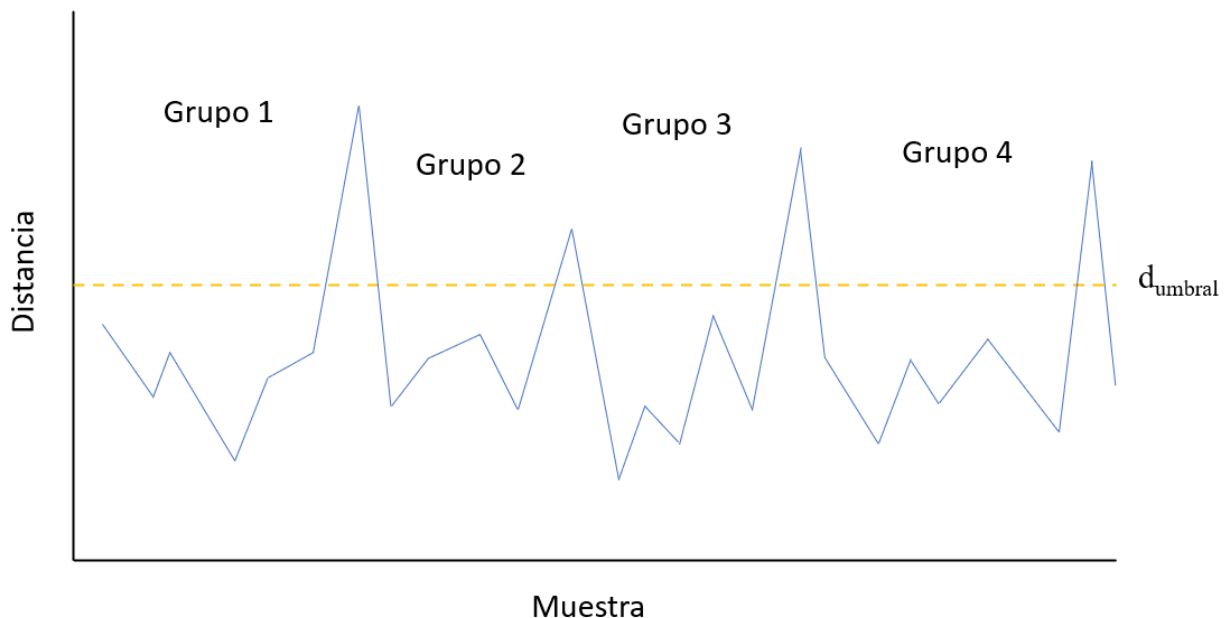


Figura 2-18 Gráfica explicativa algoritmo *Chain-map*

Algoritmo máx-min

Es un algoritmo heurístico que emplea como único elemento formal la distancia euclídea. Tampoco se necesita predefinir el número inicial de grupos a buscar, y su procedimiento es el siguiente.

1. Se selecciona un objeto al azar y se le asigna como patrón del grupo A_1 .
2. Se calculan todas las distancias euclídeas de todos los objetos restantes con respecto al objeto patrón del grupo A_1 , quedando el objeto con mayor distancia como patrón del grupo A_2 .
3. Se calculan todas las distancias euclídeas con respecto al nuevo objeto patrón. De las dos distancias obtenidas para cada objeto, se selecciona la más pequeña. De todo el conjunto de

distancias que se ha formado, se selecciona la mayor y se multiplica por un factor de ponderación f . Si la distancia resultante es mayor que la que separa a los grupos A_1 y A_2 , se crea un nuevo grupo A_3 .

$$d_{max} > f * d(z_1, z_2), 0 < f < 1$$

4. El procedimiento sigue creando nuevos grupos hasta que la distancia máxima mencionada sea menor que el valor medio de todas las distancias entre los patrones de los grupos creados.
5. Se recalculan por última vez todas las distancias euclídeas de los objetos restantes a todos los respectivos patrones de los grupos, asignando cada objeto al clúster más cercano.

Si bien no es necesario determinar de antemano el número de grupos que se desean, sí es necesario determinar el factor de ponderación (por ensayo y error), pues interviene de forma directa en el número de grupos que se crean.

K-means

Propuesto por Stuart Lloyd en 1957 (aunque fue actualizado más adelante por Hartigan y Wong, entre 1975 y 1978), este algoritmo es uno de los más conocidos y populares por su eficacia y robustez. Es un algoritmo heurístico de clasificación no supervisada que agrupa un conjunto de datos N en K grupos, basándose en sus características. El algoritmo se ejecuta minimizando la suma total de cuadrados intra-clúster (suma de las distancias al cuadrado de cada observación respecto a su centroide) y maximizando la distancia inter-clúster, dividiendo los datos en grupos homogéneos. Si se conoce el número de grupo con exactitud, este algoritmo resulta de gran sencillez, eficiencia y robustez. Sus pasos (ver Figura 2-19) son los siguientes:

1. Se escogen K elementos al azar que forman los grupos iniciales, siendo ellos los patrones o centroides (c_i) de los mismos.
2. Se asignan los objetos restantes al grupo cuya distancia euclídea sea menor.
3. Se recalcula el centroide de cada grupo como la media de los elementos que componen el mismo, buscando minimizar el índice:

$$J(M) = \sum_{i=1}^k \sum_{j=1}^{c_i} \|x_i - m_j\|^2$$

4. Los dos pasos anteriores se repiten cíclicamente hasta que todos los centroides permanezcan constantes o se cumpla cualquier otra condición de finalización. Aunque no se garantiza una solución óptima, el algoritmo siempre finaliza (converge).

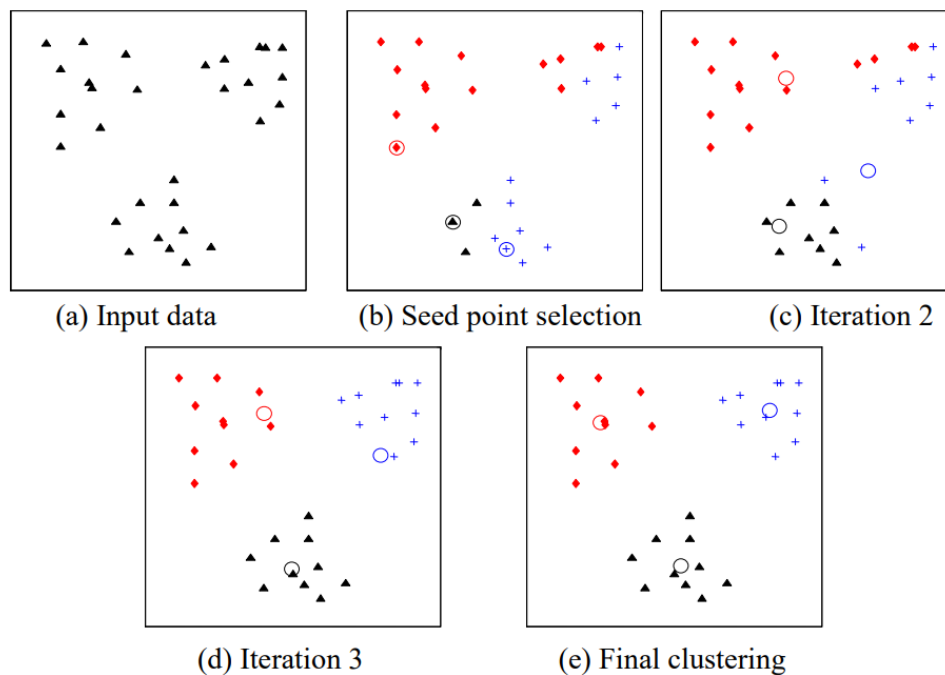


Figura 2-19 Algoritmo K-means [41]

El resultado de ejecutar este algoritmo es la creación de una partición de observaciones que siguen el esquema de Voronoi (ver Figura 2-20) donde se supone igual tamaño para los grupos.

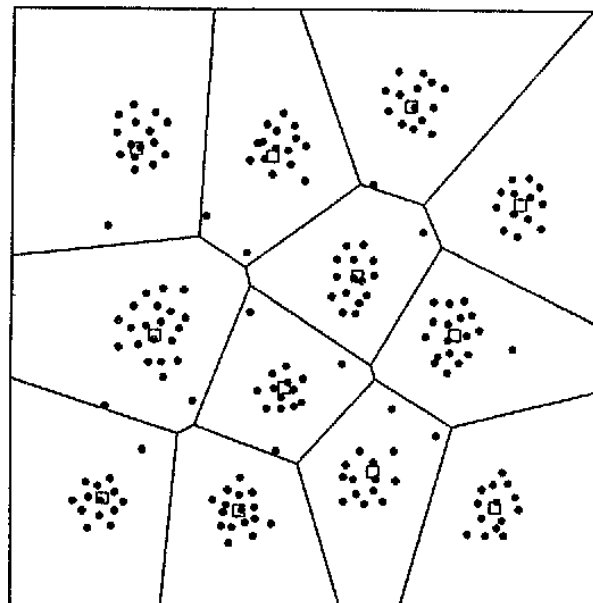


Figura 2-20 Diagrama de Voronoi con 12 clústeres [45]

El algoritmo es muy sensible al valor de K, el cual debe ser determinado de antemano. Por lo general, un valor elevado de K hace que el error disminuya, pero, a cambio, se ve mermada la información que la agrupación resultante ofrece. Como no existe un método analítico para determinar este valor, pues depende del conjunto de datos que vayamos a procesar, se suele hacer una aproximación por ensayo y error.

PAM (*Partitioning Around Medoids*)

Es una extensión del algoritmo K-means, en donde cada grupo se representa por un metoide (ver Figura 2-21), en lugar de por un centroide. Una definición de este término es: el elemento del clúster cuya distancia promedio entre él y todos los demás elementos del mismo clúster es la menor posible, y se corresponde con el elemento más central del grupo. Puede ser cercano al centroide, pero no siempre es así por no estar siempre el objeto patrón en el centro del clúster.

El hecho de utilizar metoides hace que el algoritmo resulte más robusto, viéndose menos afectado por outliers. Por lo general, el método de K-medoids se utiliza cuando se conoce o se sospecha que existen outliers y, cuando esto ocurre, es recomendable utilizar como medida de similitud la distancia Manhattan, ya que es menos sensible a los outliers que la euclídea [46].

El procedimiento que sigue este algoritmo es similar al del K-means.

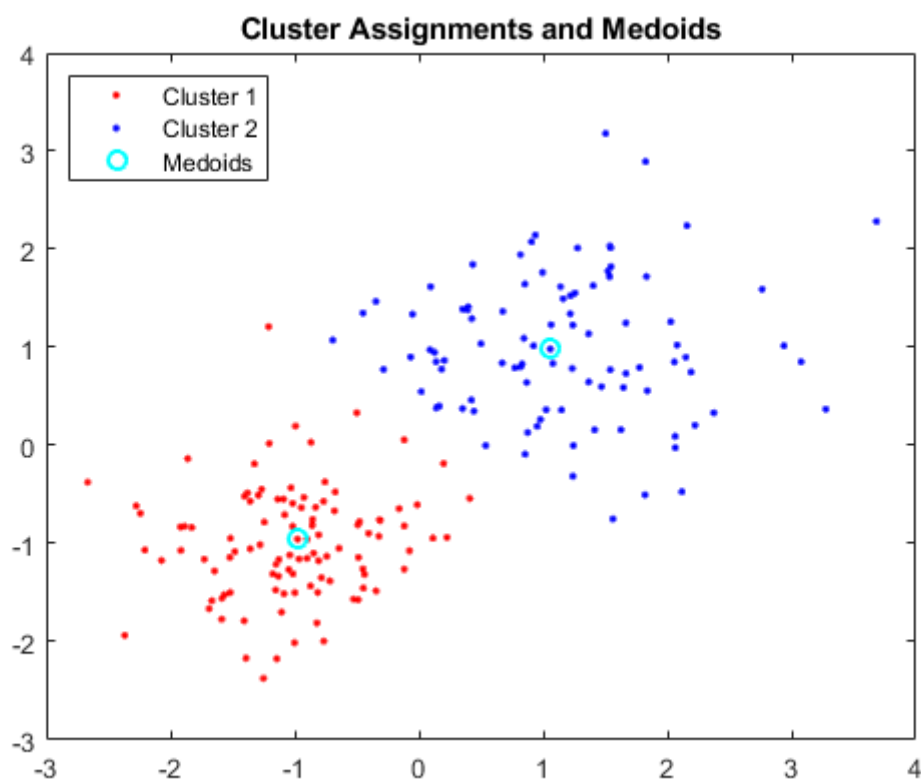


Figura 2-21 K-medoids clustering output [47]

CLARA (*Clustering Large Applications*)

Una de las limitaciones del método K-medoids es que su algoritmo requiere mucha memoria RAM, lo que se traduce en un retardo notable cuando el dataset contiene varios miles de observaciones. El algoritmo CLARA divide la base de datos original en distintas muestras de tamaño S y se ejecuta PAM sobre cada una de ellas, seleccionando la mejor clasificación de las resultantes.

La motivación de este algoritmo es aligerar la carga computacional que supondría procesar grandes bases de datos, pero sacrifica optimalidad y precisión.

DBSCAN (*Density Based Spatial Clustering of Applications with Noise*)

Este algoritmo está basado en la densidad de los grupos formados. Por el hecho de buscar formas densas que pueden ocupar zonas geográficas, también se le clasifica como un algoritmo de clusterización geográfico.

Similar al agrupamiento basado en conectividad, este algoritmo también conecta puntos (los cuales pueden formar un grupo de forma arbitraria), que cumplen un umbral de distancia en este caso, puntos que satisfacen un criterio de densidad (*density-reachable*), dejando como outliers los puntos que no lo rebasan (ver Figura 2-22). El valor de densidad viene definido por la relación entre el número de objetos y el área que ocupan.

Un grupo está formado por un conjunto de objetos densamente conectados más todos los objetos que están dentro del rango de estos. El algoritmo requiere dos parámetros:

- Un valor ε (*eps*) que indica la distancia máxima en la que se buscan los vecinos de un elemento.
- Un número mínimo de puntos vecinos entre sí (MinPts) para que estos se agrupen. Si hay sospecha de mucho ruido en la base de datos, un número elevado favorecerá la creación de grupos menos influenciados por los *outliers*.

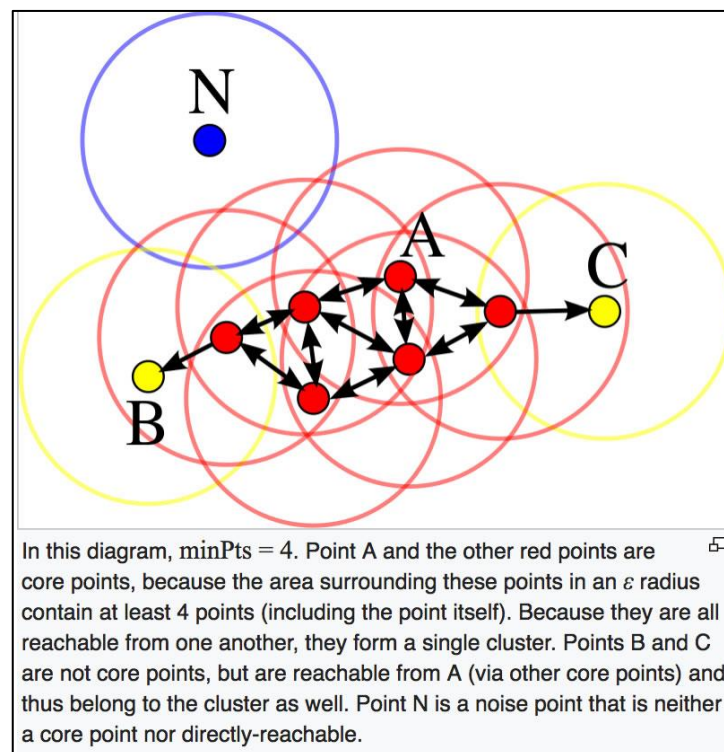


Figura 2-22 Funcionamiento del algoritmo DBSCAN [48]

Estos parámetros permiten la flexibilidad del concepto de densidad que maneja el algoritmo, pero dificultan su uso porque deben ajustarse. Además, presenta la ventaja de no tener que establecer el número de clústeres a priori y permite encontrar grupos no linealmente separables, como se puede ver en la Figura 2-23), computando en un tiempo $O(n \log(n))$.

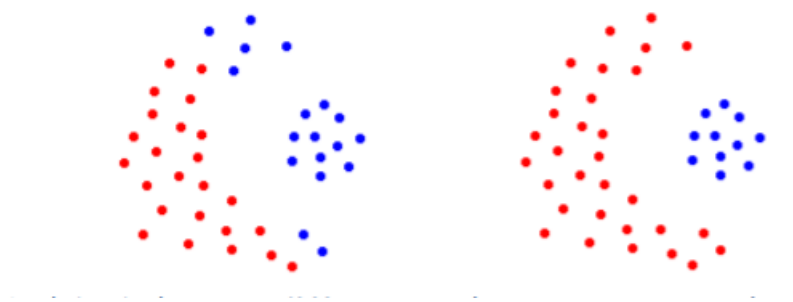


Figura 2-23 K-Means (izquierda) no es capaz de separar grupos no linealmente separables, mientras DBSCAN (derecha) si es lo es [49]

Este algoritmo es robusto frente a datos espurios al no asociarlos a ningún grupo, por lo que es evidente que resulta de gran utilidad en el contexto geoespacial en el que nos moveremos en este proyecto. Por otra parte, no es totalmente determinístico (los *border points* alcanzables pueden asignarse a cualquier grupo alcanzable), ni genera buenos resultados cuando la densidad de los distintos grupos es muy distinta.

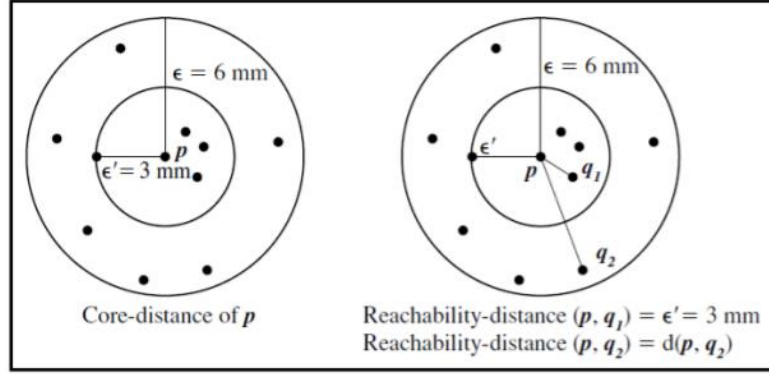
Similar a este algoritmo, podemos encontrar **HDBSCAN** (*Hierarchical DBSCAN*), que se trata básicamente de una extensión de DBSCAN para un rango de valores ϵ (*eps*), por lo que solamente necesita el tamaño mínimo de clúster como input. La ventaja que ofrece es que es capaz de detectar grupos con densidades diferentes sin tener que establecer un umbral de distancia previamente.

OPTICS (*Ordering Points To Identify Cluster Structure*)

Para resolver el complejo problema de seleccionar los parámetros adecuados (*inputs*) para el algoritmo DBSCAN, nace una extensión bajo el acrónimo de OPTICS, el cual produce un orden de agrupamiento (*cluster ordering*) obtenido a partir de un conjunto de parámetros, en un análisis automático e iterativo. Esta ordenación representa una estructura de agrupamiento basada en densidad de los datos, y equivale a obtener las agrupaciones para un amplio rango de valores de los parámetros de DBSCAN.

La idea de este concepto se basa en procesar los objetos en el orden más adecuado y mantener un registro de la densidad de sus puntos vecinos, para lo cual se consideran dos distancias (Figura 2-24):

- Distancia-núcleo (*core distance*) de un objeto p es el valor ϵ (*eps*) más pequeño que hace que sea alcanzable. Para objetos que no son núcleos, esta distancia no está definida.
- Distancia-alcanzable (*reachability-distance*) de un objeto q con respecto a otro es la mayor entre la distancia-núcleo de p y la distancia euclídea entre p y q .



Ejemplo: $\epsilon = 6 \text{ mm}$; $m = \text{MinPts} = 5$

Figura 2-24 Distancias para ordenación en OPTICS [50]

DENCLUE (*Density-based CLUstEring*)

Otra aproximación a los algoritmos basados en densidad es buscar una función de densidad. DENCLUE [51] usa el concepto de las *funciones de influencia* para catalogar la influencia que cada objeto ejerce sobre los elementos cercanos. Su funcionamiento se asemeja a las redes neuronales, pues las funciones cambiarán de estado (se activarán) cuando se supere cierto umbral. Así, imponiendo una densidad mínima g_{min} , el espacio que cumpla la condición $g(x) > g_{min}$ será un clúster.

Esta función de densidad viene dada por la suma de las funciones de influencia de todos los puntos, y dependerá de la naturaleza de los datos que se estén utilizando:

$$g(x) = \sum_{i=1}^n g_i(x)$$

El algoritmo DENCLUE trabaja en dos pasos:

1. Se construye un mapa por métodos *grid-based* para determinar que celdas están ocupadas, limitando así el espacio de búsqueda a las regiones donde existan puntos y acelerar el cálculo de la función de influencia.
2. Es el paso del agrupamiento como tal. Partiendo de una rejilla, se busca el máximo local de la función densidad y se identifican todos los puntos de la región para asociarlos a este máximo por medio del procedimiento *hill-climbing* de forma iterativa.

Las ventajas que presenta este algoritmo es que funciona muy bien con grandes conjuntos de datos que contengan mucho ruido (diferenciando eficientemente los *outliers*), permite detectar clústeres de formas arbitrarias y es notablemente más rápido que otros algoritmos, como DBSCAN. Su complejidad computacional es $O(n \log)$.

Como se explica en [52], el problema que presenta DENCLUE es que el procedimiento *hill-climbing* puede dar lugar a pequeños pasos innecesarios que se traducen en pérdida de tiempo, y nunca converge al máximo exactamente, sino que simplemente se acerca mucho. Para solventar este problema y mejorar el algoritmo, se desarrolla una nueva versión (DENCLUE 2.0) que acelera el proceso por medio de un *hill-climbing* mejorado (ver Figura 2-25), que ajusta los pasos con un coste despreciable.

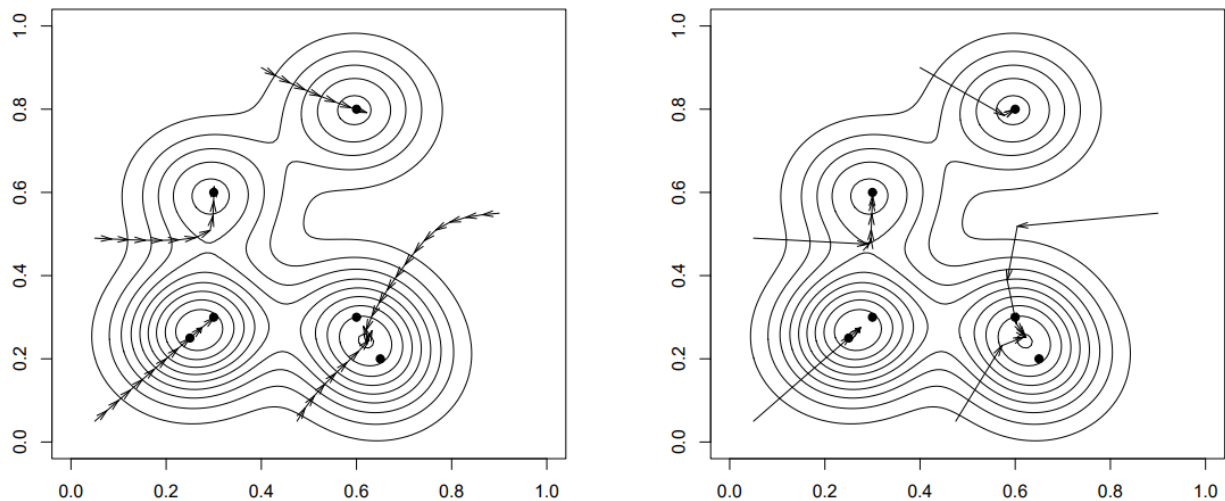


Figura 2-25 DENCLUE *hill-climbing* (izquierda) frente a DENCLUE 2.0 [52]

STING (*Statistical Information Grid*)

El algoritmo STING [53] hace uso de una estructura de rejilla jerarquizada para reducir al máximo el tiempo de cada búsqueda. Se parte de una raíz que ocupa todo el espacio de datos y se divide en un número finito de celdas de igual tamaño (ver Figura 2-26). Los autores recomiendan dividir una celda de dos dimensiones en cuatro regiones iguales, por lo que para un espacio tridimensional parece intuitivo dividirlo en ocho partes iguales. Todas las subdivisiones se organizan en un árbol semejante al que produce el algoritmo BIRCH, y cuyo nivel más bajo, al dividirse, proporcionaría una arista menor que la distancia característica de nuestro problema b . Una vez ha finalizado la partición del espacio, habiendo alcanzado el nivel de detalle deseado, se forman los clústeres mediante la asociación de celdas que contienen información similar.

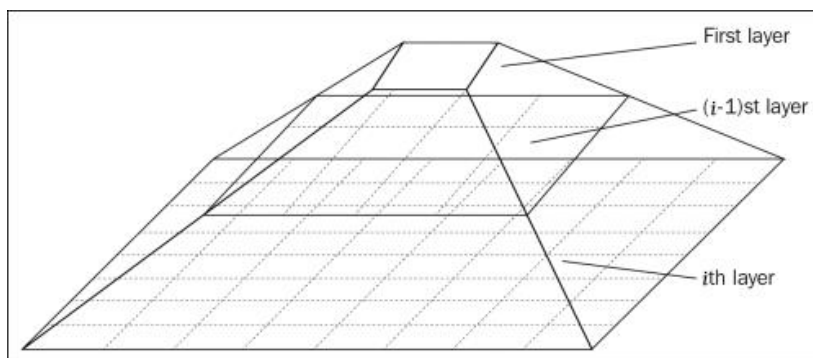


Figura 2-26 Algoritmo STING: división de la rejilla [54]

La diferencia de funcionamiento entre DBSCAN y STING radica en que el primero agrupa los datos en base a posiciones espaciales, centrándose en dos parámetros (eps y $MinPts$), mientras que, en el segundo, la agrupación sólo depende de un parámetro: la densidad. Cuando se ajusta este parámetro de manera que la granularidad tienda a cero, el comportamiento entre ambos será equivalente.

Para conseguir agrupaciones correctas, sin bordes de sierra, es conveniente la aplicación de otro algoritmo previamente. Sin embargo, este algoritmo puede servir, a su vez, como punto de partida para

otros algoritmos distintos, al ofrecer una división en clústeres bastante correcta en un tiempo verdaderamente reducido.

Para obtener puntos a una mayor velocidad, cada celda ofrece información acerca de los puntos que contiene. Si bien generar la estructura mallada implica un determinado coste, aplicando esta metodología se consigue reducir considerablemente el tiempo de búsqueda de los puntos. En el estudio realizado por los profesores Wang, Yang y Muntz [53] se compara el algoritmo STING con los conocidos algoritmos jerárquicos CLARANS (versión más eficiente de CLARA) y BIRCH, y el algoritmo DBSCAN, para analizar su rendimiento a la hora de clusterizar (ver Tabla 2-1). STING tiene tiempo de generación (*generation time*) y tiempo de consulta (*query time*).

Nº puntos	1256	2503	3910	5213	6256	12512
CLARANS	49 s	200 s	457 s	785 s	1238 s	5538 s
DBSCAN (projected)	0,2 s	0,4 s	0,7 s	1,0 s	1,2 s	2,86 s
STING (query)	0,1 s	0,11 s	0,11 s	0,12 s	0,12 s	0,14 s
STING (generation)	1,25 s	1,32 s	1,40 s	1,48 s	1,55 s	1,62 s

Tabla 2-1 Test de rendimiento para CLARANS; DBSCAN y STING (Basado en [53])

Sabiendo que BIRCH ofrece un rendimiento 20-30 veces superior a CLARANS, se deduce que el algoritmo STING también rendirá mejor que este último por un amplio margen. Su complejidad computacional es $O(n)$.

CLIQUE (*Clustering In QUES*)

CLIQUE también realiza particiones del espacio según niveles, pero cada nivel se conforma como una dimensión o subespacio más. La estructura en la que se particiona se denomina hiperrectángulo (ver Figura 2-27), que es la generalización de un rectángulo para dimensiones superiores. El funcionamiento sigue el siguiente esquema:

1. Comienza con una dimensión única y la divide en secciones, buscando aquellas de mayor densidad.
2. A continuación, incluye la segunda dimensión en el análisis, particionando el espacio en rectángulos y buscando los más densos.
3. Continúa sucesivamente añadiendo dimensiones y repitiendo el proceso.
4. Al terminar con todas las características o dimensiones de los objetos, se definen los grupos y sus relaciones mediante semejanza de densidades y otra información que se extrae, en todos los niveles.

Respecto a su eficiencia, este algoritmo puede trabajar con grandes *datasets*, no le afecta el orden de las variables que se deben introducir, y computa clústeres en un tiempo $O(n)$.

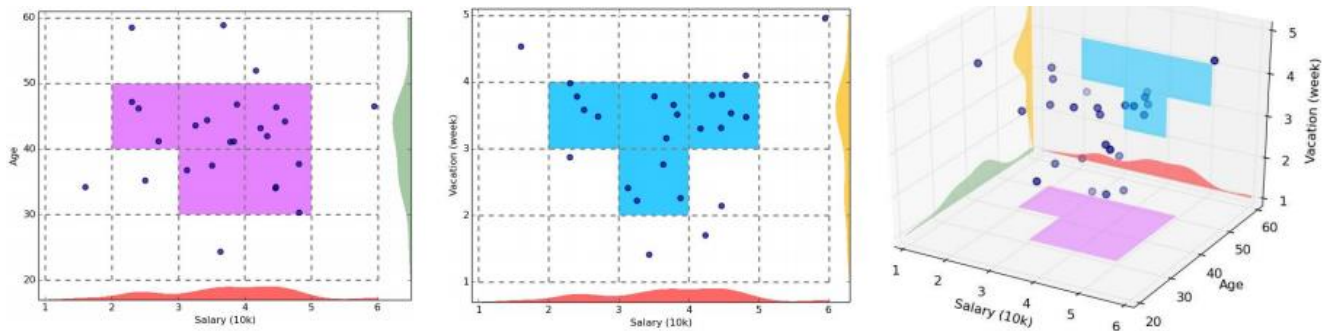


Figura 2-27 Ejemplo de algoritmo CLIQUE [55]

2.5.2 Comparativa de los algoritmos presentados

Como ya se había mencionado anteriormente, no hay una respuesta clara para determinar qué algoritmo es más eficaz o más eficiente. Un método específico puede dar buenos resultados para un conjunto de datos concreto y, sin embargo, hacerlo de forma pobre en otro conjunto distinto en función del tamaño y dimensiones de los datos, el objetivo y la estructura. A pesar de todo, existe consenso acerca de las características que todo algoritmo de clusterización debería intentar satisfacer [56]:

- Escalabilidad: La capacidad del algoritmo de trabajar correctamente con conjuntos de datos de gran tamaño.
- Análisis de mezclas de atributos: capacidad de analizar atributos individuales, así como mezclas de atributos.
- Encontrar grupos con forma arbitraria: lo ideal es que un algoritmo sea lo más genérico posible, siendo capaz de detectar clústeres de cualquier forma.
- Mínimo número de parámetros a introducir: muchos algoritmos requieren que el usuario introduzca parámetros predefinidos para el funcionamiento del algoritmo como, por ejemplo, el número de grupos. Sin embargo, es deseable en grandes conjuntos de datos requieran el mínimo número de *inputs* de cara a evitar que se encuentren resultados sesgados.
- Soportar ruido: los algoritmos de clusterización deben ser capaces de mostrarse insensibles ante la presencia de desviaciones, para ofrecer resultados de calidad. La detección de anomalías es un problema diferente.
- Insensibilidad al orden de los inputs: hay muchos algoritmos que ofrecen resultados totalmente diferentes cuando el *dataset* es introducido en órdenes distintos. Esto es indeseable, por lo que un buen algoritmo debe ser insensible al orden en el que los datos se le presentan.

- Manejo de grandes datasets: la creciente velocidad en la que hoy en día se generan los datos obliga a los algoritmos a que sean capaces de manejar grandes cantidades sin que su rendimiento y calidad se vean muy afectados.
- Interpretabilidad y usabilidad: es deseable que los resultados sean fácilmente entendibles para facilitar las tareas de comparación posteriores.

Por tanto, presentadas estas exigencias y los algoritmos más interesantes para este proyecto, se ha elaborado la Tabla 2-2 para facilitar la comparación.

Algoritmo	Inputs	Optimizado para...	Estructura del clúster	Distingue outliers
PARTICIONALES				
K-means	Nº clústeres	Grupos separados	Esférica	No
PAM	Nº clústeres	Grupos separados, dataset pequeño	Esférica	No
CLARA	Nº clústeres	Dataset relativamente grande	Esférica	No
DENSITY-BASED				
DBSCAN	ϵ , minPts	Grupos arbitrarios, dataset grande	Arbitraria	Sí
HDBSCAN	min_cluster_size, min_samples	Grupos arbitrarios, dataset grande	Arbitraria	Sí
OPTICS	minPts	Grupos arbitrarios, dataset grande	Arbitraria	Sí
DENCLUE	ϵ , minPts	Grupos arbitrarios, dataset grande	Arbitraria	Sí
GRID-BASED				
STING	Nº celdas en nivel inferior, nº de objetos en celda	Dataset espacial grande	Límites verticales y horizontales	Sí
CLIQUE	Tamaño rejilla, nº mínimo de puntos entre celdas	Dataset grande de alto nº de dimensiones	Arbitraria	Sí

Tabla 2-2 Tabla comparativa de los algoritmos más destacados

Tras comparar los resultados recogidos a lo largo de esta sección, se considera que los algoritmos más adecuados para la realización de este proyecto son aquellos basados en densidad. En concreto se utilizarán los algoritmos DBSCAN y HDBSCAN, por haber sido empleados en otros trabajos y existir implementaciones para los mismos. Esto se debe a varios motivos fundamentales:

- Son capaces de discriminar *outliers*, lo que resulta de gran interés por permitir la posible detección de comportamientos anómalos entre los barcos que componen la base de datos. Teniendo en cuenta el tipo de datos al que nos enfrentamos, es altamente probable que el *database* contenga ruido por no constituir una fuente de información fiable en todos sus campos.
- El número de parámetros que hace falta configurar para su correcta implementación es pequeño, lo que supone una ventaja a la hora de configurarlos.
- Su escalabilidad permite trabajar adecuadamente con grandes bases de datos.

- Es capaz de detectar zonas agrupamientos de formas arbitrarias, lo que facilitará la detección de puertos y zonas de fondeo sin atender a su geometría.

2.6 Comparativa y selección del lenguaje

2.6.1 Python vs R

A la hora de afrontar el proyecto, la primera pregunta que se plantea es dónde y cómo se va a trabajar. Para ello se presentan a continuación dos de los lenguajes más utilizados en *Data Science*:

- **R:** Es un entorno de software libre (licencia GNU GLP) y lenguaje de programación interpretado, es decir, ejecuta las instrucciones a medida que se introducen, sin necesidad de compilar el código de forma previa, lo cual también facilita la detección de errores. Fue creado en 1995 por Ross Ihaka y Robert Gentleman para suceder al lenguaje S y mejorar sus prestaciones de cara a la visualización y análisis de datos (ver Figura 2-28). Fue concebido inicialmente para su uso en computación estadística, por lo que una parte importante de los usuarios que hacen uso de este lenguaje pertenecen al mundo de las matemáticas.



Figura 2-28 Logotipo R [57]

- **Python:** Es un lenguaje de programación interpretado y multiparadigma de código abierto (*Licencia Python Software Foundation License*), creado a finales de los años 80 por Guido van Rossum para sustituir al lenguaje de programación ABC (ver Figura 2-29). Por ser multiparadigma, permite distintos estilos de programación: programación orientada a objetos, programación imperativa y programación funcional. Es conocido por ser un lenguaje de programación ágil y sencillo, con una curva de aprendizaje muy corta, lo que le ha llevado, tal como se especifica en [58], a ser el número uno en cuanto a lenguaje más deseado para programar por aquellos que no lo están usando todavía.



Figura 2-29 Logotipo Python [59]

Ambos lenguajes tienen cierta madurez y se encuentran en constante evolución. Detrás de ellos se esconde una gran comunidad de desarrolladores sin ánimo de lucro que comparten sus trabajos y proyectos para dar apoyo a las diferentes páginas de desarrollo colaborativo que dan soporte a estos entornos. Como ejemplo se puede mencionar GitHub que, para el caso de R, cuenta con más de 43000 repositorios, y para Python, más de 91000.

Como se afirma en [60], la ciencia de los datos se ha convertido en uno de los campos de mayor desarrollo por parte de las empresas. Dentro del uso de los datos, toma gran importancia la visualización de información compleja para extraer conclusiones y mejorar la toma de decisiones. Tanto Python como R cuentan con varios paquetes o *plugins* enfocados en la visualización de los datos. En el caso de R, dos de los más usados son *ggplot2* y *rgl*. Por otro lado, en Python también se encuentran disponibles varios paquetes para hacer representaciones de datos, como son *matplotlib*, para gráficos de todo tipo con muy poco código, o *Seaborn*, basada en la anterior. Relacionado con esto último, y siendo de especial relevancia para el éxito de este trabajo, ambos cuentan con librerías para la representación gráfica de mapas geográficos. Si bien R es un lenguaje poderoso que sobresale en aplicaciones de visualización de datos, los *plugins* de Python están recortando paulatinamente la distancia que les separa. Además de estos *plugins*, ambos lenguajes poseen librerías para implementar algoritmos de *clustering*, lo que resulta imprescindible para la realización de este proyecto.

Si bien ambos lenguajes nos resultan similares en cuanto a aplicabilidad, presentan también algunas diferencias. En primer lugar, una realidad reconocida por casi todos es que R es un lenguaje de programación más lento que Python, ya que requiere que sus objetos se almacenen en una memoria física. Esto hace que, para tratar con grandes volúmenes de datos, Python resulte más adecuado si para nuestro estudio la velocidad se considera un factor importante. Además, si anteriormente no se ha trabajado con estos lenguajes, será sencillo acostumbrarse a Python, pues su sencillez permite que todo tipo de usuarios, sin necesidad de ser expertos, puedan trabajar cómodamente.

Así, de cara a tomar una decisión sobre qué lenguaje utilizar y teniendo en cuenta el objetivo y tipo del proyecto, se ha elaborado una tabla comparativa (Tabla 2-3) con los aspectos que se consideran mayor interés y relevancia:

	<i>Open Source</i>	Instalación en Windows	Rapidez	Visualización gráfica de los resultados	Adecuado para Big Data	Librerías de algoritmos	Amplio dominio	Comunidad en línea
Python								
R								

Tabla 2-3 Python vs R

La columna “amplio dominio” se refiere a la variedad de ámbitos de aplicación del lenguaje ya que, aunque no sea algo vinculante con el éxito del proyecto, se considera de importancia en la parte formativa por resultar de mayor utilidad para el futuro.

Por tanto, teniendo en cuenta lo reseñado en este apartado y, aunque ambos lenguajes pudieran resultar efectivos a la hora de encontrar los resultados deseados, se seleccionó Python para abordar este proyecto.

3 DESARROLLO DEL TFG

En el siguiente capítulo se describirá la implementación de los algoritmos seleccionados para afrontar el proyecto. En primer lugar, se expone como se llevó a cabo la instalación de las herramientas precisas para la realización del mismo. Una vez establecido el entorno de trabajo, se procederá a realizar pruebas con los algoritmos y se irán ajustando los parámetros con el objetivo de analizar los resultados que nos ofrecen.

3.1 Instalación del software

Tras la decisión de utilizar Python como lenguaje para el desarrollo del trabajo, el siguiente paso fue proceder a la instalación del mismo y de todas las herramientas necesarias para la consecución de los objetivos.

3.1.1 Anaconda®

Anaconda® es una distribución libre y abierta de los lenguajes Python y R, utilizada principalmente para *Data Science* y aprendizaje automático (ML). Se trata de un gestor de paquetes y entornos, y posee una colección de más de 7500 paquetes *open source*, considerada por muchos como la suite más completa para la ciencia de datos con Python.

Anaconda Distribution se agrupa en 4 soluciones tecnológicas que se pueden ver en la Figura 3-1, y que se instalarán de forma automática cuando llevemos a cabo el proceso de instalación.

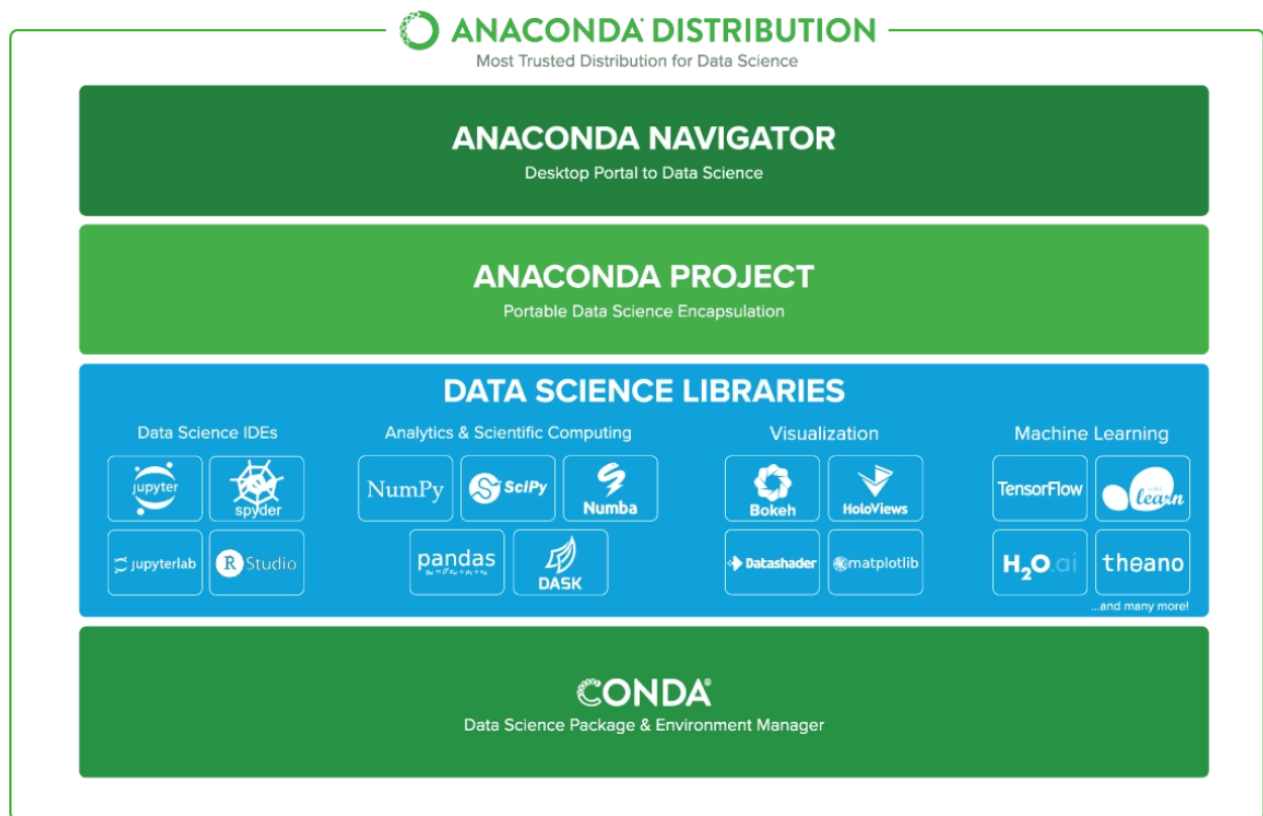


Figura 3-1 Distribución Anaconda [61]

De entre las numerosas características con las que cuenta esta suite, son las más reseñables:

- Libre, de código abierto, con un soporte comunitario muy amplio y documentación bien detallada.
- Multiplataforma (Linux, macOS y Windows)
- Permite instalar y administrar paquetes, dependencias y entornos con diferentes versiones de Python.
- Cuenta con un elevado número de herramientas para la ciencia de datos y visualización.
- Los proyectos son portables.
- Ofrece una gran variedad de aplicaciones relacionadas con el aprendizaje de máquina y modelos de aprendizaje, además de facilitar la escritura de complejos algoritmos paralelos para la ejecución de tareas.
- Cuenta con una interfaz gráfica de usuario GUI muy sencilla e intuitiva (Anaconda Navigator), y brinda la posibilidad de acceder a recursos de aprendizaje avanzados.

Para el desarrollo del presente proyecto se procedió a la instalación de la versión gratuita de la suite en su opción para 64-bits y Python 3.8, “Anaconda Individual Edition”, la cual viene con más de 250 paquetes instalados de forma predeterminada.

Para ello, se visitó la página web oficial de Anaconda, accediendo después a la pestaña “*Products > Individual Edition*”. Al igual que con Python, se siguieron los pasos marcados por un intuitivo asistente de instalación.

Una vez finalizada, se crea un acceso directo en el escritorio para lanzar “Anaconda Navigator”, interfaz a la que se accede durante todo el proyecto, para utilizar las funciones que ofrece la distribución de manera más intuitiva, y que se presenta en la Figura 3-2.

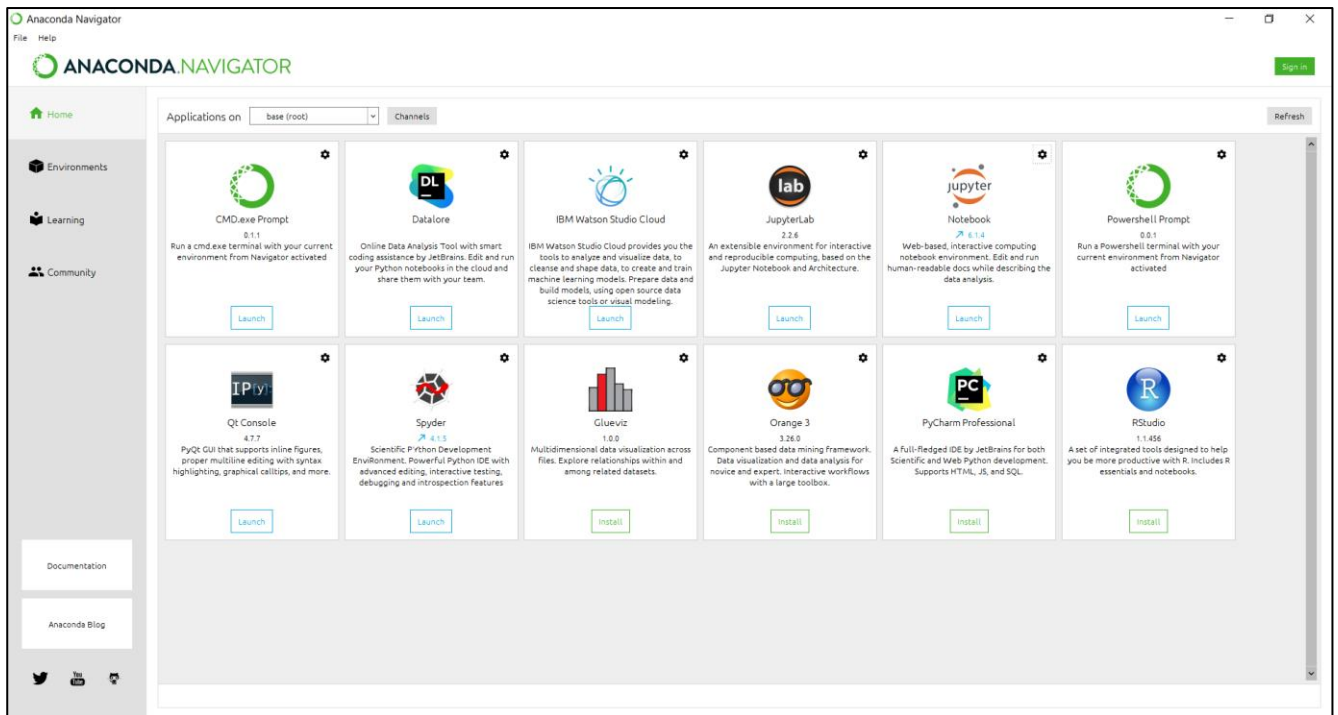


Figura 3-2 Anaconda Navigator

De las aplicaciones que se pueden apreciar en la Figura 3-2, toma especial relevancia “CMD.exe Prompt” que ejecuta el terminal de Anaconda y desde el cual podemos realizar cualquier gestión que necesitemos por medio de comandos como, por ejemplo, instalar librerías externas.

También cabe resaltar la importancia de Jupyter Notebook como API (*Application Programming Interface*) de código abierto basada en web en el desarrollo de este proyecto. Se trata de una interfaz notebook (también llamada “cuaderno computacional”), es decir, un entorno virtual de cuaderno que combina la funcionalidad del software de procesamiento de textos con ambos *shell* y *kernel* del lenguaje de programación.

3.1.1.1 Bibliotecas

El siguiente paso es comprobar que se dispone de todas las librerías necesarias para llevar a cabo las tareas de programación y visualización de datos. Accediendo a la pestaña “environments” y seleccionando el entorno base, que será el que se utiliza por defecto, se verifica utilizando el buscador y los filtros, la correcta instalación e implementación de los siguientes *plugins*:

1. **Matplotlib:** es una biblioteca de código abierto para la generación de gráficos de excelente calidad a partir de datos contenidos en listas o *arrays* en el lenguaje de programación Python y su extensión matemática NumPy. Es multiplataforma y puede ser usada desde *scripts* o desde una consola de Python.

2. **NumPy**: es una librería de código abierto para Python que da soporte a la creación de vectores y matrices de gran tamaño multidimensionales, junto a una gran colección de funciones matemáticas de alto nivel para operar con ellas.
3. **Pandas**: es una biblioteca de software libre escrita como extensión de NumPy, para la manipulación y análisis de datos en Python de alto rendimiento. Ofrece concretamente estructuras de datos y operaciones para manipular tablas numéricas y series temporales. Fue creada originalmente para la gestión de datos financieros, y como alternativa al uso de hojas de cálculo.
4. **Scikit-Learn**: Es, probablemente, la biblioteca de código abierto más útil para nuestro propósito, pues proporciona una amplia gama de algoritmos de aprendizaje supervisados y no supervisados en Python. Está construida sobre SciPy (*Scientific Python*) e incluye varios algoritmos de clasificación, regresión y análisis de grupos, entre los que destaca el algoritmo DBSCAN, del que se hará uso en este proyecto.

Fuera de esta lista por no estar incluida dentro de las librerías que ofrece Anaconda, pero que es fundamental para este trabajo, está **Cartopy**, una librería para trabajar con mapas que posibilita el análisis de datos geoespaciales. Esta librería hace uso de las librerías NumPy, Shapely y PROJ. 4 e incluye una interfaz construida sobre Matplotlib para la creación de mapas de calidad. Al tener instalado “conda”, la manera en la que se realizó la instalación es muy sencilla, ejecutando el siguiente comando en la terminal de Anaconda:

```
conda install -c conda-forge cartopy
```

Habiendo comprobado e instalado las librerías arriba reseñadas, el entorno de trabajo en el que nos moveremos queda listo para ser explotado.

3.2 Datos AIS: Campos del mensaje

Para el desarrollo de este trabajo se empleará un fragmento de una base de datos AIS facilitada por el COVAM, ya que procesar todos los datos supondría una carga computacional excesiva, y no es este el objetivo del proyecto, sino evaluar el comportamiento de algoritmos de *clustering* al entregarles una base de datos de esta naturaleza. Estos mensajes poseen un formato que será descrito a continuación en la Tabla 3-1, basándose en la información proporcionada en [62] y [63].

Campo	Tipo de dato (Python)	Descripción
MMSI	int	Número MMSI del buque (identificador AIS)
TIME	datetime	Fecha y hora (UTC) de la emisión
LATITUDE	float	Latitud geográfica (WGS84)
LONGITUDE	float	Longitud geográfica (WGS84)
COG	float	Rumbo sobre fondo (grados)
SOG	float	Velocidad sobre fondo (nudos)
HEADING	int	Rumbo del buque (grados). Un valor de 511 indica que no hay datos del rumbo
NAVSTAT	int	Estado de navegación de acuerdo con especificación AIS
PAC	bool	Position accuracy. 1= error menor de 10 metros; 0 = error mayor de 10 metros
ROT	int	Rate of turn, velocidad de giro de la embarcación. Se mide en grados por minuto.
IMO	int	Número IMO del buque
NAME	string	Nombre del buque
CALLSIGN	string	Distintivo de llamada del buque
TYPE	int	Tipo de buque de acuerdo con especificación AIS
A	int	Distancia en metros de la antena a la proa del buque
B	int	Distancia en metros de la antena a la popa del buque
C	int	Distancia en metros de la antena al costado de babor del buque
D	int	Distancia en metros de la antena al costado de estribor del buque
DRAUGHT	float	Calado del buque en metros
DEST	string	Puerto de destino (introducido manualmente)
ETA	string	Tiempo estimado de llegada en formato completo fecha/hora
SRC	string	Fuente (Source) de los datos AIS: Terrestre (TER) o Satélite (SAT)
ZONE	string	Nombre de la zona donde se encuentra el buque
ECA	bool	Indica si el barco se encuentra en zona ECA/SECA (Zona de control de emisiones)

Tabla 3-1 Campos de un mensaje AIS

3.3 Evaluación de los algoritmos

El análisis que se realizó para evaluar el comportamiento de los algoritmos tiene por objetivo detectar las zonas donde los barcos se encontraban potencialmente parados, como pueden ser inicios/finales de rutas marítimas o zonas de pesca. Esto resulta ser de gran utilidad a la hora de particionar rutas marítimas a modo de análisis previo antes de llevar a cabo un análisis más exhaustivo.

Con la intención de cumplir con el objetivo expuesto, se realizó un filtrado de los datos, volcando varias columnas de los mensajes cuyo campo “sog” era igual 0 en un archivo csv (*Comma Separated Values*). Para realizar los cálculos y los gráficos, se seleccionaron únicamente las columnas de “Latitud” y “Longitud” para computar dentro del algoritmo, pues determinan las posiciones espaciales de los buques y se consideran los únicos campos verdaderamente fiables en los mensajes AIS.

Teniendo en cuenta la zona geográfica a la que corresponde el conjunto de datos, también se contemplaba la opción de detectar buques realizando *bunkering*, una actividad que durante los últimos años ha tenido una repercusión mediática considerable (ver Figura 3-3).



Figura 3-3 Suministro de combustible en alta mar de un barco a otro (*bunkering*) [64]

La zona del Estrecho de Gibraltar es internacionalmente conocida por su gran actividad de repostaje en la mar, maniobra que realizan los barcos en navegación que elimina la necesidad de parar en puerto para conseguir combustible. El motivo detrás de esto es que Gibraltar, exento de tasas, ofrece los precios más económicos de todo el Mediterráneo, además de tratarse de una zona estratégica al permitir a los navíos no desviarse de su ruta para repostar. Los buques que realizan estas maniobras como proveedores a menudo se encuentran fondeados o incluso a la deriva, lo que supone un grave peligro para los barcos que discurren por la zona, incrementando la probabilidad de sufrir un abordaje. Además, es frecuente que durante esta actividad se produzcan vertidos tanto por derrames como por accidentes, lo que supone una importante amenaza a la fauna y la flora marinas.

La tarea de evaluar el resultado del *clustering* puede suponer un desafío, pues no resulta sencillo definir cuando un resultado se considera aceptable, ya que esto depende de la subjetividad del analista.

Es por esta razón que existen técnicas e índices para ayudar con esta tarea. Existen dos tipos de validación:

1. *Validación externa*: basadas en información conocida con anterioridad.
2. *Validación interna*: basadas en la información que ofrecen los resultados.

Para nuestro caso, emplearemos como medidas de validación interna el número de grupos y *outliers* que ofrece el resultado, y el *Coeficiente Silueta*. Este coeficiente, dado un punto i del conjunto de datos, se basa en las siguientes variables:

- Cohesión $a(i)$: distancia promedio del punto a todos los demás, dentro de su mismo clúster.
- Separación $b(i)$: distancia promedio del punto a todos los demás puntos del clúster más cercano.

De esta manera, la silueta del objeto i se define como:

$$s(i) = \frac{b(i) - a(i)}{\max [a(i), b(i)]}$$

Este parámetro toma valores $-1 \leq s(i) \leq 1$. Observaciones con un valor cercano a 1 estarán bien agrupados, un valor cercano a 0 indicaría que la observación se encuentra entre dos grupos, y un valor cercano a -1 indicaría que la observación está mal asignada a su clúster.

El Coeficiente Silueta es el promedio a través de todas las observaciones, siendo más favorable cuanto mayor sea su valor. Según Kauffman y Rousseeuw [65], un valor por encima de 0,7 indica un buen agrupamiento; entre 0,5 y 0,7 un grado de agrupamiento razonable; un valor entre 0,25 y 0,5 una estructura débil, y por debajo de 0,25 indicaría que no se ha encontrado ningún agrupamiento reseñable.

3.3.1 DBSCAN

La primera implementación que se realizó fue la del algoritmo basado en densidad DBSCAN, cuyo código puede encontrarse en el Anexo I: Código DBSCAN. Este algoritmo es uno de los que ofrece la librería Scikit-learn, por lo que, para su importación, se usó la sentencia:

```
from sklearn.cluster import DBSCAN
```

El algoritmo presentado se ejecuta por defecto utilizando la distancia euclídea. Además, es especialmente sensible a los parámetros (ϵ , MinPts), por lo que fue necesario determinar el valor óptimo entre las infinitas combinaciones que pueden tomar. Para ello, se tuvieron en cuenta las siguientes consideraciones:

Minimum Samples (*MinPts*)

No existe un método que determine automáticamente el mejor valor para este parámetro en DBSCAN, por lo que para establecerlo hay que apoyarse en el conocimiento del dominio y del conjunto de datos utilizados. Sin embargo, realizando labores de investigación, se encontraron algunas reglas generalmente aceptadas pero que, al no poder asegurar su precisión y fiabilidad en todos los casos, no se consolida como norma:

- Cuanto más grande el *dataset*, mayor debe ser el valor de MinPts.
- Si se espera que el *dataset* tenga un ruido considerable, se debe escoger un mayor valor.
- Generalmente, MinPts debe ser mayor o igual a la dimensionalidad del conjunto de datos.
- Para un conjunto de datos 2D, se puede usar el valor por defecto de MinPts = 4.
- Si el *dataset* tiene más de dos dimensiones, se elegirá un valor para MinPts = 2*dimensiones.

Epsilon (ϵ)

Después de seleccionar un valor para MinPts, se procede a calcular un valor para ϵ . Para ofrecer una referencia para comprender el desarrollo del trabajo, es necesario puntualizar que un valor de $\epsilon = 0,0167$ se corresponde con, aproximadamente, una milla náutica ($\epsilon = 0,1$ equivale a 6 MN u 11,112km). Esto se debe a que un valor de 1° en el mapa se corresponde a 60 millas náuticas, por utilizar la proyección cilíndrica equidistante (*plate carrée*) que ofrece Cartopy. Esta proyección es utilizada en el procesamiento de los mapas globales por su correspondencia entre píxeles y posición geográfica.

Otro factor a tener en cuenta acerca de las operaciones de este apartado y el que le sigue es que el número total de posiciones que procesarán los algoritmos es de 24233, valor que deberá tenerse en cuenta para valorar los resultados, en función del número de grupos y de *outliers*.

Para encontrar un buen punto de inicio, existe una técnica para determinar el valor óptimo, descrita en [66]. Básicamente, esta técnica propone calcular la distancia media entre cada punto y sus *k-nearest neighbors*, es decir, sus $k = \text{MinPts}$ vecinos. Tras hacerlo, se grafican los puntos en orden ascendente en una gráfica *k-distance*. El valor más adecuado de ϵ para el número mínimo de vecinos establecido se encuentra en el punto de máxima curvatura (popularmente conocido como el “codo”), que coincide con la distancia media entre todos los puntos del conjunto. Existe una librería que se puede importar de Scikit-learn para realizar esta labor, usando la sentencia:

```
from sklearn.neighbors import NearestNeighbors
```

Por tanto, teniendo en cuenta lo mencionado, atendiendo a la naturaleza del conjunto de datos, se comenzó por dar un valor de MinPts = 20. Para este valor, se obtuvo la siguiente gráfica *k-distance* (Figura 3-4):

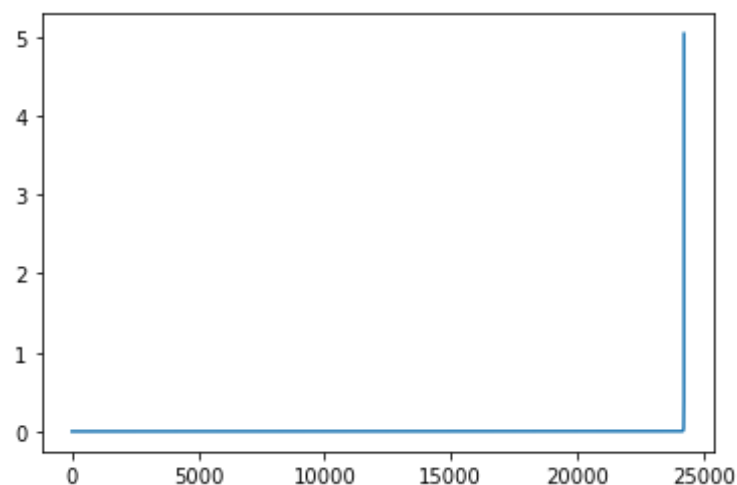


Figura 3-4 Gráfica *k-distance* para MinPts=20

Haciendo zoom en la gráfica (Figura 3-5), se tomó la decisión de probar $\epsilon=0.1$, aunque se sospechaba que 6 millas náuticas sería un valor demasiado elevado. Realizando pruebas se observó que este valor no se veía apenas afectado por el que toma MinPts, para este conjunto de datos.

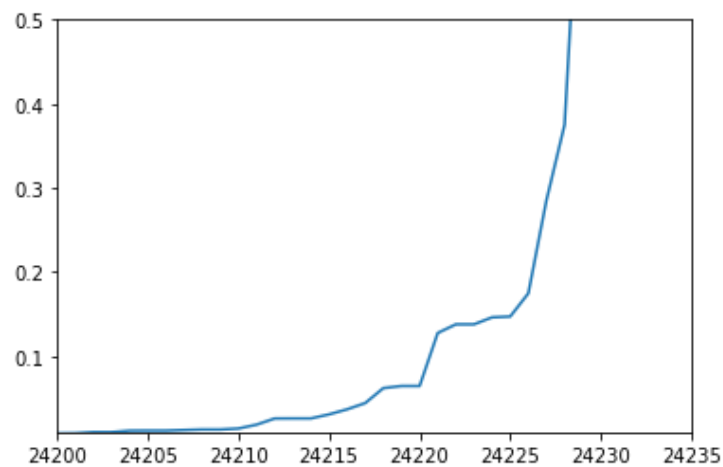


Figura 3-5 Zoom en la zona de interés de la Figura 3-4

Como punto de partida, se ejecutó el algoritmo bajo estos parámetros (MinPts = 20, $\epsilon=0.1$), se obtuvo el resultado presentado en la Figura 3-6.

Para comprender los mapas expuestos en la memoria, se exponen las siguientes consideraciones:

- Los puntos de distintos colores hacen referencia a grupos que el algoritmo ha sido capaz de discriminar.
- Los puntos morados que se aprecian a lo largo de todo el mapa y que aparecen de forma distribuida en el mismo, son los *outliers*.

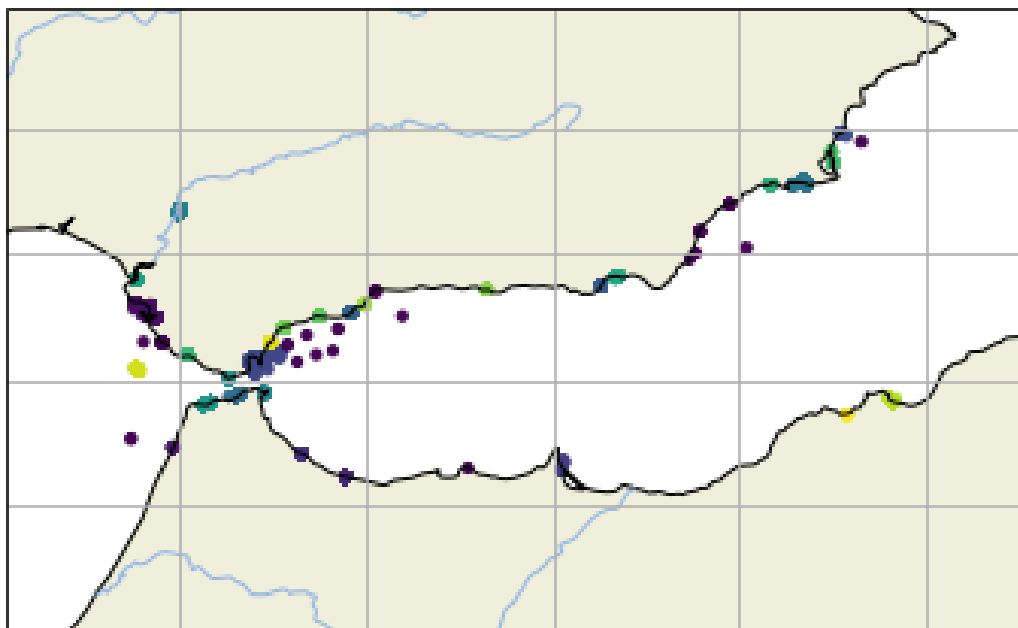


Figura 3-6 Plot para MinPts = 20 y $\epsilon=0.1$

```
Estimated number of clusters: 28
Estimated number of noise points: 47
Silhouette Coefficient: 0.923
```

Confirmando lo expuesto acerca del valor de ε , se observa que los grupos abarcan áreas demasiado amplias en una zona donde interesa más precisión, por tratarse de un área por donde transita un elevado número de barcos diariamente.

Este resultado no se consideraba adecuado, por lo que, a modo de orientación, se construyó la siguiente tabla:

MinPts	Grupos	Outliers	C. Silueta
4	32	16	0,917
8	30	27	0,916
10	28	45	0,923
15	28	45	0,923
20	28	47	0,923
30	25	121	0,918
40	24	165	0,916
50	23	205	0,913
75	22	256	0,91
100	21	331	0,909
150	18	725	0,884
200	17	896	0,871
250	16	1123	0,855
300	16	1123	0,855
350	14	1744	0,805

Tabla 3-2 Parámetros de validación para diferentes valores de MinPts

Se puede apreciar en la Tabla 3-2 que, para un valor aproximado de $10 < \text{MinPts} < 20$ se obtiene el mejor coeficiente de silueta, por lo que se decidió continuar la investigación con un valor $\text{MinPts} = 20$

Dado que en esta escala puede resultar complicado apreciar los diferentes grupos y, por tanto, no se explotarían al máximo los resultados del análisis, se disminuyó la escala enfocando zonas de mayor interés.

Focalizando en la zona del Estrecho de Gibraltar, se extrajo un gráfico de menor escala (Figura 3-7) y se estudió en contraste con una carta sacada del programa Open CPN (Figura 3-8).

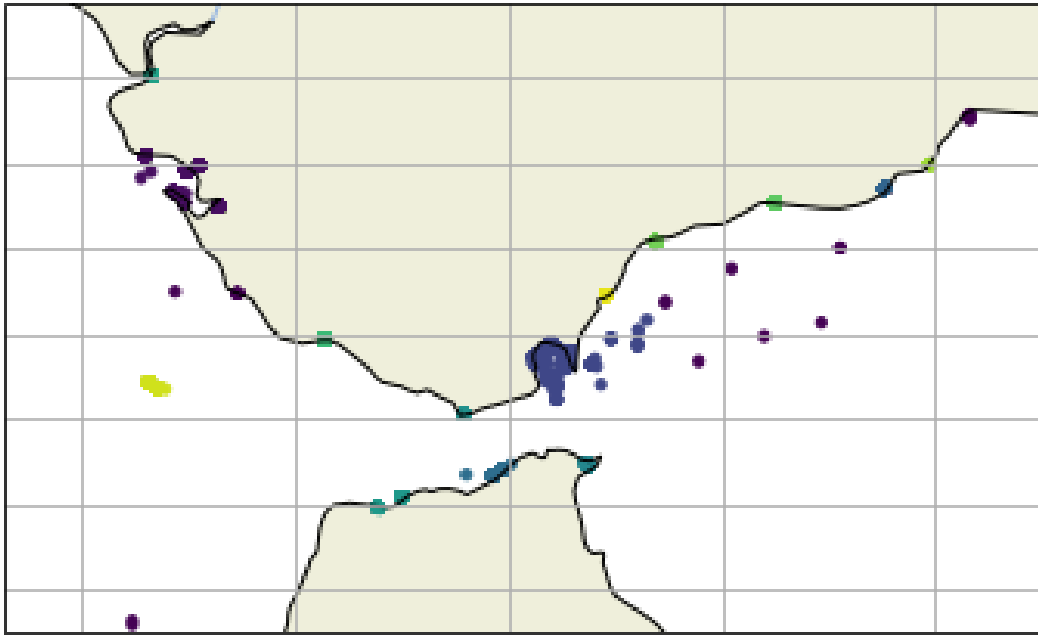


Figura 3-7 Zoom del plot (MinPts = 20 y $\epsilon = 0.1$) en la zona del Estrecho de Gibraltar

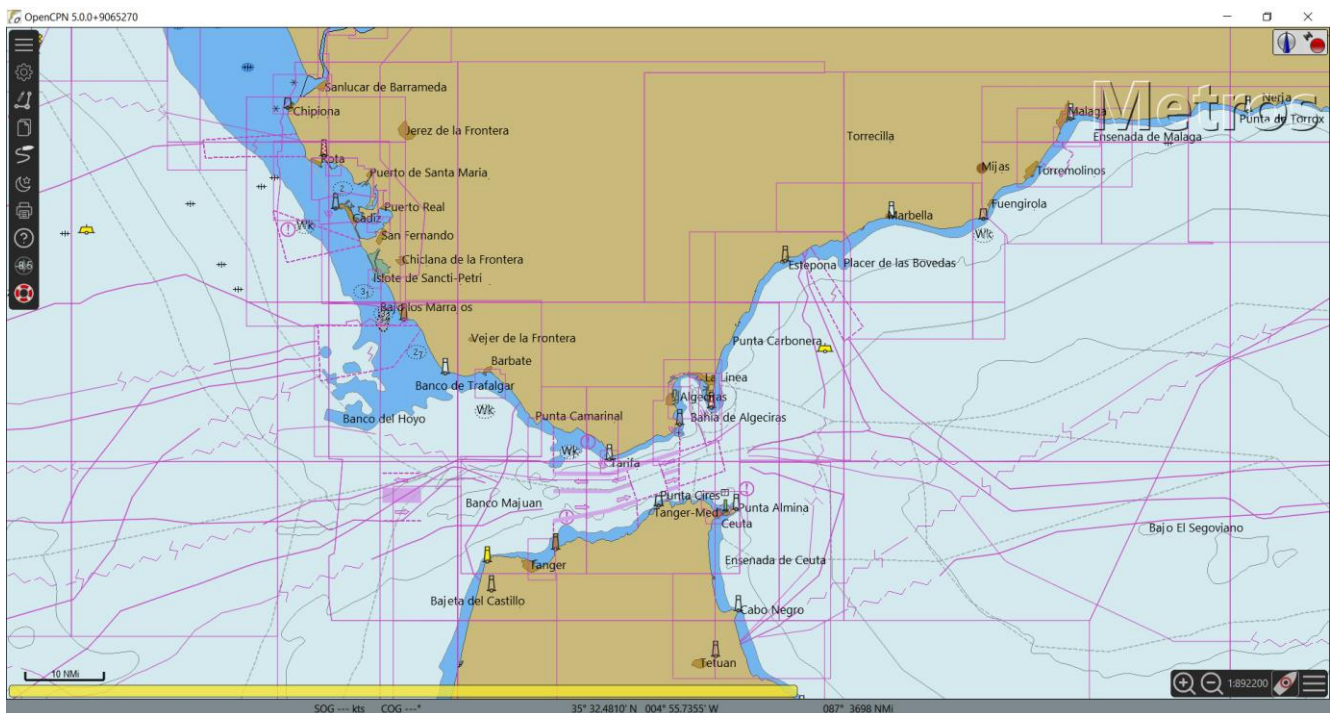


Figura 3-8 Estrecho de Gibraltar en OpenCPN

Comparando las figuras se realizan las siguientes observaciones:

- Existen grupos que se localizan en zonas de bajos, probablemente formados por barcos faenando.
- En la bahía de Algeciras existe una gran concentración de barcos, lo que resulta lógico al tratarse de uno de los puertos más grandes y concurridos de España. Lo más probable es que los grupos los conformen barcos fondeados. También se detectó un número considerable de agrupamientos en la bahía de Cádiz.

- Se detectaron otros clústeres en los puertos de Málaga, Fuengirola, Marbella, Estepona, Sotogrande, Cartagena, Tarifa, Tánger, Melilla, Barbate y Sanlúcar de Barrameda.
- Existen otros clústeres en zonas no identificadas como zonas de pesca, los cuales podrían resultar contactos de interés.

Aún se esperaba mejorar el análisis, por lo que se disminuyó la escala, centrándola primero en el puerto de Algeciras por ser el lugar de mayor actividad, factor que se puede aprovechar para evaluar el comportamiento del algoritmo con más precisión.

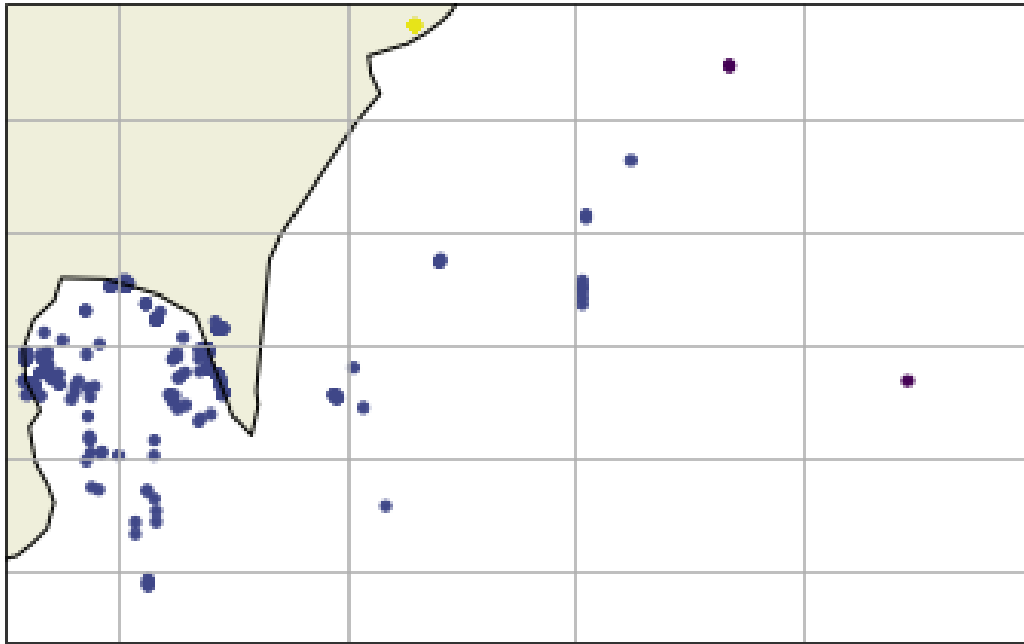
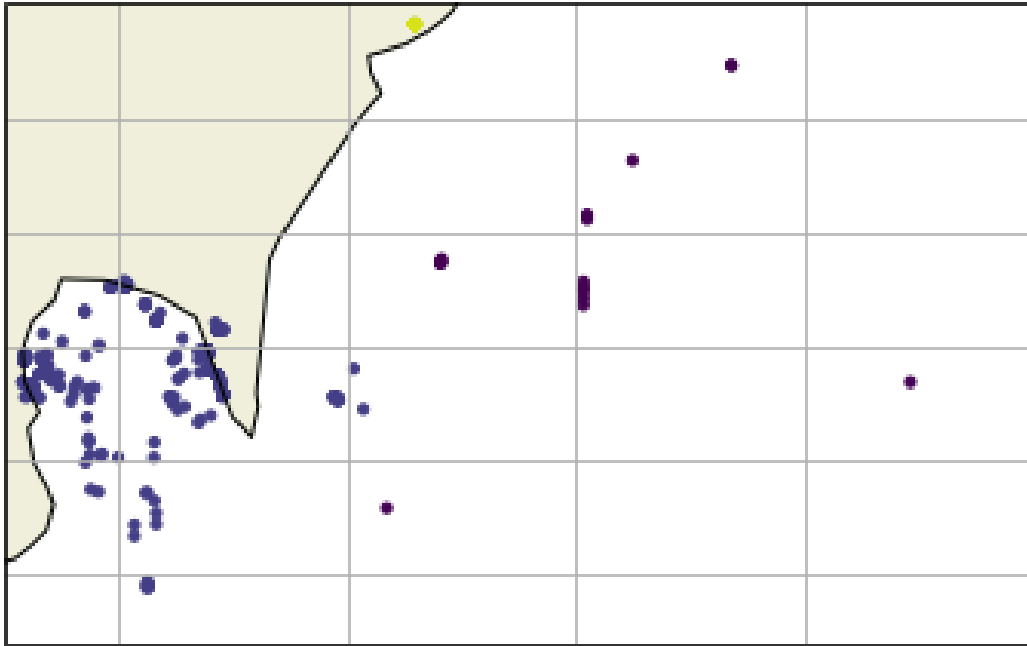


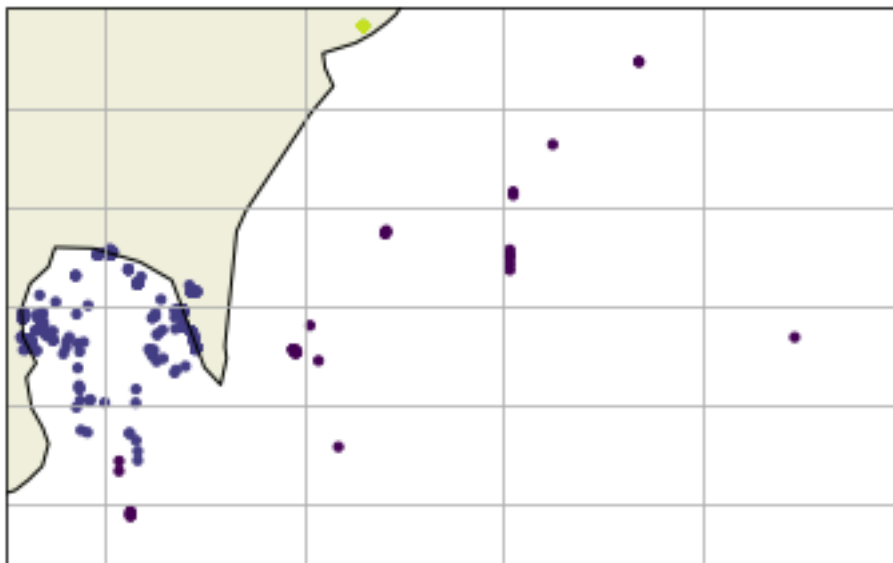
Figura 3-9 Plot para $\text{MinPts} = 20$ y $\varepsilon = 0.1$, centrado en Algeciras y el Mar de Alborán

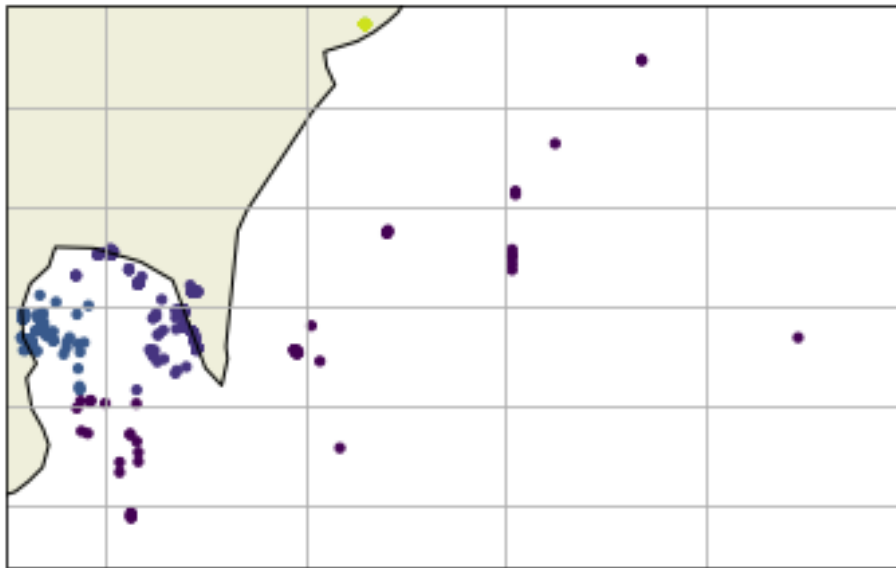
Apoyándose en la Figura 3-9 se observó que para $\varepsilon = 0.1$ los agrupamientos no satisfacían correctamente los resultados esperados, pues existen buques que se encuentran en el exterior de esta zona que han sido incluidos junto a todos los barcos de la bahía. Es por ello que se realizaron pruebas con un valor de ε menor, buscando que estos barcos se separen.

Estableciendo $\varepsilon = 0.05$ **Figura 3-10 Plot centrado en Algeciras para $\varepsilon = 0.05$**

```
Estimated number of clusters: 33  
Estimated number of noise points: 78  
Silhouette Coefficient: 0.943
```

Los resultados presentados en la Figura 3-10 consiguen, sin lugar a dudas, resultados más satisfactorios. No sólo incrementó el valor del coeficiente de Silueta, sino que se logró separar parcialmente los barcos exteriores del grupo de Algeciras. A pesar de todo, se siguió disminuyendo el valor de épsilon para mejorar los últimos resultados.

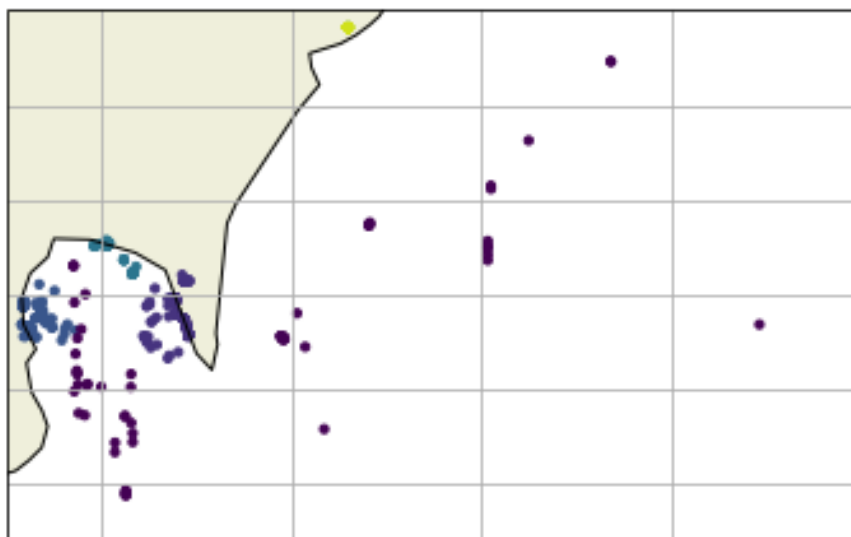
Estableciendo $\varepsilon = 0.03$ **Figura 3-11 Plot centrado en Algeciras para $\varepsilon=0.03$**

Estableciendo $\varepsilon = 0.02$ **Figura 3-13 Plot centrado en Algeciras para $\varepsilon=0.02$**

```
Estimated number of clusters: 40
Estimated number of noise points: 157
Silhouette Coefficient: 0.949
```

En la Figura 3-13 se demuestra que disminuyendo el valor de épsilon a $\varepsilon=0.02$ se consigue la división de los barcos de la bahía de Algeciras en dos grupos, uno a cada lado del eje que la divide. Además, barcos fuera de los muelles en la entrada a Algeciras se consideran como *outliers*. Por otro lado, atendiendo a los resultados numéricos, se observa que aumentó el número de clústeres y de *outliers*, así como el coeficiente de silueta.

Aun así, se trató de separar los barcos frente a la playa de Getares del grupo del puerto de Algeciras, para lo que se redujo aún más el valor de ε .

Estableciendo $\varepsilon = 0.014$ **Figura 3-14 Plot centrado en Algeciras para $\varepsilon=0.014$**

Estimated number of clusters: 41
Estimated number of noise points: 170
Silhouette Coefficient: 0.960

A la vista de la Figura 3-14, se llegó a un buen resultado utilizando una configuración de $\epsilon=0.014$ y $\text{MinPts} = 20$. El coeficiente de Silueta era el más elevado, y dentro de la bahía se consiguió diferenciar con bastante precisión el puerto de Algeciras del Puerto de Gibraltar. Además, con esta última prueba se distinguió del resto un nuevo grupo, centrado en el puerto de Campamento.

Para analizar el resultado general, se volvió a establecer la escala inicial para comparar el resultado con el que se obtenía al principio:

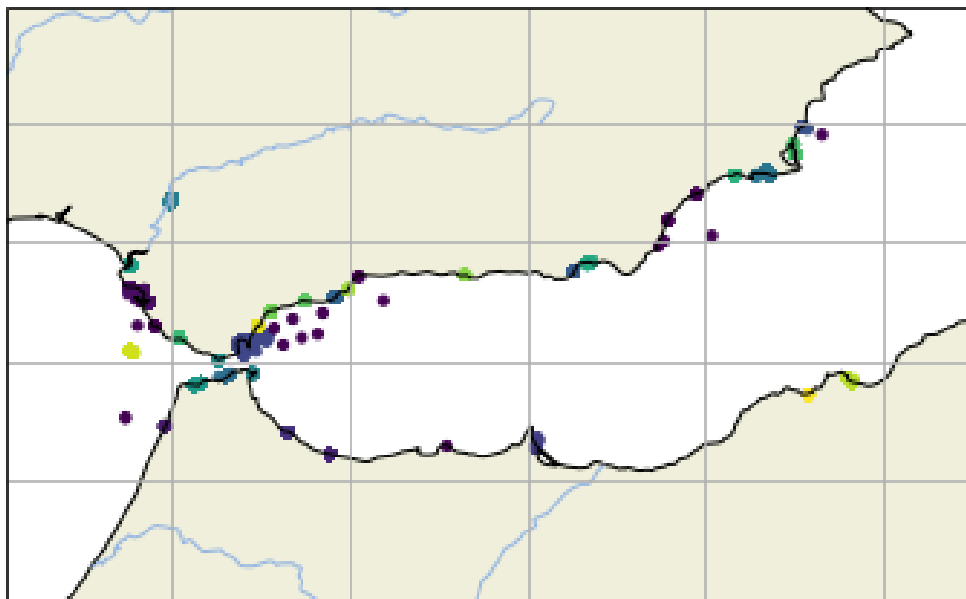


Figura 3-15 Plot para $\epsilon=0.1$, escala inicial

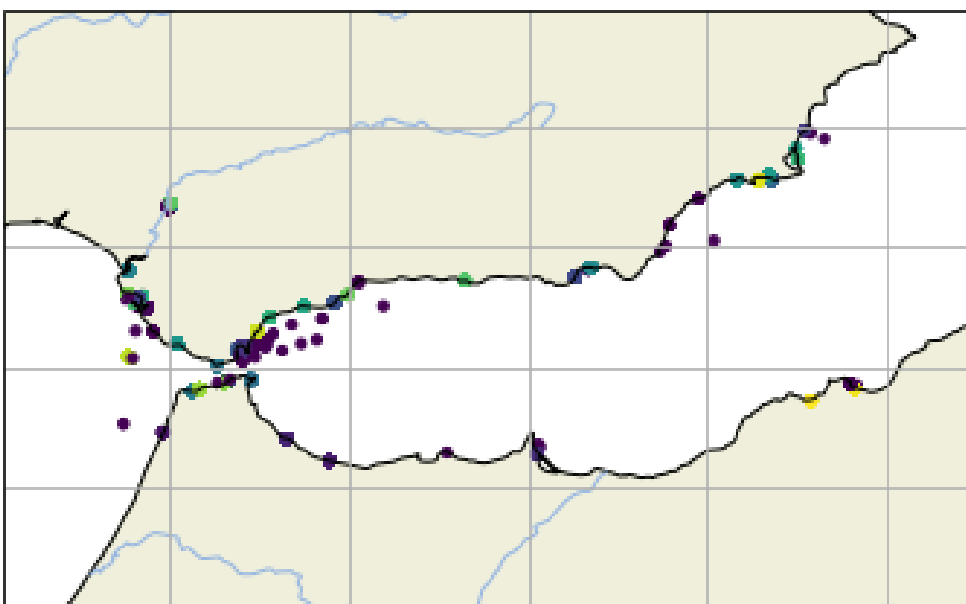


Figura 3-16 Plot para $\epsilon=0.014$, escala inicial

A la vista de las dos representaciones (Figura 3-15 y Figura 3-16) y de los valores analíticos ofrecidos, se observa que el algoritmo mejora sus capacidades en el segundo caso, pues es capaz de detectar un mayor número de grupos, que refleja la precisión de diferenciar áreas congestionadas que deban dividirse en más de un clúster, además de obtener un mejor coeficiente de Silueta. En esta escala, se ve con gran claridad cómo se han dividido algunos grupos en los puertos de Cádiz, Tánger, Melilla, Cartagena, Sevilla y la zona de Orán.

Para comprobar que el algoritmo funciona correctamente, se ampliaron las áreas anteriormente mencionadas.

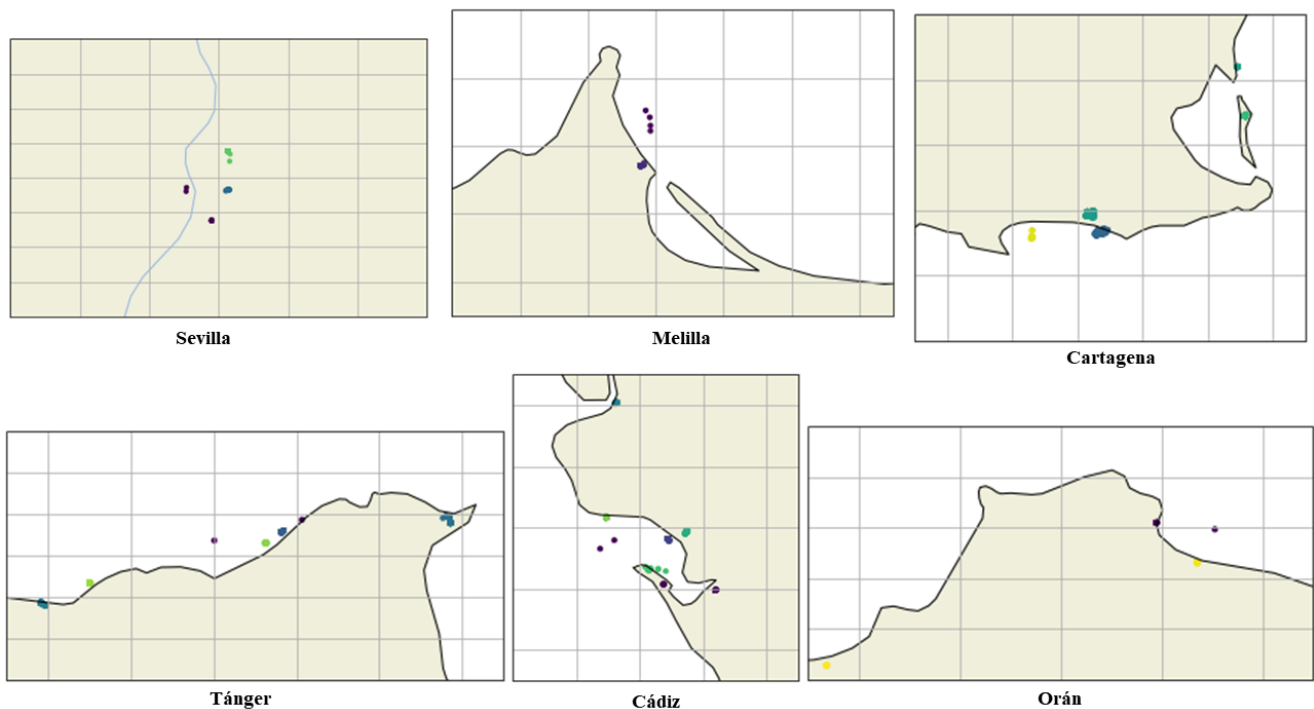


Figura 3-17 Ampliación del plot sobre zonas destacadas

La Figura 3-17 sirve para corroborar el correcto funcionamiento del algoritmo, el cual, trabajando con estos parámetros, es capaz de distinguir entre los grupos con una buena precisión.

Para entender los grupos de Sevilla, se extrajo un mapa de Google Maps de la zona (Figura 3-18).

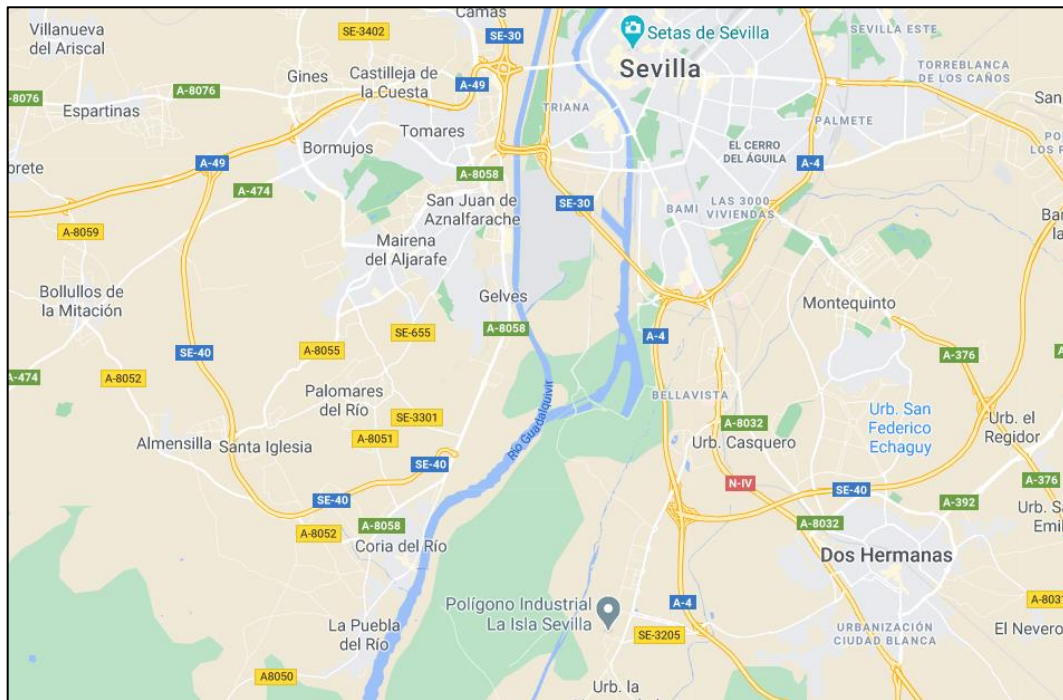


Figura 3-18 Vista de Google Maps del Guadalquivir a su paso por Sevilla

Se observa que los clústeres que aparecen a ambos lados del río en la representación de Cartopy se distribuyen a lo largo de la bifurcación del Guadalquivir.

En el caso de la bahía de Cádiz, los grupos que aparecen en la Figura 3-19 se corresponden con barcos que se encuentran en la Base Naval de Rota, en el puerto de Cádiz, en Puerto Sherry, en el Puerto de Santa María, y con *outliers* en el Arsenal de la Carraca y en medio de la bahía, conformados por barcos independientes o grupos de entidad menor.

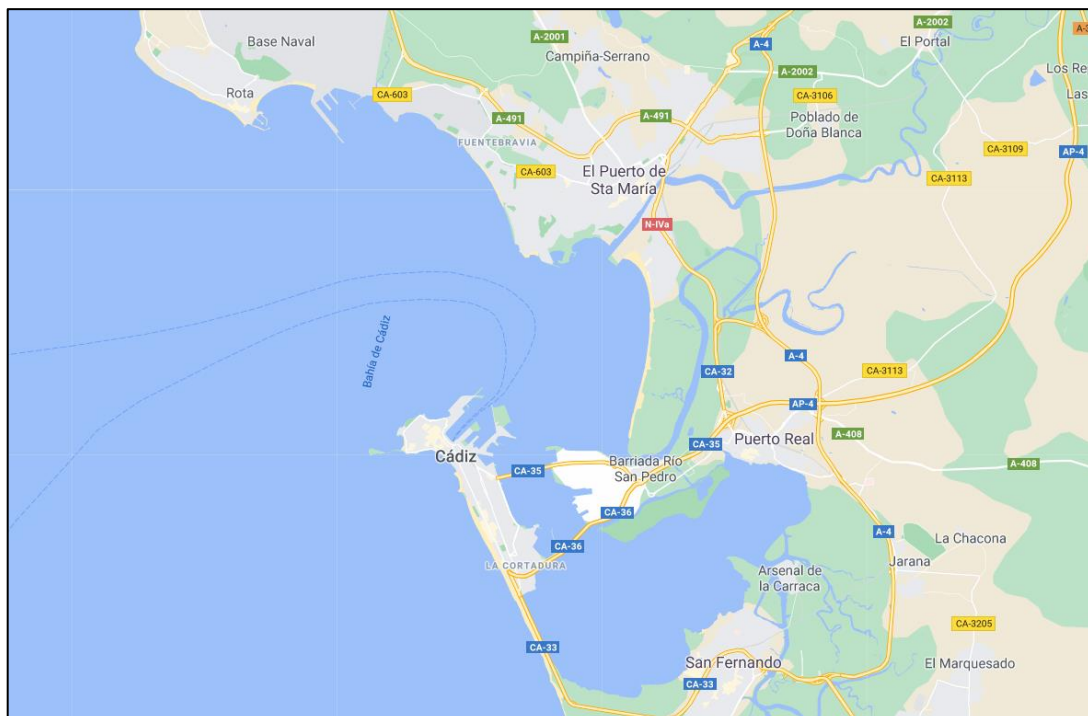


Figura 3-19 Vista de Google Maps de la Bahía de Cádiz

Atendiendo al tipo de datos que se manejan, es razonable considerar que los resultados obtenidos hasta el momento eran, realmente, una aproximación. Si bien cabe la posibilidad de que la precisión lograda utilizando la distancia euclídea pueda satisfacer el funcionamiento del algoritmo para el objetivo que se plantea, lo más correcto es utilizar distancias esféricas debido a la geometría del planeta. Por ello, se realizaron pruebas similares utilizando la distancia Haversine, descrita a continuación.

Distancia “Haversine” – Fórmula del semiverseno

El algoritmo DBSCAN trabaja por defecto con la distancia euclídea, pero teniendo en cuenta que este trabajo busca agrupar posiciones geoespaciales, resulta natural usar la distancia que las une por círculo máximo, dado que la Tierra es aproximadamente una esfera (ver Figura 3-20). La distancia Haversine puede ser definida como la distancia angular entre dos puntos de la superficie de la Tierra.

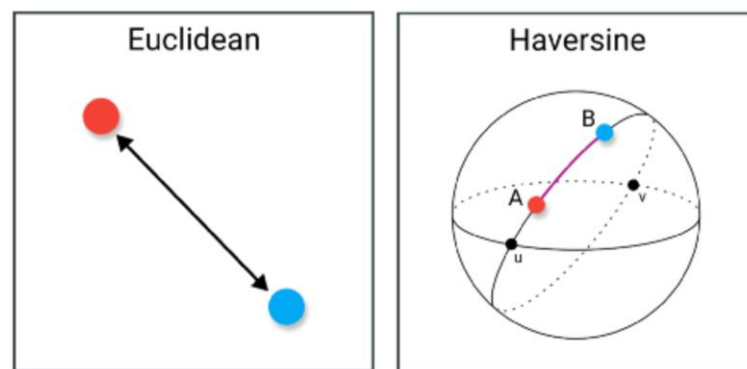


Figura 3-20 Distancia euclídea vs Distancia Haversine (basado en [67])

Además, el algoritmo utilizado por el módulo *NearestNeighbors* para calcular las distancias entre los puntos cercanos se cambió a “*ball_tree*”, que reduce la complejidad de la búsqueda de puntos vecinos.

Para el correcto funcionamiento de DBSCAN con la distancia Haversine, es necesario entregarle al algoritmo los parámetros en radianes. Se consideró como distancia epsilon inicial de 3 kilómetros, dato que también debe ser convertido a radianes. Para dichas conversiones, se modificó el algoritmo de la siguiente manera:

```
NM_per_radian = 6371.0088
epsilon = 3 / kms_per_radian
db = DBSCAN (eps=epsilon, min_samples=20, algorithm='ball_tree',
metric='haversine').fit(np.radians(X))
```

El resultado de esta operación se presenta en la Figura 3-21 y la Figura 3-22.

Estableciendo distancia 3 km

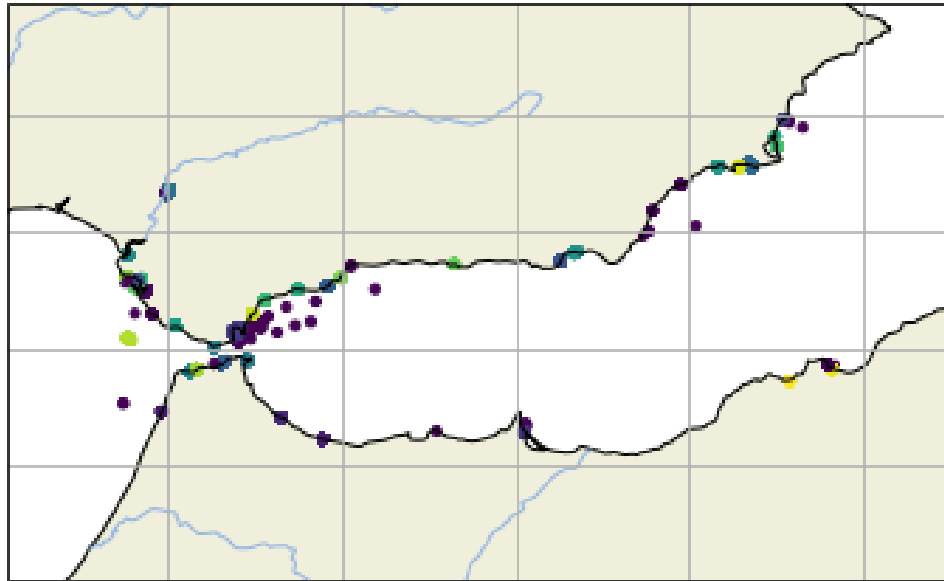


Figura 3-21 Plot para $\epsilon = 3$ km utilizando distancia Haversine

Estimated number of clusters: 35
Estimated number of noise points: 112
Silhouette Coefficient: 0.948

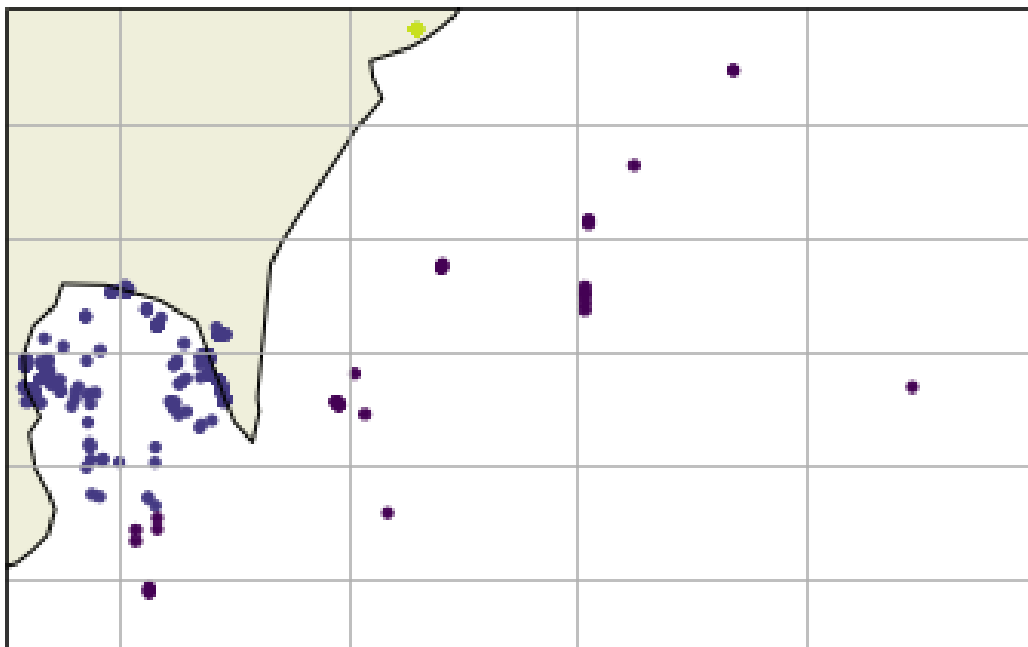
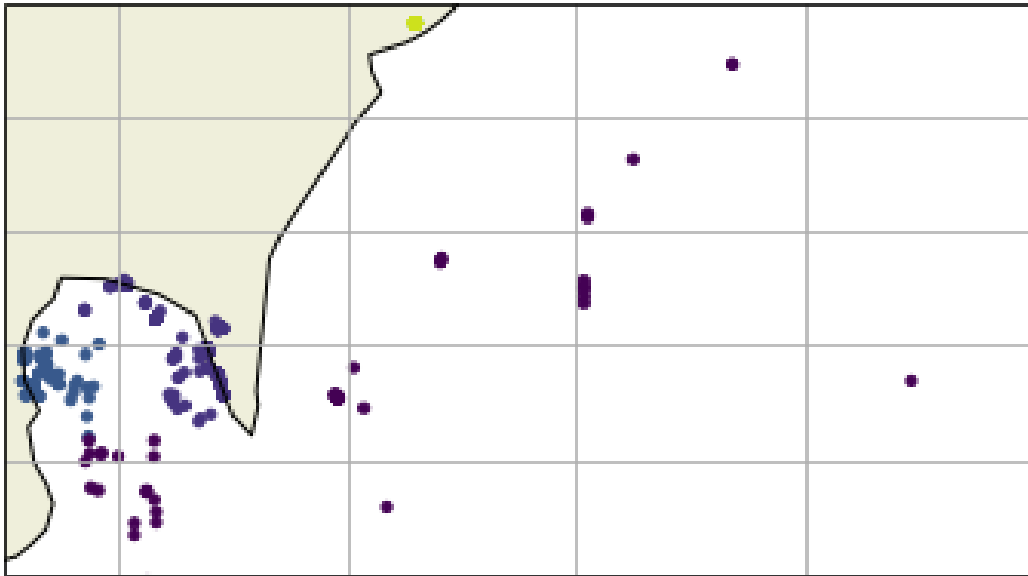


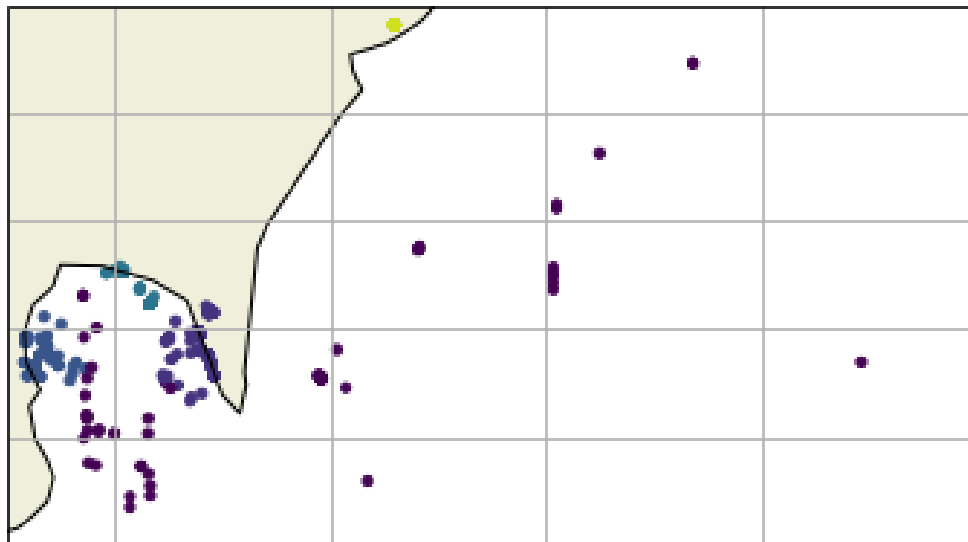
Figura 3-22 Plot de Algeciras para $\epsilon = 3$ km, distancia Haversine

El resultado inicial obtenido para la zona de Algeciras proporcionó unos resultados similares a los de la Figura 3-11. Para acercarse a los resultados obtenidos con la métrica euclídea para $\epsilon=0.014$, era necesario llevar a cabo un reajuste de los parámetros. Para ello, se realizaron pruebas con otras distancias.

Estableciendo distancia 2 km**Figura 3-23 Plot de Algeciras para $\epsilon = 2$ km, distancia Haversine**

```
Estimated number of clusters: 40  
Estimated number of noise points: 161  
Silhouette Coefficient: 0.949
```

Para una distancia ϵ de 2 km con distancia Haversine, se obtuvieron unos resultados similares a los de la Figura 3-13, por lo que se volvió a disminuir la distancia para buscar agrupaciones de menor entidad.

Estableciendo distancia 1.5 km**Figura 3-24 Plot de Algeciras para $\epsilon = 1.5$ km, distancia Haversine**

```
Estimated number of clusters: 41  
Estimated number of noise points: 173  
Silhouette Coefficient: 0.960
```

Para la configuración de la Figura 3-24 se consiguieron resultados casi idénticos a los que se consideraron óptimos para la distancia euclídea, con una diferencia mínima en el número de *outliers*.

Con intención de hilar más fino a partir de este punto, se extrajo una imagen satelital de Google Maps de la zona, de manera que se puedan apreciar las características físicas de la bahía.

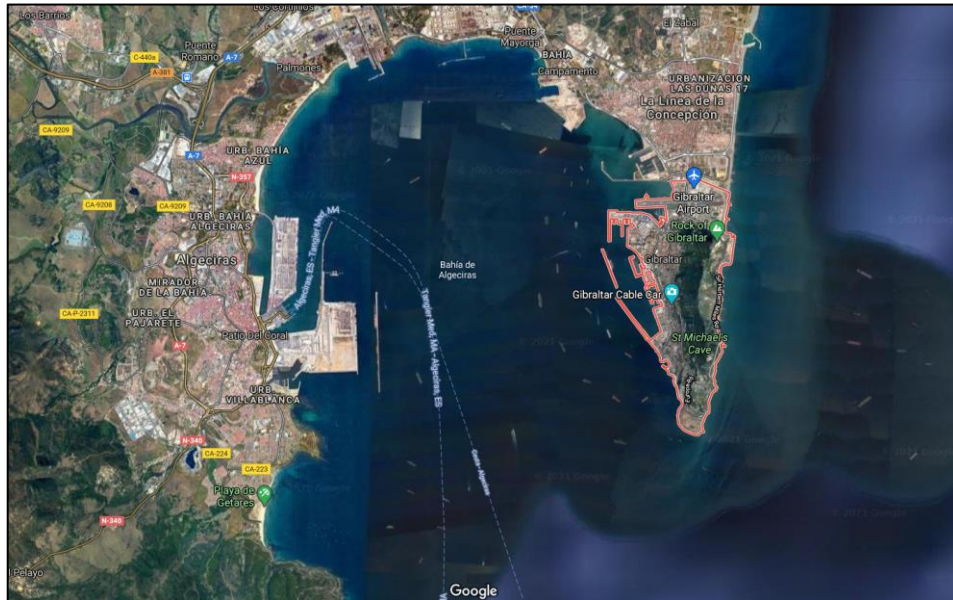


Figura 3-25 Mapa de Algeciras-Gibraltar extraído de Google Maps

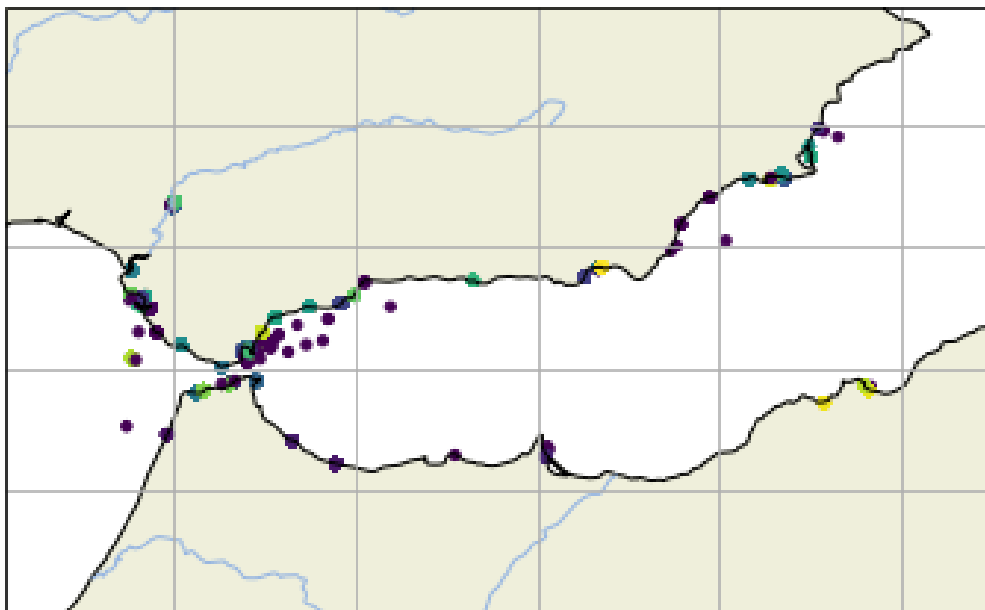
Si se comparan las figuras, se puede apreciar con claridad que barcos del exterior del puerto de Gibraltar están siendo asignados al mismo grupo que los que se encuentran dentro, así como los de la dársena que se encuentra inmediatamente hacia el norte. También se observa que los barcos del muelle de Campamento, al norte de Gibraltar, y los del siguiente amarradero se engloban bajo el mismo grupo. Por otro lado, en la Figura 3-25, se aprecian las rutas marítimas Algeciras-Tánger y Algeciras-Ceuta, que en la Figura 3-24 se ven representadas por *outliers*, probablemente por la escasa entidad de los grupos.

Aprovechando esta información, se realizó una prueba con un valor ϵ más pequeño para separar los grupos, y un valor $\text{min_samples}=10$ para detectar grupos donde hasta ahora solo había *outliers*.

Estableciendo distancia 1 km y min_samples=10**Figura 3-26 Zoom de Algeciras para $\epsilon = 1$ km y min_samples=10, distancia Haversine**

```
Estimated number of clusters: 49  
Estimated number of noise points: 124  
Silhouette Coefficient: 0.955
```

Se observa en la Figura 3-26 que ahora ha sido posible discriminar los diferentes grupos de barcos dentro de la bahía con una alta precisión, por lo que se considerarán estos valores como óptimos para este algoritmo. El cuadro general de la zona del Estrecho de Gibraltar – Mediterráneo, para este conjunto de datos y utilizando esta configuración, queda según se puede ver en la Figura 3-27, dejando los *outliers* mostrados en la Figura 3-28.

**Figura 3-27 Plot general $\epsilon = 1$ km y min_samples=10, distancia Haversine**

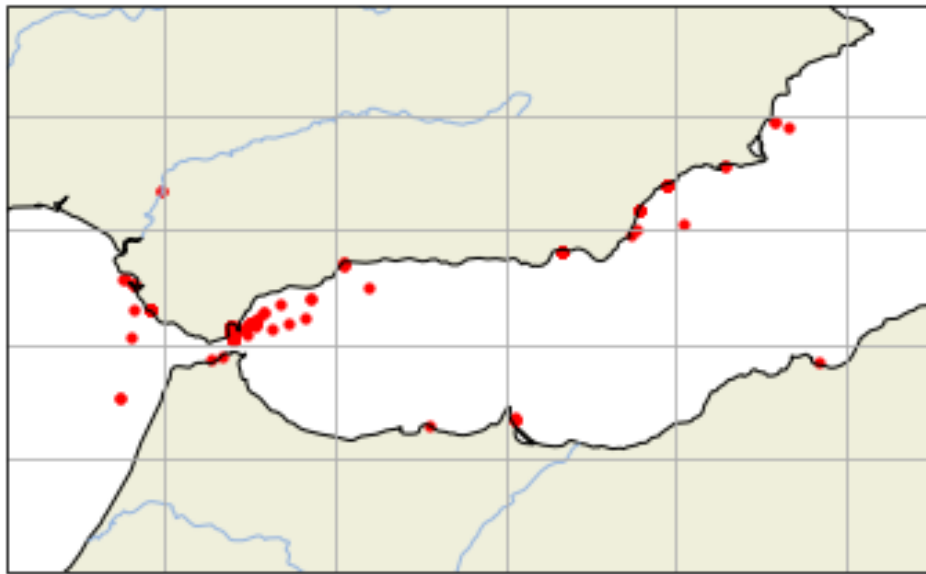


Figura 3-28 Outliers resultantes para $\epsilon = 1$ km y $\text{min_samples}=10$, distancia Haversine

Llama la atención la cantidad de ruido detectado en el Mar de Alborán y Algeciras, por lo que se decidió inspeccionar la zona reduciendo la escala del plot (ver Figura 3-29).

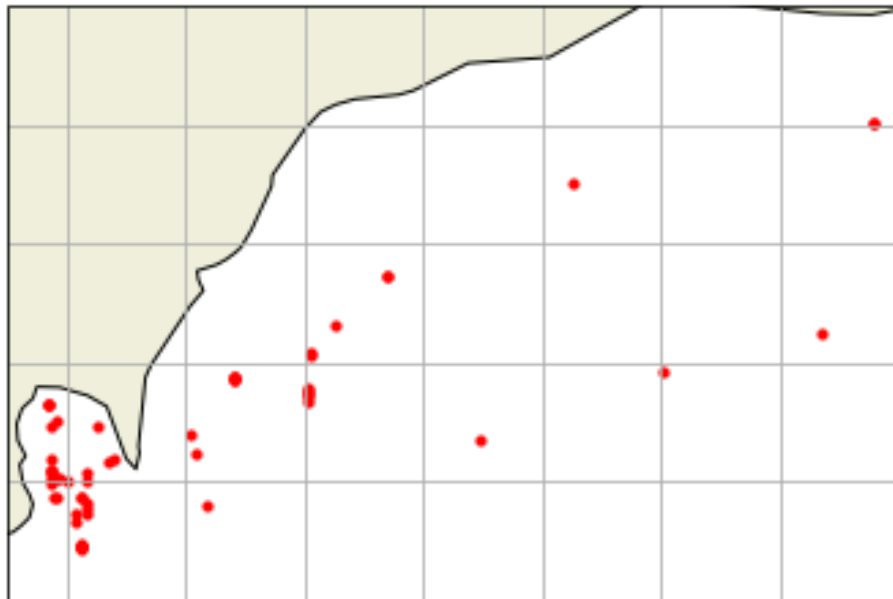


Figura 3-29 Zoom sobre los outliers del Mar de Alborán

A raíz de esto, se realizaron las siguientes observaciones:

- La mayor concentración de outliers se encuentra entre Algeciras y Gibraltar, lo que probablemente se traduzca en la presencia de buques individuales fondeados a la espera de entrar en puerto.
- Por otro lado, teniendo en cuenta la Figura 3-25, los barcos detectados inmediatamente al este de Gibraltar, son también barcos fondeados que el algoritmo no es capaz de agrupar debido a que, al asumir densidades similares para los clústeres, no alcanzando el umbral en este caso.

Por tanto, en vista de los resultados obtenidos a lo largo de esta sección, se llegó a la conclusión de que la mejor combinación de parámetros en DBSCAN para discriminar los agrupamientos de manera precisa, para el conjunto de datos del que disponemos, es la siguiente:

MinPts = 10

$\epsilon = 1 / 3440.0695$

Métrica = distancia Haversine

3.3.2 HDBSCAN

Una vez estudiado el comportamiento de DBSCAN al procesar el conjunto de datos, se siguió explorando con otro algoritmo diferente para comparar sus resultados. Por ello, se tomó la decisión de implementar el algoritmo HDBSCAN (*Hierarchical DBSCAN*) (Anexo II: Código HDBSCAN), el cual aprovecha las ventajas de un algoritmo jerárquico, ejecutando DBSCAN para rango de valores para ϵ . El algoritmo integra los resultados para encontrar el agrupamiento que ofrezca mayor estabilidad, permitiendo encontrar grupos de densidades diferentes y aumentando la robustez frente a la selección de parámetros.

Cabe reseñar que, aunque fuera creado a su imagen y semejanza, este algoritmo no forma parte de la librería de Scikit-learn, por lo que, para instalarlo, se utilizó la terminal de Anaconda ejecutando el siguiente comando:

```
conda install -c conda-forge hdbscan
```

Para introducir el algoritmo a nuestro código, fue necesario importarlo de la siguiente manera:

```
from hdbscan import HDBSCAN
```

De la misma forma que con el algoritmo DBSCAN, se presentó el dilema de decidir sobre los parámetros del algoritmo. Aunque HDBSCAN puede ser configurado en base a un gran número de parámetros, en la práctica sólo son algunos los que de verdad tienen un efecto significativo en los resultados, por lo que fue necesario reflexionar sobre los mismos para hallar la combinación más favorable.

Sin embargo, antes de pasar a la selección de parámetros, es necesario introducir brevemente la noción de clúster dentro de HDBSCAN, para lo que se usarán ilustraciones extraídas de [68].

Para un conjunto de datos X , se supone que está formado por una mezcla de distribuciones normales que se pueden representar en función de la distribución de probabilidad de este conjunto de datos, como se muestra en la Figura 3-30.

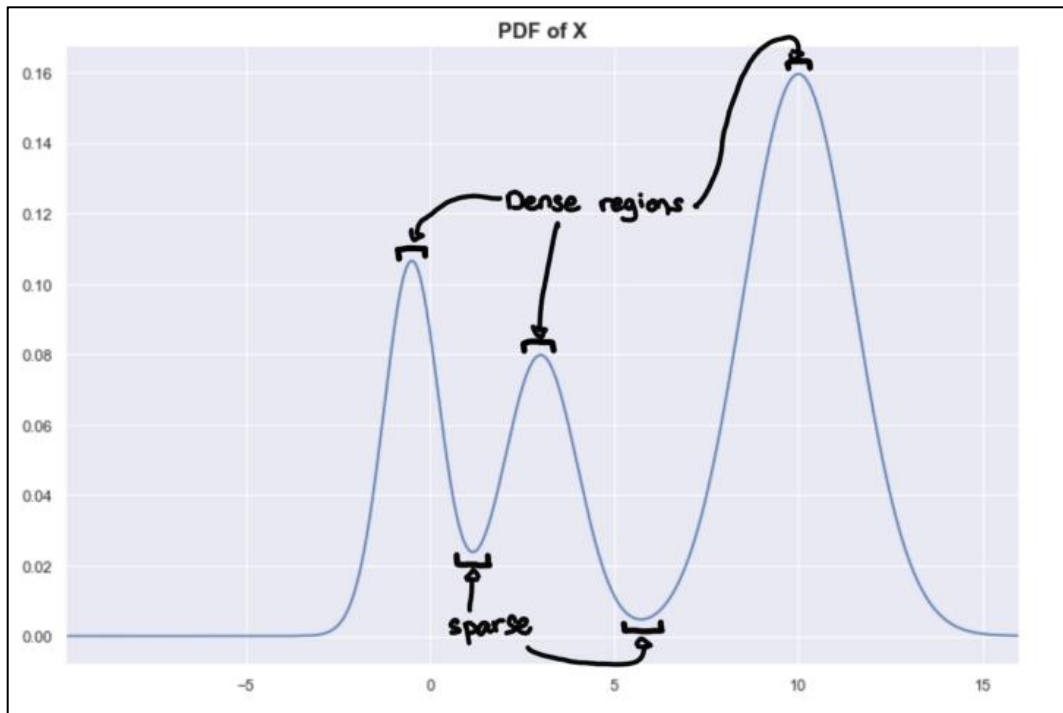


Figura 3-30 Distribución de la probabilidad de un conjunto de datos X [68]

Prestando atención a este último gráfico, se entiende de forma sencilla que el algoritmo HDBSCAN considera como grupos aquellas regiones con una alta probabilidad (o gran densidad), lo que se corresponde con los picos, separados por zonas más dispersas.

El problema aparece cuando dos picos están separados por un valle que posee también una densidad significativa, lo que hace dudar en el número de grupos a considerar. Es por ello que resulta de gran importancia el establecer un umbral global, que ayude a determinar el número de grupos (ver Figura 3-31).

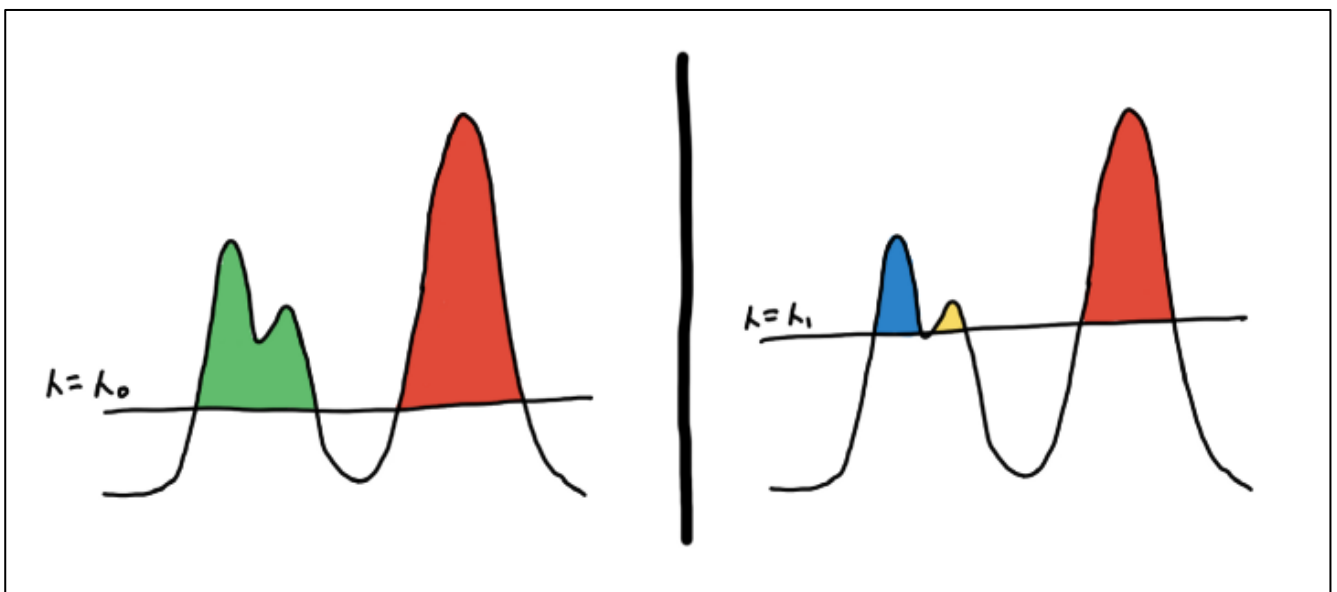


Figura 3-31 Representación del umbral global para un conjunto de datos arbitrario [68]

Resulta realmente atractiva la simplicidad aparente de este algoritmo, por la posibilidad de hacerlo funcionar seleccionando únicamente este parámetro; sin embargo, si se pretenden discriminar de forma eficaz agrupaciones de densidades diferentes, existe otro hiperparámetro que conviene ajustar para recuperar algunos grupos que se perderían.

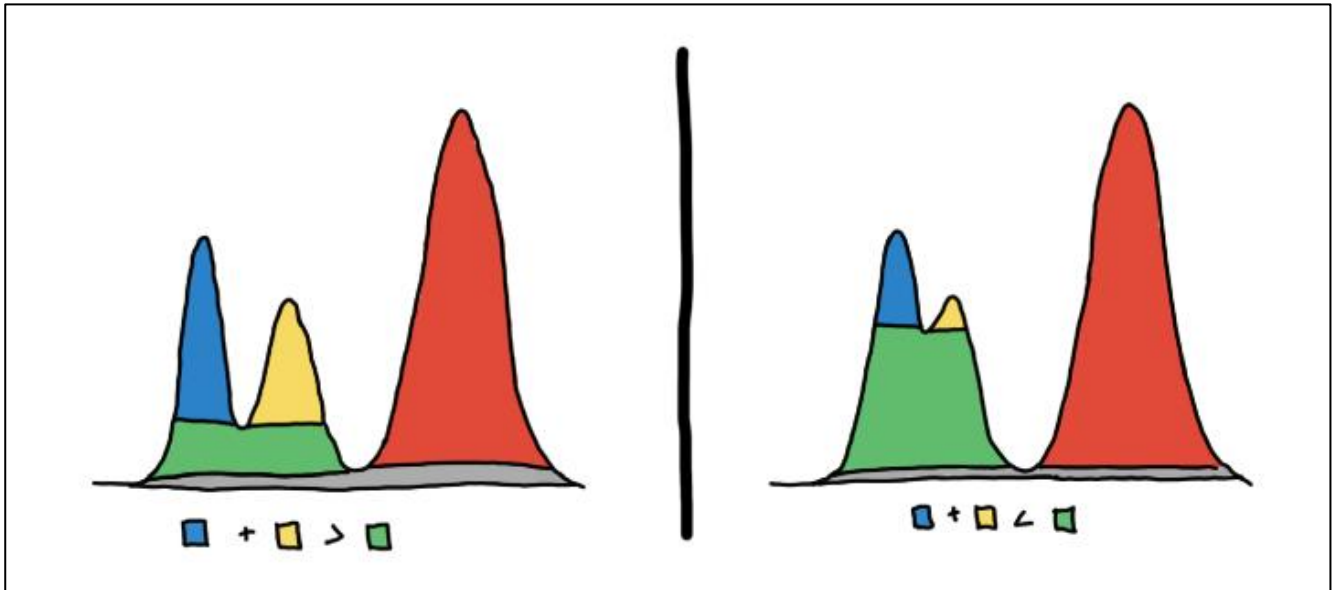


Figura 3-32 Representación del umbral secundario para dos conjuntos de datos arbitrarios [68]

En la Figura 3-32 se puede observar de forma intuitiva la repercusión del ajuste del hiperparámetro mencionado a la hora del agrupamiento. A la izquierda se ve que la suma de las áreas azul y amarilla es mayor que el área de color verde, lo que significa que los picos son más prominentes y que deben tratarse por separado. Por el contrario, a la derecha se observa como los picos son simples protuberancias dentro de un área más grande, por lo que deben ser considerados como parte del grupo verde.

El área de las regiones es una medida de la persistencia, y el método que el algoritmo utiliza para su evaluación, y que ejecuta por defecto, se llama “*eom*” (*Excess Of Mass*) y se encarga de encontrar la mayor estabilidad en la formación de clústeres.

Al obtener múltiples conjuntos de niveles de acuerdo con los diferentes valores del umbral secundario, se obtienen jerarquías (ver Figura 3-33). A medida que disminuye el umbral, esta jerarquía varía y los grupos comienzan a unirse, formando clústeres de mayor entidad.

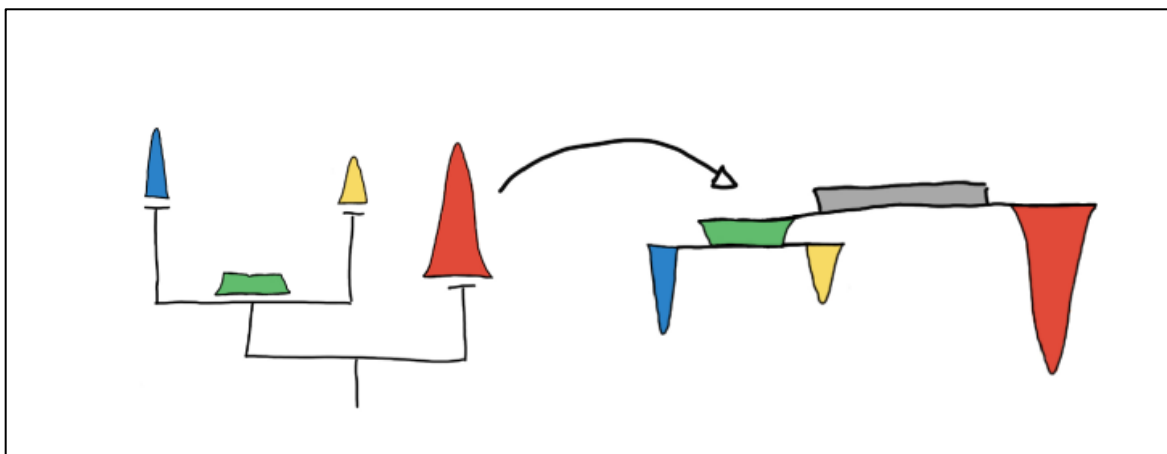


Figura 3-33 Árbol de jerarquías para un conjunto de datos arbitrario [68]

A la hora de utilizar HDBSCAN, se puede generar el árbol de jerarquías para un determinado valor del umbral, haciendo uso del API `clusterer.condensed_tree_.plot()`, el cual puede resultar útil para el ajuste de parámetros.

Antes de determinar sus parámetros principales, cabe mencionar que la métrica que se utilizó para HDBSCAN, guiándose por razonamientos ya presentados en la explicación del algoritmo DBSCAN, fue la distancia Haversine.

Minimum cluster size (*min_cluster_size*)

La idea más intuitiva para la selección de este parámetro es darle el valor del mínimo número de puntos que queremos que conformen un grupo. Para realizar la primera prueba, se escogió como valor *min_cluster_size* = 10. El resultado fue el siguiente:

```
Estimated number of clusters: 684
Estimated number of noise points: 3944
Silhouette Coefficient: 0.461
```

Es evidente que estos resultados no son buenos, pues los agrupamientos se consideran demasiado pequeños, de manera que no conseguirían determinar con claridad zonas reseñables.

La razón por la que esto ocurre son las protuberancias que se mencionan anteriormente, las cuales justifican la formación de un elevado número de grupos que deberían formar parte de uno superior. La manera de ponerle solución es aumentar el valor de *min_cluster_size* para aplanar dichas irregularidades en los picos, de manera que el método *eom* sea capaz de determinar el grupo principal, en lugar de un elevado número de pequeños grupos. Para comprobar la veracidad de esto, se comparan en la Figura 3-34 las diferencias entre los árboles jerárquicos para dos valores muy diferentes de este parámetro.

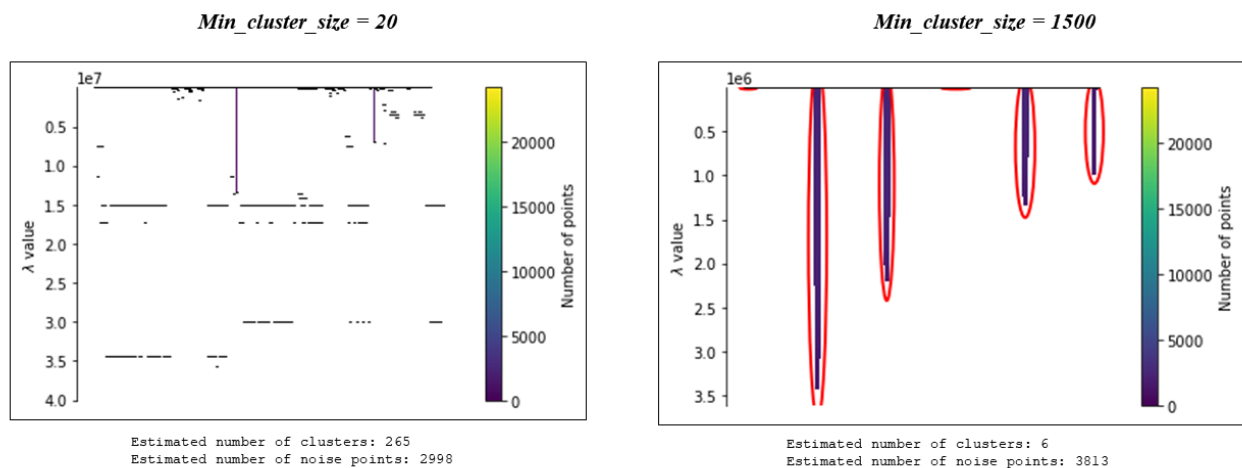


Figura 3-34 Comparativa árboles jerárquicos para distintos parámetros de *min_cluster_size*

Se observa que árbol de la izquierda no escala bien debido a su complejidad, y no se puede representar adecuadamente. Sin embargo, aumentando notablemente el valor de *min_cluster_size*, se pueden mostrar e identificar los grupos más relevantes, centrándose el algoritmo en estas estructuras.

Teniendo en cuenta lo dispuesto, se recopilamos los valores que ofrecían otras configuraciones, obteniendo los resultados presentados en la Tabla 3-3:

min_cluster_size	Grupos	Outliers	C. Silueta
10	684	3944	0,461
20	265	2998	0,596
40	92	1936	0,719
60	60	1105	0,882
80	51	957	0,895
100	45	921	0,897
120	40	716	0,921
140	37	1095	0,893
160	34	1158	0,881
180	32	1321	0,871

Tabla 3-3 Parámetros de validación para diferentes valores de *min_cluster_size*

En vista de los datos obtenidos, se decide seleccionar *min_cluster_size* = 120 para continuar el estudio, pues los valores para el número de grupos y *outliers* es razonable, y el coeficiente de silueta es bueno. El resultado que se consigue se presenta en la Figura 3-35.

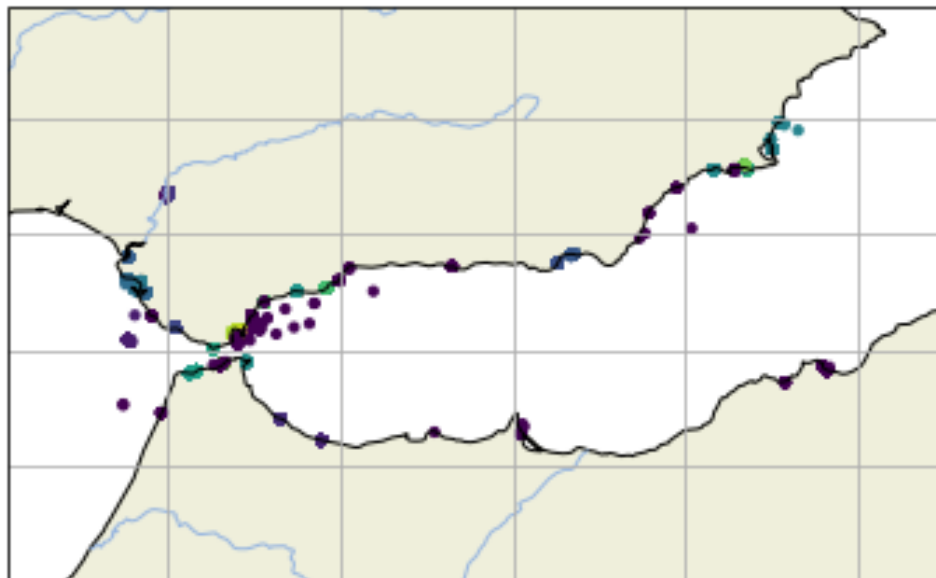


Figura 3-35 Plot para *min_cluster_size* = 120

```
Estimated number of clusters: 40
Estimated number of noise points: 716
Silhouette Coefficient: 0.921
```

A primera vista, y teniendo en cuenta los resultados obtenidos con DBSCAN, llama la atención la presencia de grupos diferentes en Algeciras, por lo que se procedió a aumentar la escala en esta zona (Figura 3-36)

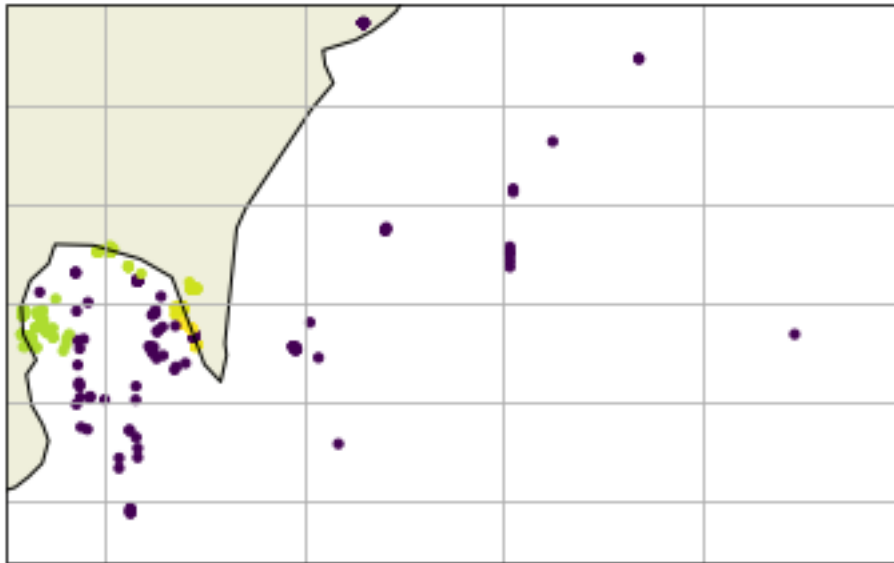


Figura 3-36 Plot de Algeciras, *min_cluster_size* = 120

Se comprueba que en la bahía de Algeciras se discriminan tres grupos diferentes de barcos, dejando una gran cantidad de los mismos como *outliers*.

También se comprobó el puerto de Cádiz por la cantidad de grupos que debería presentar.

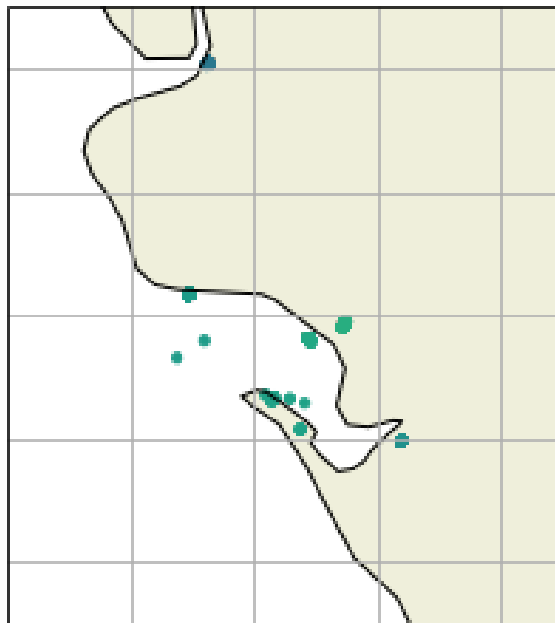


Figura 3-37 Plot de la Bahía de Cádiz, *min_cluster_size* = 120

En este caso, barcos que deben pertenecer a grupos diferentes, considerando que se puede lograr una mayor precisión, se encuentran agrupados bajo el mismo conjunto.

Los resultados que se presentan en la Figura 3-36 motivan a aumentar el valor de *min_cluster_size* para encontrar grupos más grandes, pero en la Figura 3-37 se observa la necesidad del efecto contrario, pues interesaría separar clústeres. Es por ello que, manteniendo el valor de *min_cluster_size*, se modificó el valor de *min_samples*.

Parámetro *min_samples*

Este parámetro, si no se especifica, asume el mismo valor de “*min_cluster_size*” por defecto. Es el número de valores en un vecindario para que un punto pueda considerarse “*core point*”. La aproximación que mejor define el efecto de este parámetro sobre los resultados es que determina cuan conservativo se es a la hora de agrupar. Así, un valor grande significa que el algoritmo se comportará de manera más conservativa, declarando más puntos como ruido y determinando áreas más densas.

En búsqueda de resultados más adecuados, se dieron distintos valores al parámetro “*min_samples*” manteniendo “*min_cluster_size*” =120, y se recogieron los datos obtenidos en la Tabla 3-4:

min_samples	Grupos	Outliers	C. Silueta
10	40	616	0,921
20	40	542	0,927
40	40	598	0,925
60	40	634	0,924
80	40	688	0,922
100	40	636	0,924
120	40	697	0,922
140	37	1079	0,893
160	34	1108	0,883

Tabla 3-4 Parámetros de validación para diferentes valores de *min_samples*

Como aumentar el parámetro *min_samples* suaviza los resultados por ser más conservativo a la hora de agrupar, se probó a disminuir su valor considerablemente. Así, para un valor igual a 5, los resultados fueron los siguientes:

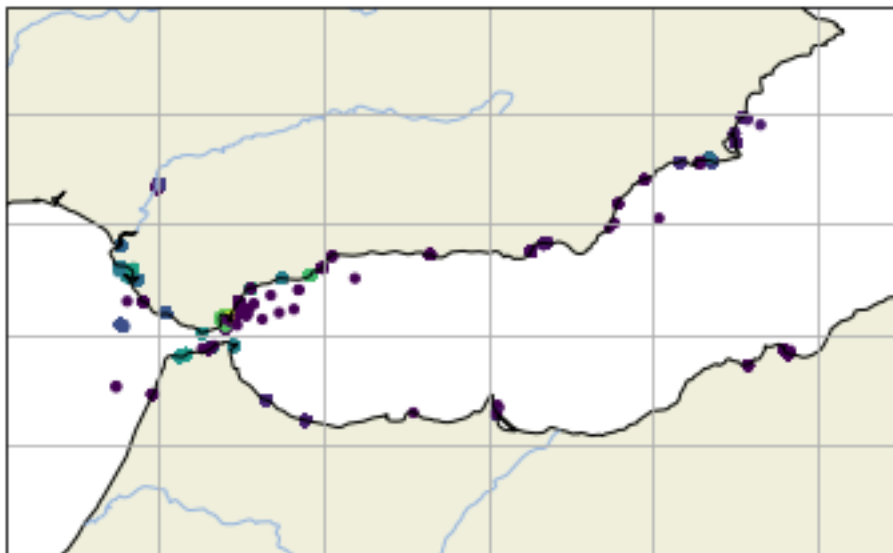


Figura 3-38 Plot para *min_cluster_size* = 120 y *min_samples* = 5

Estimated number of clusters: 46
 Estimated number of noise points: 631
 Silhouette Coefficient: 0.838

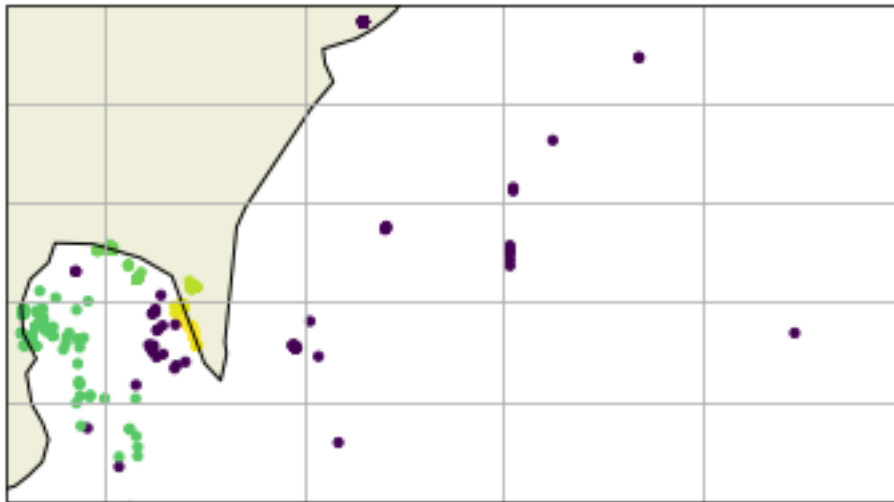


Figura 3-39 Algeciras - $min_cluster_size = 120$ y $min_samples = 5$

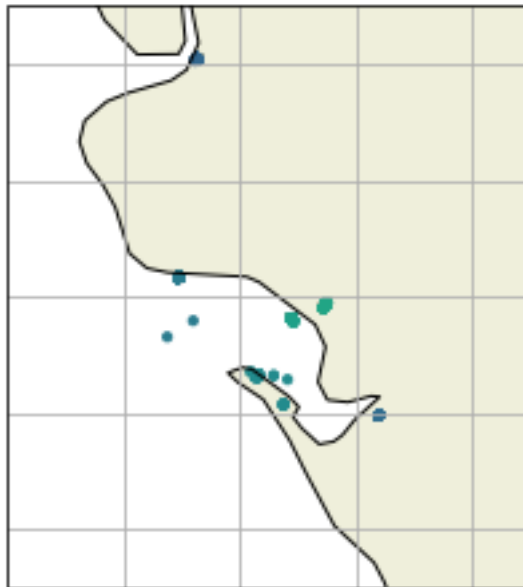


Figura 3-40 Cádiz - $min_cluster_size = 120$ y $min_samples = 5$

En cuanto a números, los resultados son similares y, aunque ayude a separar el grupo de Cádiz (Figura 3-40) y Algeciras (Figura 3-39), no se consigue una buena discriminación en otros puertos, como es el caso de Melilla, Orán o la zona de Murcia. Esto se debe a que los grupos pequeños se encuentran alejados de los núcleos principales de densidad.

Se puede comprobar que el algoritmo HDBSCAN resulta algo menos intuitivo a la hora de configurar, sin ofrecer, además, mejores resultados. Al aumentar $min_cluster_size$ y disminuir el parámetro $min_samples$, se ha perdido un número importante de clústeres pequeños en zonas de menor densidad. Sin embargo, existe un parámetro que puede ayudar con este problema, $cluster_selection_epsilon$.

Parámetro “*cluster_selection_epsilon*”

El método de extracción de grupos haciendo uso de este parámetro actúa como un híbrido entre DBSCAN y HDBSCAN, devolviendo resultados DBSCAN para los afectados por el valor del parámetro, pero resultados HDBSCAN para el resto.

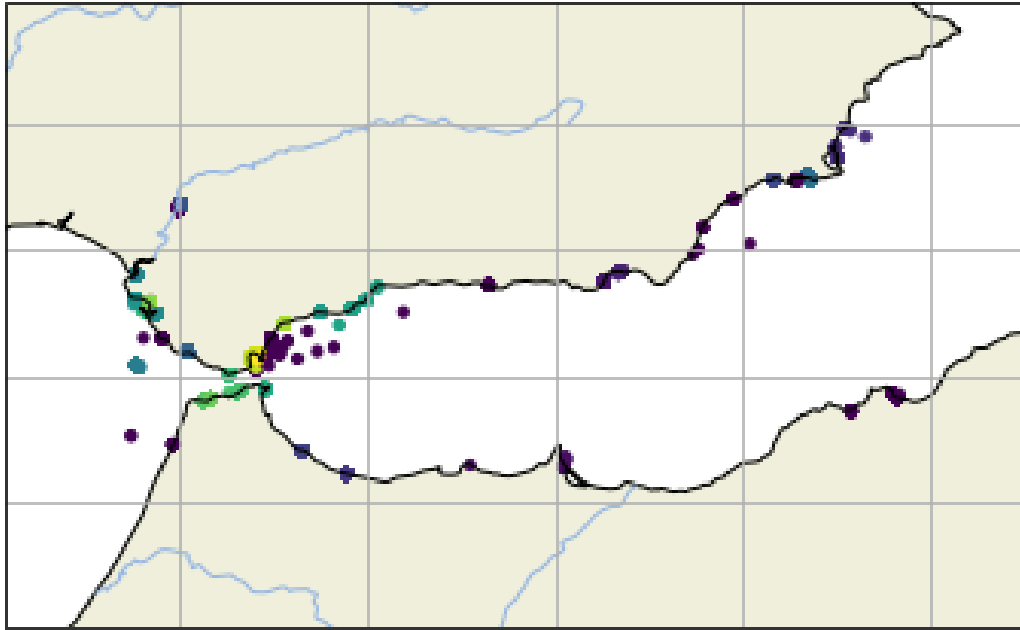


Figura 3-41 Plot *min_cluster_size* = 120, *min_samples* = 5 y *cluster_selection_epsilon* = 1km

```
Estimated number of clusters: 31
Estimated number of noise points: 345
Silhouette Coefficient: 0.935
```

Comparando la Figura 3-41 con la Figura 3-38, se puede ver que ha disminuido el número de clústeres y *outliers*, se han logrado diferenciar en otros puertos agrupamientos reseñables, como es el caso de Tánger, y se discriminan mejor aquellas zonas de escala más reducida, como el puerto de Cádiz, que ahora cuenta con varios grupos diferenciados.

En vista de los resultados obtenidos a lo largo de toda la sección de HDBSCAN, se ha comprobado que, aun siendo similar en naturaleza a DBSCAN, ofrece resultados muy diferentes. Atendiendo a los resultados, se concluye que, para el conjunto de datos estudiado, resulta de mayor utilidad el uso de DBSCAN.

3.3.3 Consideraciones y modificaciones al del dataset

Hasta este punto de la memoria, se ha presentado el efecto de dos algoritmos íntimamente relacionados entre sí, los cuales de forma clara producen resultados muy diferentes. Esto se debe, entre otras cosas, al conjunto de datos estudiado, lo cual lleva a preguntarse cómo actuarían sobre otro *dataset*.

El conjunto de datos utilizado hasta este momento era un extracto de una base de mensajes AIS más grande, que había sido recortada y filtrada por su campo “*sog* = 0”. Sin embargo, tal como se expone en [69], la información transmitida en los mensajes AIS está sujeta a números factores humanos y físicos que pueden llevar a error. Es por ello que se puede considerar como únicos campos fiables del mensaje los que transmiten latitud y longitud geográficas.

Con el objetivo de analizar la base de datos que se pretendía estudiar, se elaboró el siguiente diagrama de caja y bigotes para el campo “sog”.

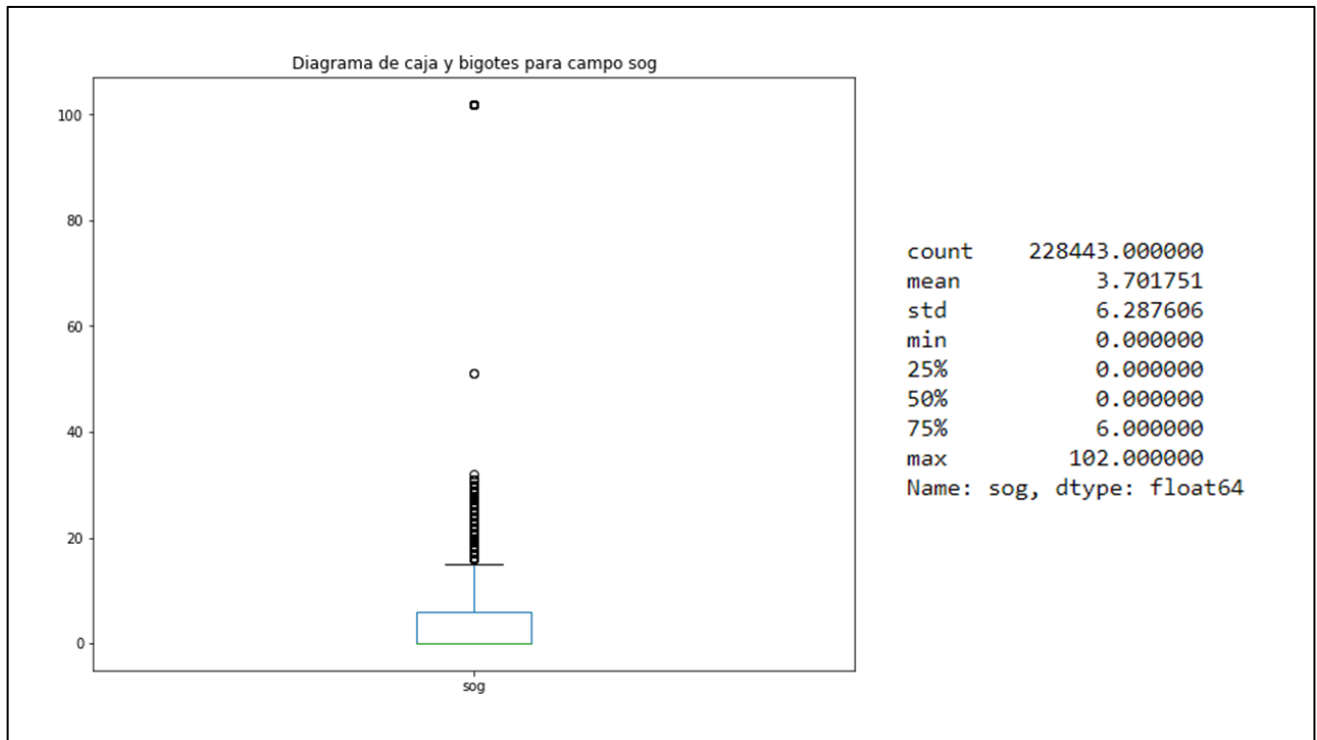


Figura 3-42 Box plot del campo “sog”

En la Figura 3-42 se comprueba la existencia de valores anómalos dentro del campo “sog” de los mensajes AIS del *dataset*, con valores que superan incluso los 100 nudos, algo que resulta incongruente. En el diagrama, la línea verde hace referencia a la mediana del conjunto, la línea superior de la caja representa el tercer cuartil, y la línea paralela a la caja marca valor máximo de la distribución. En este diagrama no se aprecian el valor del primer cuartil (línea inferior de la caja) ni el valor mínimo de la distribución, pero coinciden con la mediana. Sin embargo, lo que realmente nos interesa es la presencia de *outliers* en la distribución, los cuales aparecen representados por círculos que se encuentran por encima del extremo superior (o debajo del extremo inferior, si se diera el caso) en función de su valor.

A parte de los *outliers*, se consideraba que la precisión del campo “sog” podía no ser correcto o suficientemente preciso, por lo que se tomó la decisión de escribir un código (Anexo III: Cálculo de la velocidad real de los buques) que permitiera calcular la velocidad real de los buques de un *dataset* de mensajes AIS. Para un conjunto de datos de aproximadamente 230.000 entradas, se añadió al *dataframe* inicial una columna con estos datos bajo el nombre de “speed”, como se muestra en la Figura 3-43.

```
In [74]: X.sample(10)
```

```
Out[74]:
```

	mmsi	time	lon	lat	sog	speed
116318	224906140	2019-12-11 18:25:46.998	-5.391165	36.175705	5	8.298045
78796	224241390	2019-12-12 15:42:34.602	-4.617030	36.539082	0	0.013670
151176	236112261	2019-12-12 19:15:59.535	-5.360532	36.139866	0	0.009011
105475	224646000	2019-12-11 19:17:37.483	-5.387600	36.094551	13	7.863601
107847	224704000	2019-12-11 09:12:26.665	-5.370957	36.062881	22	32.125651
205668	305342000	2019-12-11 07:17:47.32	-5.611612	35.920948	13	13.516931
112703	224784000	2019-12-12 16:02:47.293	-5.439040	36.133038	0	0.028353
18584	210754000	2019-12-12 15:50:29.253	-4.580833	36.300552	9	8.450019
21782	211553000	2019-12-12 18:22:08.112	-5.199332	36.173168	4	7.531144
51081	224068550	2019-12-12 23:38:29.834	-4.618323	36.539608	0	0.017783

Figura 3-43 *Dataframe* incluyendo los valores de velocidad calculados

Para una muestra aleatoria de 10 unidades, se puede observar que, si bien hay valores que realmente se aproximan al valor real, otros difieren de forma considerable. Para permitir una vista general de las velocidades calculadas, se elaboró un diagrama de dispersión de las columnas *speed* y *sog*.

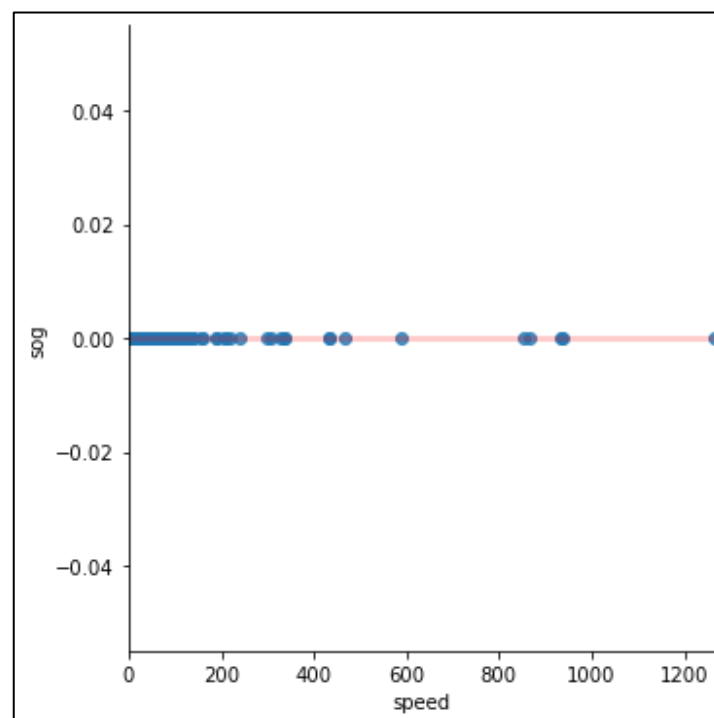


Figura 3-44 Diagrama de dispersión "sog" vs "speed"

En la Figura 3-44 se puede ver como el cálculo de las velocidades reales ha dejado valores que no sólo son positivos, sino que adquieren números muy elevados. Aquí se abre la posibilidad de que varios barcos hayan emitido sus mensajes AIS bajo el mismo número de identificación, error que puede haber sido cometido por equivocación, o intencionadamente para ocultar una identidad.

La comprobación de que el campo de la velocidad no ofrecía datos completamente veraces, llevó a pensar que los resultados obtenidos en apartados anteriores quizás no fueran precisos. Además, el hecho

de que un buque parado emita mensajes AIS en un intervalo aproximado de dos minutos puede provocar un aumento de densidad local y la detección de clústeres que engloban, en realidad, únicamente ese barco.

En consecuencia, se desarrolló un código que rectificara la base de datos utilizada (ver Anexo IV: Filtro para extraer buques parados) bajo las siguientes consideraciones

- Los buques parados emitirán mensajes en un intervalo aproximado de dos minutos, pudiendo formar un grupo por sí mismos si permanecen en la misma área durante un periodo prolongado, por lo que será necesario eliminar esas posiciones “redundantes”.
- Los buques en la mar se encuentran a merced de las corrientes y el viento, por lo que todo buque con una velocidad inferior a un nudo se considerará parado.
- Los mensajes emitidos por el mismo barco al mismo tiempo deben ser eliminados (duplicidad).
- Se descartará el primer mensaje de cada barco por la imposibilidad de calcular su velocidad respecto a una posición anterior.

3.3.3.1 DBSCAN – Dataset de mayor tamaño

Finalmente, tras el estudio del comportamiento de los algoritmos y sus parámetros, y después de desarrollar un código capaz de filtrar una base de datos para extraer buques parados de forma eficaz, de manera que se pudieran identificar zonas de parada o inicios/finales de ruta, se procedió a analizar un *dataset* de un tamaño diez veces superior, considerando que sería una aproximación más cercana a la realidad. Tras procesar la base de datos, el número de entradas se redujo a 5806.

El *dataset* al que se hace referencia contiene datos AIS de barcos que emitieron sus mensajes entre los días 11 y 12 de diciembre de 2019, en el área geográfica del Estrecho de Gibraltar.

Comenzando con DBSCAN, resulta evidente que, al trabajar con una base de datos de mayores proporciones, será necesario elevar el valor de MinPts, pues en una zona geográfica habrá una concentración de barcos más grande en un intervalo de tiempo. También se establecerá un mayor valor para ϵ , pues interesa encontrar agrupamientos de mayor tamaño, con el fin de detectar áreas relevantes en una escala suficientemente grande como para conseguir una visión general.

Por ello, realizando pruebas de forma iterativa, se consiguieron localizar agrupamientos importantes estableciendo MinPts = 50 y ϵ = 1 km para el algoritmo DBSCAN, utilizando métrica Haversine.

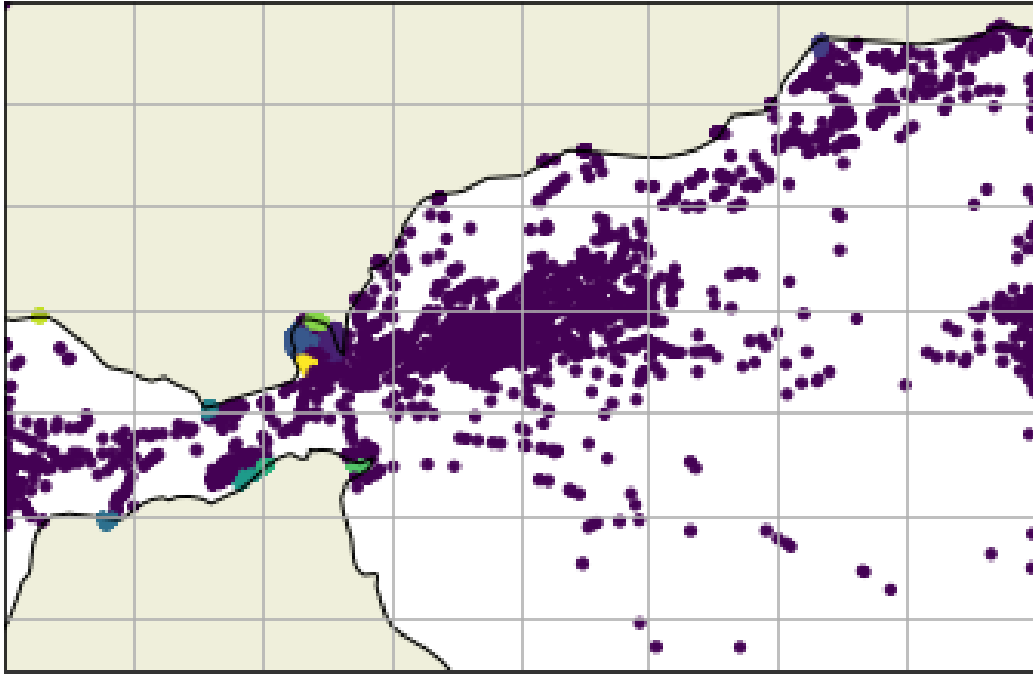


Figura 3-45 Zonas de parada usando el algoritmo DBSCAN - MinPts = 50 y $\varepsilon = 1\text{km}$

Estimated number of clusters: 11
Estimated number of noise points: 2611
Silhouette Coefficient: 0.177

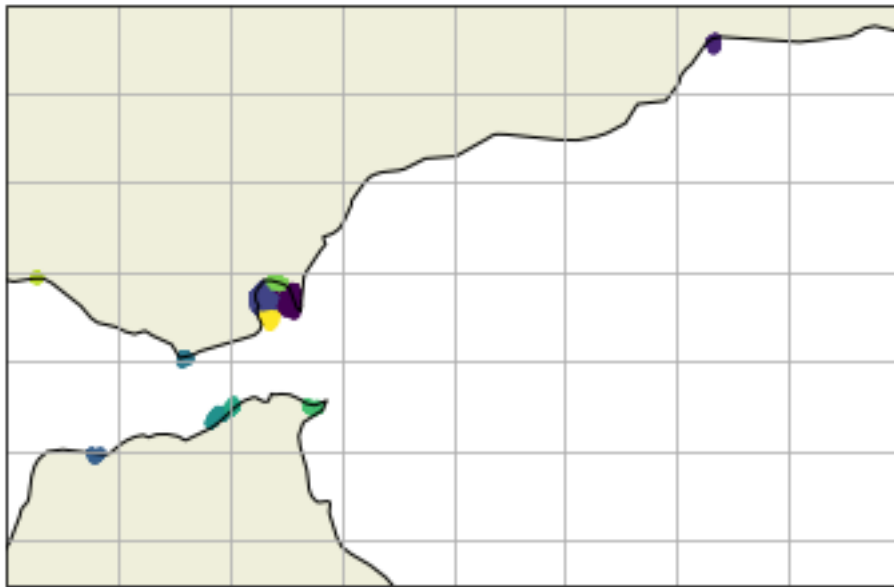


Figura 3-46 Zonas de parada, algoritmo DBSCAN (Sin outliers) - MinPts = 50 y $\varepsilon = 1\text{km}$

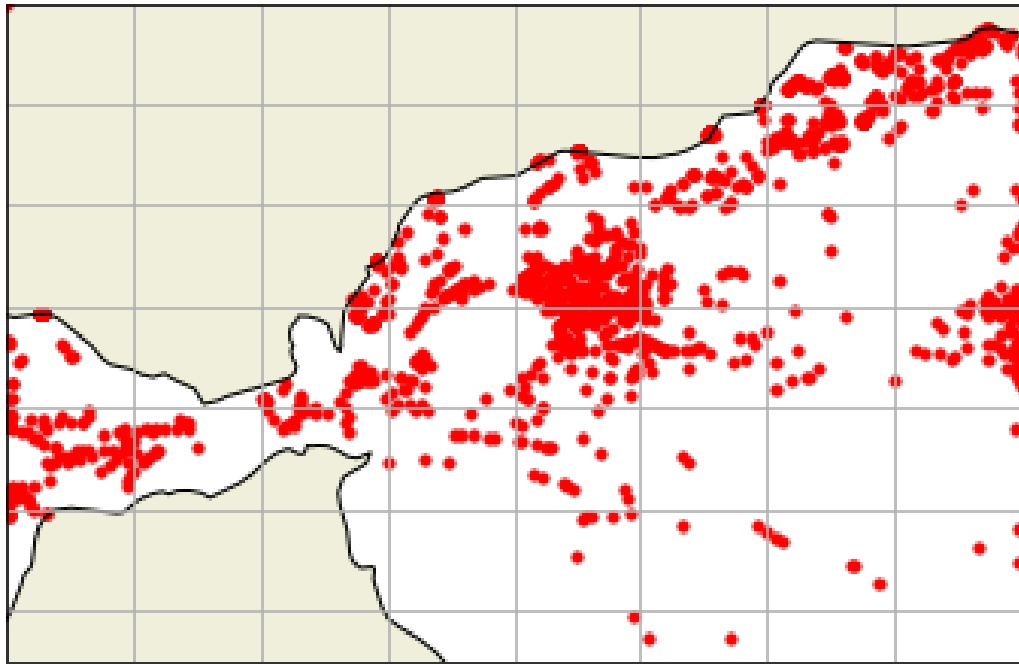


Figura 3-47 Outliers resultantes de la clusterización con DBSCAN – MinPts = 50 y $\epsilon = 1\text{km}$

Los resultados presentados en la Figura 3-45 y la Figura 3-47 poseen un valor importante, pues al contrastarlas con cartas náuticas y mapas satelitales se pueden extraer conclusiones relevantes en materia de seguridad marítima y monitorización en la zona del Estrecho de Gibraltar. Por ello, se extrajeron los siguientes mapas de Google Maps y OpenCPN.

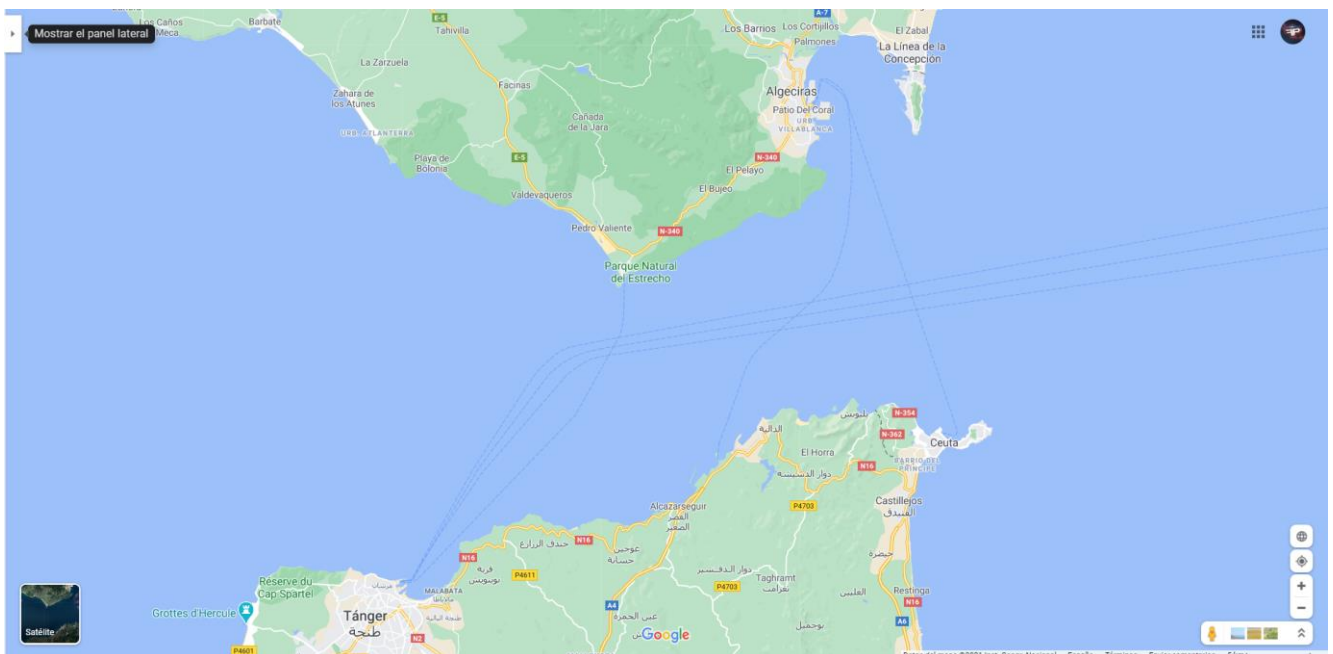


Figura 3-48 Rutas marítimas habituales en el Estrecho de Gibraltar, en Google Maps

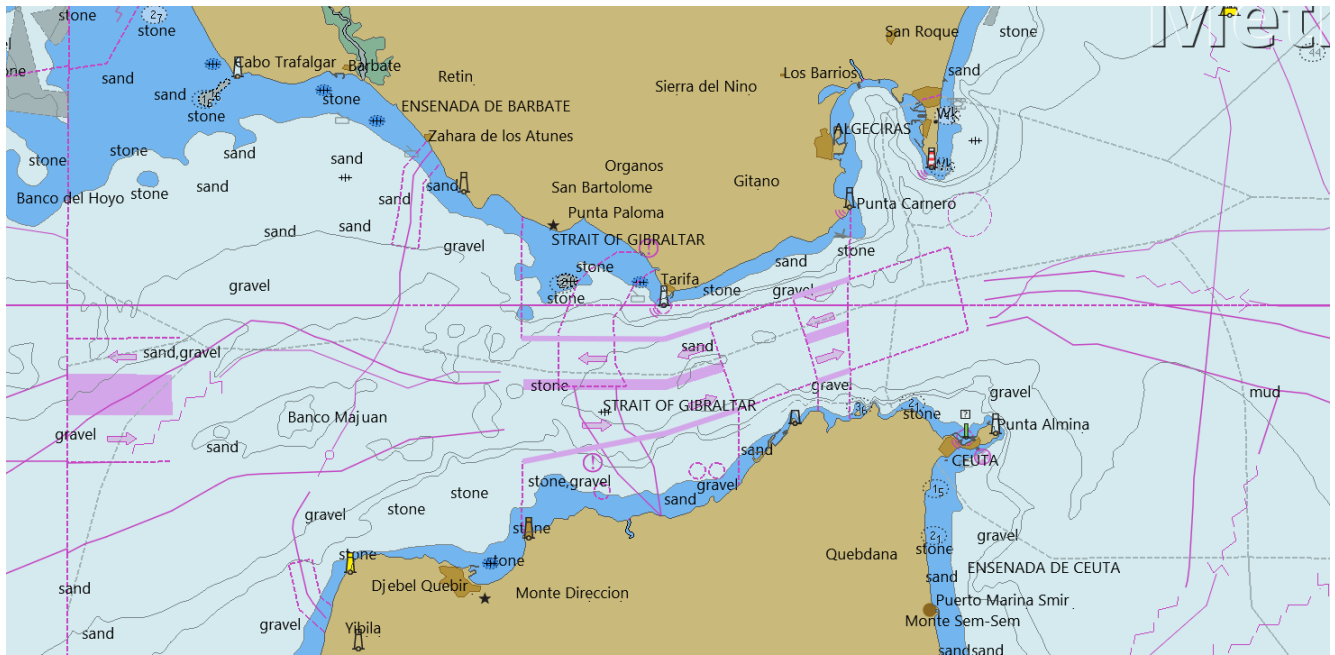


Figura 3-49 Recorte extraído de una carta náutica digital en la zona del Estrecho de Gibraltar

Contrastando los resultados obtenidos con la información que ofrecen la Figura 3-48 y la Figura 3-49, se observa que el algoritmo ha sido capaz de localizar puntos de inicio/final de rutas marítimas habituales en la zona estudiada, así como otras zonas de interés donde existe una gran acumulación de posiciones de barcos. En concreto, se realiza el siguiente análisis:

- Se pueden localizar los extremos de la ruta marítima Algeciras – Ceuta.
- Se pueden localizar los extremos de la ruta marítima Algeciras– Tánger Med.
- Se pueden localizar los extremos de la ruta marítima Tarifa– Tánger.
- Se pueden localizar los extremos de la ruta marítima Gibraltar– Tánger Med.
- Se puede localizar un grupo en el puerto de Málaga.
- Se puede localizar un grupo en el puerto de Barbate.
- Se localiza un fondeadero a la entrada de la bahía de Algeciras (clúster amarillo).

Esta configuración ofrece un buen rendimiento a la hora de localizar los puertos más importantes de la zona, diferenciando incluso los distintos muelles que los componen. Esto se puede apreciar en la Figura 3-50.

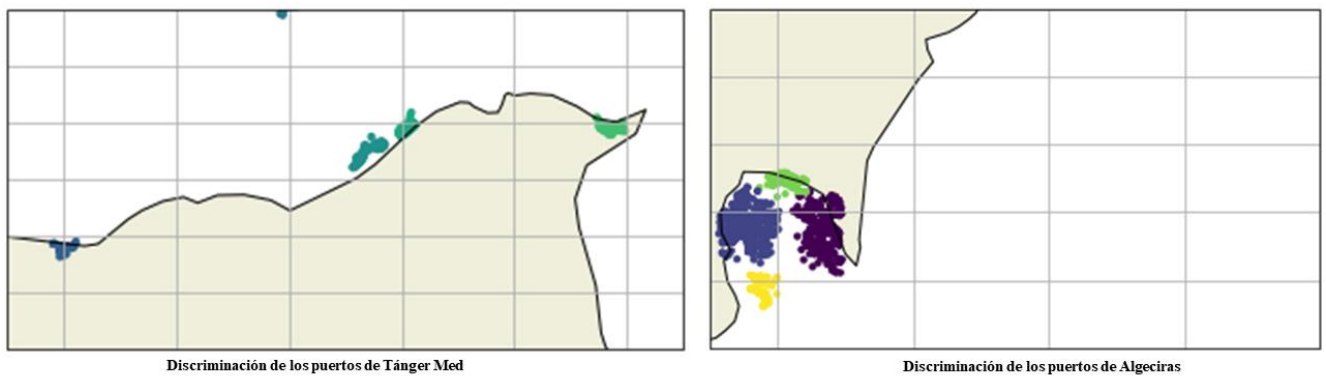


Figura 3-50 Discriminación de los muelles de puertos importantes en el Estrecho de Gibraltar

Esta configuración ofrece unos resultados satisfactorios, sin embargo, aumentando el valor de ϵ sería posible localizar también otro tipo de zonas, como se muestra en la Figura 3-51.

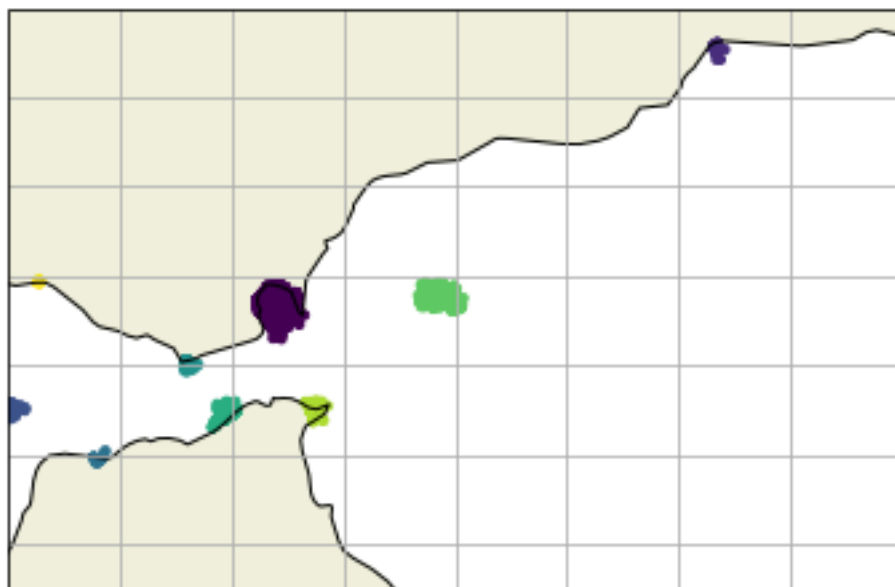


Figura 3-51 Zonas de parada utilizando el algoritmo DBSCAN (Sin outliers) - MinPts = 50 y $\epsilon = 2\text{km}$

Es fácil observar que los grupos que antes estaban separados en Tánger Med y Algeciras ahora forman dos únicos clústeres. Además, se han localizado dos grupos nuevos:

- Al oeste del Estrecho de Gibraltar, un clúster azul oscuro que corresponde con una zona de separación de tráfico a la entrada del Mar Mediterráneo. Probablemente se trate un grupo de barcos pesqueros en la zona que separa los dos carriles.
- Al este de Gibraltar, un grupo de color verde de gran extensión, que coincide con una zona habitual donde buques petroleros realizan prácticas de *bunkering*.

La imposibilidad de obtener una mezcla de ambos resultados, localizando estos últimos grupos y discriminando los distintos puertos del Estrecho se debe a que el algoritmo DBSCAN no puede localizar clústeres con densidades distintas, siendo evidente que un puerto tendrá una mayor densidad que una zona de fondeo en la mar. Para intentar solventar este problema, se procesó esta base de datos con el algoritmo HDBSCAN.

3.3.3.2 HDBSCAN - Dataset de mayor tamaño

Siguiendo la línea del anterior apartado, se modificó el valor de los parámetros para obtener buenos resultados, al existir mayores concentraciones de puntos en la zona del Estrecho de Gibraltar. Por tanto, estableciendo unos valores de $min_cluster_size = 110$ y $min_samples = 105$, se alcanzaron los siguientes resultados.

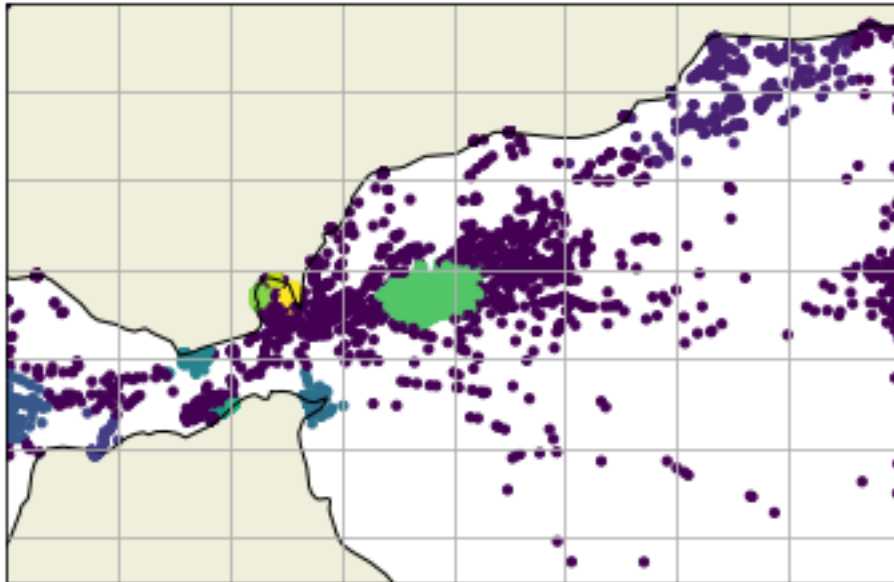


Figura 3-52 Zonas de parada utilizando el algoritmo HDBSCAN - $min_cluster_size = 110$ y $min_samples = 105$

```
Estimated number of clusters: 11  
Estimated number of noise points: 1750  
Silhouette Coefficient: 0.320
```

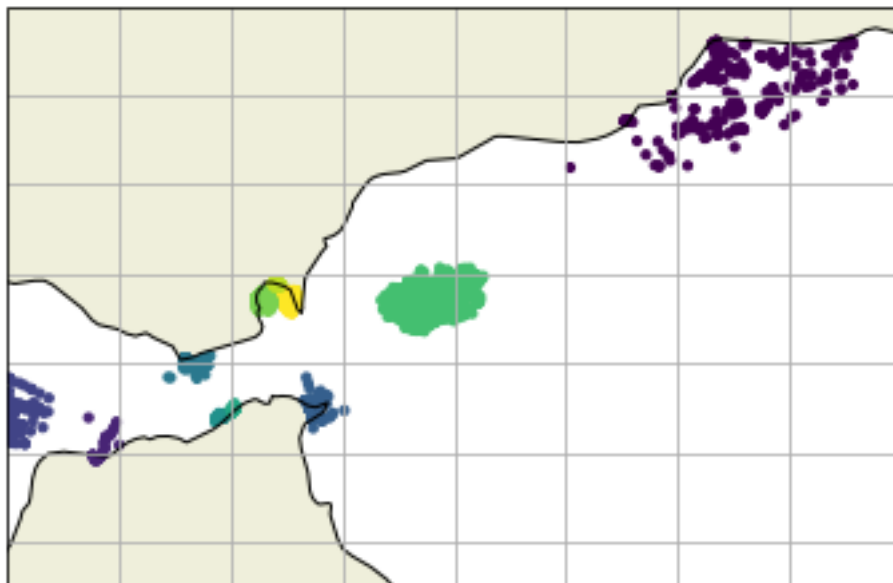


Figura 3-53 Zonas de parada utilizando el algoritmo HDBSCAN (Sin outliers) - $min_cluster_size = 110$ y $min_samples = 105$

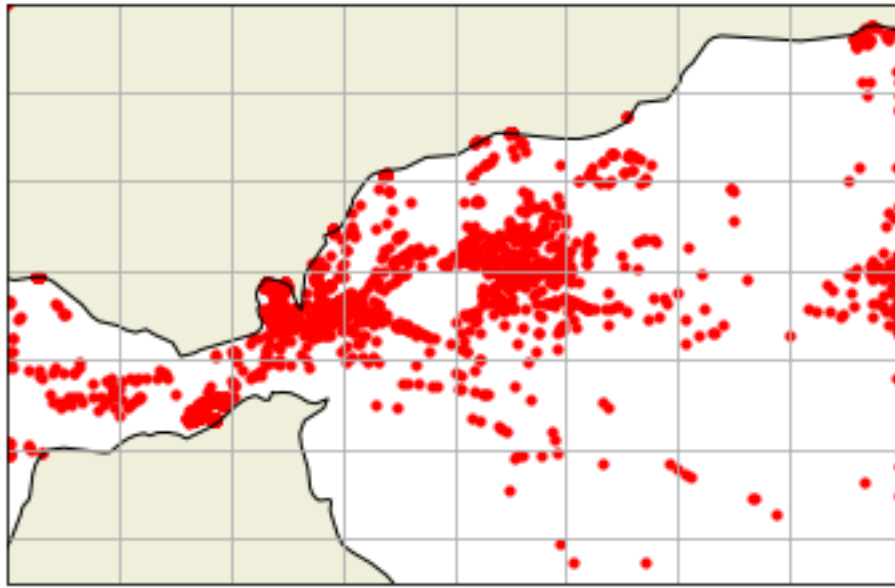


Figura 3-54 Outliers resultantes de la clusterización con DBSCAN – $min_cluster_size = 110$ y $min_samples = 105$

Gracias a las Figura 3-52, la Figura 3-53 y la Figura 3-54, se puede comprobar que el algoritmo HDBSCAN bien configurado puede de ofrecer unos resultados mejores que DBSCAN, permitiendo localizar grupos de barcos parados en la mar, a la vez que discrimina entre grupos centrados en puertos próximos entre sí, gracias a su capacidad de detectar grupos de distintas densidades.

A la vista de los resultados, y teniendo en cuenta todo lo dispuesto a lo largo del capítulo, se concluye que el algoritmo HDBSCAN, proporciona unos resultados más precisos que el algoritmo DBSCAN para la tarea que nos ocupa, además de ofrecer más información respecto a la distribución de los barcos a lo largo de la zona.

4 CONCLUSIONES Y LÍNEAS FUTURAS

4.1 Conclusiones

A lo largo del desarrollo de este proyecto se revisó el Estado del arte del análisis de clusterización, se seleccionaron los algoritmos DBSCAN y HDBSCAN para ser implementados, y se pusieron en funcionamiento utilizando una base de datos compuesta por mensajes AIS. El Desarrollo del TFG consistió en evaluar los resultados que ofrecían estos algoritmos para distintos valores de los parámetros, con el objetivo de encontrar una combinación que permitiera discriminar eficazmente puntos de inicio/final de rutas marítimas y zonas de parada habituales. Ambos algoritmos fueron capaces de localizar grupos con la suficiente precisión como para discriminar distintas zonas incluso dentro de un mismo puerto, sin generar un gran número de *outliers*. Sin embargo, a raíz de los resultados obtenidos, se concluyó que el algoritmo HDBSCAN ofrecería resultados de mayor valor para el propósito del proyecto por los motivos expuestos en el capítulo anterior, si bien cabe mencionar que los resultados obtenidos con el algoritmo DBSCAN no resultaban insatisfactorios.

Por tanto, como se comentaba al inicio de este estudio, se confirma que no existe un algoritmo mejor que otro, sino que será necesario explorar, para cada caso específico, el abanico de posibilidades que existen para encontrar el más adecuado. Será de especial relevancia conocer profundamente el conjunto de datos que se pretende estudiar, y mantener vivo el espíritu explorador.

De esta manera, concluye el trabajo habiendo cumplido el objetivo principal del mismo, llevando a cabo el análisis de dos sistemas de aprendizaje no supervisado, DBSCAN y HDBSCAN, logrando encontrar información de valor dentro de una base de datos relativamente grande. Trabajando de forma heurística, se logró satisfacer el propósito de encontrar puntos de inicio y fin de rutas marítimas habituales, así como otras zonas de parada dentro del área geográfica del Estrecho de Gibraltar.

Para finalizar, resaltar que, para alcanzar el objetivo principal, se cumplió la premisa de satisfacer también los objetivos específicos y complementarios que se habían planteado. Para el éxito del proyecto, se llevó a cabo una revisión del estado del arte y trabajos relacionados que posibilitó la selección de los algoritmos a utilizar, así como la familiarización con herramientas de *Machine Learning* e inteligencia artificial empleando el lenguaje de programación Python.

4.2 Líneas futuras

Otra conclusión que se saca de la realización de este proyecto es la infinidad de posibilidades que abarca el mundo de la inteligencia artificial, y el potencial que tiene en el ámbito de la seguridad

marítima. Es por ello que se proponen las siguientes líneas futuras, como propuestas para la ampliación de este proyecto y el desarrollo de trabajos relacionados:

- Implementación de otros algoritmos basados en densidad para la detección de rutas marítimas habituales, como pueden ser OPTICS o DENCLUE.
- Implementación de algoritmos *grid-based* para la detección de rutas marítimas habituales, como puede ser el algoritmo CLIQUE.
- Desarrollo de una plataforma de clusterización en tiempo real utilizando algoritmos basados en densidad.
- Implementación de los algoritmos presentados para la detección de puntos de caída o *waypoints* que definan rutas marítimas, para complementar los resultados obtenidos en este trabajo.
- Análisis del comportamiento de la flota pesquera filtrando la base de datos por el tipo de barco que indican los mensajes AIS, con objeto de detectar comportamientos anómalos.
- Aplicación de los algoritmos presentados para la clusterización de rutas previamente particionadas, siguiendo la línea de otros trabajos ya existentes.

5 BIBLIOGRAFÍA

- [1] C. d. España, «Cámara de Comercio de España - ¿Qué es la digitalización?,» [En línea]. Available: <https://www.camara.es/blog/innovacion-y-competitividad/que-es-digitalizacion>. [Último acceso: 22 febrero 2021].
- [2] C. Europea, «Shaping Europe's digital future,» [En línea]. Available: <https://ec.europa.eu/digital-single-market/en/scoreboard/spain>. [Último acceso: 22 febrero 2021].
- [3] C. Europea, «Índice de la Economía y la Sociedad Digitales (DESI) 2020,» 2020.
- [4] G. Novatech, «Las 10 V's del Big Data,» [En línea]. Available: <https://www.grupo-novatech.com/las-10-vs-del-big-data/>. [Último acceso: 21 enero 2021].
- [5] «¿Qué es el Big Data?,» [En línea]. Available: <https://www.oracle.com/es/big-data/what-is-big-data.html#link6>. [Último acceso: 21 enero 2021].
- [6] T. D. J. A. C. Ruiz, «Big Data en los entornos de Seguridad y Defensa,» [En línea]. Available: http://www.ieee.es/Galerias/fichero/docs_investig/DIEEEINV03-2013_Big_Data_Entornos_DefensaSeguridad_CarrilloRuiz.pdf. [Último acceso: 20 enero 2021].
- [7] M.-E. V. ,. F. O. J. A. Ioanna Lytra, «A Big Data Architecture for Managing Oceans of Data and Maritime Applications,» [En línea]. Available: https://www.researchgate.net/publication/322997369_A_big_data_architecture_for_managing_oceans_of_data_and_maritime_applications. [Último acceso: 23 febrero 2021].
- [8] «Big Data. ¿Qué es y cómo funciona?,» [En línea]. Available: <http://blog.masterbase.com/big-data-que-es-y-como-funciona>. [Último acceso: 21 enero 2021].
- [9] «IDC's Global DataSphere Forecast Shows Continued Steady Growth in the Creation and Consumption of Data,» [En línea]. Available: <https://www.idc.com/getdoc.jsp?containerId=prUS46286020>. [Último acceso: 21 enero 2021].

-
- [10] SAS, «Análítica del big data: Qué es y por qué es importante,» [En línea]. Available: https://www.sas.com/es_es/insights/analytics/big-data-analytics.html. [Último acceso: 22 enero 2021].
- [11] Acciona, «Qué es el Big Data Analytics: la datificación de la sociedad,» [En línea]. Available: <https://www.imnovation-hub.com/es/transformacion-digital/que-es-el-big-data-analytics-la-datificacion-de-la-sociedad/>. [Último acceso: 22 enero 2021].
- [12] «How to Build a Scalable Data Analytics Pipeline,» [En línea]. Available: <https://www.freecodecamp.org/news/scalable-data-analytics-pipeline/>. [Último acceso: 22 enero 2021].
- [13] «Los 4 tipos de Data Analytics,» [En línea]. Available: <https://kurios.la/los-4-tipos-de-data-analytics/>. [Último acceso: 22 enero 2021].
- [14] Iberdrola, «¿Qué es la Inteligencia Artificial?,» [En línea]. Available: <https://www.iberdrola.com/innovacion/que-es-inteligencia-artificial>. [Último acceso: 22 enero 2021].
- [15] «How big data and AI work together,» [En línea]. Available: <https://enterpriseproject.com/article/2019/10/how-big-data-and-ai-work-together>. [Último acceso: 23 enero 2021].
- [16] «Andrew Ng's AI For Everyone – The Definitive Starting Block for AI Novices,» [En línea]. Available: <https://montrealethics.ai/andrew-ngs-ai-for-everyone-the-definitive-starting-block-for-ai-novices/>. [Último acceso: 23 enero 2021].
- [17] D. Calvo, «Conociendo Las Redes Neuronales Con Diego Calvo,» [En línea]. Available: <https://www.bravent.net/conociendo-las-redes-neuronales-con-diego-calvo>. [Último acceso: 23 enero 2021].
- [18] Forbes, «Gartner: 10 Critical Tech Trends For The Next Five Years,» [En línea]. Available: <https://www.forbes.com/sites/ericsavitz/2012/10/22/gartner-10-critical-tech-trends-for-the-next-five-years/?sh=2c7e9d4941b4>. [Último acceso: 23 enero 2021].
- [19] M. A. Menares, «El MDA y su relación con la estrategia marítima,» junio 2010. [En línea]. Available: <https://revistamarina.cl/revistas/2010/6/arenas.pdf>. [Último acceso: 23 enero 2021].
- [20] J. Maiz, «Centro de Operaciones y Vigilancia de Acción Marítima de la Armada,» 8 septiembre 2017. [En línea]. Available: <https://www.defensa.com/en-abierto/centro-operaciones-vigilancia-accion-maritima-armada>. [Último acceso: 24 enero 2021].
- [21] M. d. I. G. García, «EL COVAM DE LA ARMADA AL SERVICIO DE LA COMUNIDAD MARÍTIMA,» [En línea]. Available: <https://armada.defensa.gob.es/ArmadaPortal/ShowPropertyServlet?nodePath=/BEA%20Repository/Desktops/Portal/ArmadaEspannola/Pages/conocenos/especiales/05actividades/99pirateria/02COVAMIEE/02COVAMIEE-es/argCOVAMIEEES>. [Último acceso: 24 enero 2021].
- [22] IMO, «Organización Marítima Internacional,» [En línea]. Available: [https://www.imo.org/es/About/Conventions/Paginas/International-Convention-for-the-Safety-of-Life-at-Sea-\(SOLAS\),-1974.aspx](https://www.imo.org/es/About/Conventions/Paginas/International-Convention-for-the-Safety-of-Life-at-Sea-(SOLAS),-1974.aspx). [Último acceso: 25 enero 2021].

- [23] C. Systems, «ABOUT AUTOMATIC IDENTIFICATION SYSTEMS,» [En línea]. Available: <https://comarsystems.com/what-is-ais/>. [Último acceso: 25 enero 2021].
- [24] «Marine Traffic,» [En línea]. Available: <https://www.marinetraffic.com/en/ais/home/centerx:4.6/centery:24.3/zoom:2>. [Último acceso: 25 enero 2021].
- [25] «SVB,» [En línea]. Available: <https://www.svb-marine.es/es/nasa-marine-receiver-plotter-a-i-s-radar.html>. [Último acceso: 25 enero 2021].
- [26] B. L. S. M. M. N.G. B. Ristic, «Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction,» [En línea]. Available: N.G. B. Ristic, B. La Scala, M. Morelande, Statistical analysis of motion patterns in ais. [Último acceso: 02 febrero 2021].
- [27] M. V. K. B. Giuliana Pallotta, «Vessel Pattern Knowledge Discovery from AIS Data: A Framework for Anomaly Detection and Route Prediction,» [En línea]. Available: https://www.researchgate.net/publication/254864311_Vessel_Pattern_Knowledge_Discovery_from_AIS_Data_A_Framework_for_Anomaly_Detection_and_Route_Prediction. [Último acceso: 02 febrero 2021].
- [28] F. G. P. K. Y. W. N. N. W. Zhang, An advanced method for detecting possible near miss ship collisions from AIS data.
- [29] Y. H. A. Y. N. C. F. C. F.-T. C. J. Chang, Vessel traffic analysis for maritime intelligent transportation system.
- [30] J. C. Y. Su, Spatial cluster detection for the fishing vessel monitoring systems.
- [31] F. L. G. P. J. Chen, A quantitative approach for delineating principal fairways of ship passages through a strait.
- [32] G. P. M. V. Virginia Fernandez Arguedas, Maritime Traffic Networks: From Historical Positioning Data to Unsupervised Maritime Traffic Monitoring.
- [33] T. N. S. A. T. Organization, «2019 HIGHLIGHTS SCIENCE AND TECHNOLOGY ORGANIZATION,» 2019.
- [34] M. A. B. e. al, «“A Cloud Architecture for big data analytics and real-time anomaly detection in global maritime applications”,» Nápoles, 2019.
- [35] J. A. M. e. al., «"Detection of maritime anomalous behavior in a successful MARISA North Sea trial".,» Lerizi (La Spezia), 2019.
- [36] D. F. e. al, «“Anomaly Detection in the Maritime Domain: Comparison of Traditional and Big Data Approach”,» 2019.
- [37] E. C. A.-L. J. CMRE, «Proceedings of the Big Maritima Data Workshop - January 2019».
- [38] «Introducción al Análisis Cluster. Consideraciones generales,» [En línea]. Available: <https://www.ugr.es/~gallardo/pdf/cluster-1.pdf>. [Último acceso: 26 enero 2021].
- [39] I. J. B. Sánchez, «Técnicas de Agrupamiento Para el Analisis de Datos Cuantitativos y Cualitativos,» septiembre 2003. [En línea]. Available: https://www.researchgate.net/profile/Ignacio_Benitez/publication/239526131_Tecnicas_de_Agrupamiento_para_el_Analisis_de_Datos_Cuantitativos_y_Cualitativos/links/00b7d51c1

- 5cca2cb1f000000/Tecnicas-de-Agrupamiento-para-el-Analisis-de-Datos-Cuantitativos-y-Cu. [Último acceso: 28 enero 2021].
- [40] A. K. Jain, Algorithms for Clustering Data, 1988.
- [41] A. K. Jain, «Data Clustering: 50 Years Beyond K-Means,» [En línea]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.151.4286&rep=rep1&type=pdf>. [Último acceso: 26 enero 2021].
- [42] J. L. D. Ruano, «Técnicas de agrupamiento para identificación y control por modelos locales,» julio 2003. [En línea]. Available: <https://riunet.upv.es/bitstream/handle/10251/2684/tesisUPV1703.pdf?sequence=1&isAllowed=y>. [Último acceso: 2021 enero 31].
- [43] C. V. Royo, «Aplicación comparativa de cuatro modelos de clustering para datos GTE_x,» enero 2019. [En línea]. [Último acceso: 31 enero 2021].
- [44] J. Sander, «Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications,» [En línea]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.1629&rep=rep1&type=pdf>. [Último acceso: 31 enero 2021].
- [45] T. Schreiber, «A Voronoi Diagram Based Adaptive K-Means-Type Clustering Algorithm for Multidimensional Weighted Data,» [En línea]. Available: <https://www.semanticscholar.org/paper/A-Voronoi-Diagram-Based-Adaptive-K-Means-Type-for-Schreiber/13b3c334bda636948ccbfa274fcd45c7bce9dc01/figure/1>. [Último acceso: 01 febrero 2021].
- [46] J. A. Rodrigo, «Clustering y heatmaps: aprendizaje no supervisado,» septiembre 2017. [En línea]. Available: https://rpubs.com/Joaquin_AR/310338. [Último acceso: 31 enero 2021].
- [47] Mathworks, «kmedoids,» [En línea]. Available: <https://es.mathworks.com/help/stats/kmedoids.html>. [Último acceso: 02 febrero 2021].
- [48] «DBSCAN and Misc. clustering topics,» 27 octubre 202. [En línea]. Available: <http://faculty.juniata.edu/rhodes/ml/dbscan.htm>. [Último acceso: 02 febrero 2021].
- [49] J. B. Jorba, «GeoCluster: Librería de algoritmos de agrupamiento para objetos geoespaciales,» [En línea]. Available: <https://core.ac.uk/download/pdf/289986809.pdf>. [Último acceso: 01 febrero 2021].
- [50] U. d. Valladolid, «Análisis Clúster II,» [En línea]. Available: <http://www.eio.uva.es/~valentin/ad3d/anadat/acp/marques/AnalClustII.pdf>. [Último acceso: 19 febrero 2021].
- [51] D. A. K. Alexander Hinneburg. [En línea]. Available: <http://resources.mpi-inf.mpg.de/d5/teaching/ss00/proseminar-papers/6/keim-kdd98.pdf>. [Último acceso: 01 febrero 2021].
- [52] H.-H. G. Alexander Hinneburg, «DENCLUE 2.0: Fast Clustering based on Kernel Density Estimation,» [En línea]. Available: <http://www2.cs.uh.edu/~ceick/ML/Denclue2.pdf>. [Último acceso: 02 febrero 2021].
- [53] J. Y. R. M. Wei Wang, «STING : A Statistical Information Grid Approach to Spatial Data Mining,» [En línea]. Available:

- <http://www.cs.ecu.edu/~dingq/CSCI6905/readings/STING.pdf>. [Último acceso: 01 febrero 2021].
- [54] «Recommendation system and STING,» [En línea]. Available: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781783982103/6/ch06lv1sec62/recommendation-system-and-sting. [Último acceso: 02 febrero 2021].
- [55] «CLIQUE: Grid-Based Subspace Clustering,» [En línea]. Available: <https://github.com/annoviko/pyclustering/issues/381>. [Último acceso: 02 febrero 2021].
- [56] P. Andritsos, «Data Clustering Techniques,» [En línea]. Available: https://www.researchgate.net/publication/2847269_Data_Clustering_Techniques. [Último acceso: 01 febrero 2021].
- [57] «<https://www.r-project.org/>,» [En línea]. Available: <https://www.r-project.org/>. [Último acceso: 04 febrero 2021].
- [58] Viewnext, «Python vs R para Data Science,» [En línea]. Available: <https://www.viewnext.com/python-vs-r-para-data-science/>. [Último acceso: 04 febrero 2021].
- [59] «<https://www.python.org/>,» [En línea]. Available: <https://www.python.org/>. [Último acceso: 04 febrero 2021].
- [60] «Ventajas e inconvenientes de Python y R para la ciencia de datos,» [En línea]. Available: <https://www.bbvaapimarket.com/es/mundo-api/ventajas-e-inconvenientes-de-python-y-r-para-la-ciencia-de-datos/>. [Último acceso: 04 febrero 2021].
- [61] L. Toro, «Anaconda Distribution: La Suite más completa para la Ciencia de datos con Python,» [En línea]. Available: <https://blog.desdelinux.net/ciencia-de-datos-con-python/>.
- [62] «AISHub - AIS Data Exchange,» [En línea]. Available: <https://www.aishub.net/api>. [Último acceso: 23 febrero 2021].
- [63] VesselFinder, «AIS Dataset - AIS API - VesselFinder,» [En línea]. Available: <https://api.vesselfinder.com/docs/response-ais.html>. [Último acceso: 23 febrero 2021].
- [64] ABC, «Qué es el «bunkering» y por qué España quiere acabar con él,» ABC, 2013.
- [65] L. K. a. P. Rousseuw, Finding Groups in Data: An Introduction to Cluster, New York: John Wiley & Sons Inc, 1990..
- [66] N. R. a. I. S. Sitanggang, «IOP Conference Series: Earth and Environmental Science,» [En línea]. Available: <https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf>. [Último acceso: 24 febrero 2021].
- [67] ICHI.PRO, «9 medidas de distancia en ciencia de datos,» [En línea]. Available: <https://ichi.pro/es/9-medidas-de-distancia-en-ciencia-de-datos-159983401462266>. [Último acceso: 04 marzo 2021].
- [68] ICHI.PRO, «Comprensión de HDBSCAN y la agrupación en clústeres basada en densidad,» [En línea]. Available: <https://ichi.pro/es/comprendion-de-hdbscan-y-la-agrupacion-en-clusteres-basada-en-densidad-19918965781006>. [Último acceso: 2 marzo 2021].
- [69] L. Š. Z. K. Rino Bošnjak, «Automatic Identification System in Maritime Traffic and Error Analysis,» 2012.

ANEXO I: CÓDIGO DBSCAN

```
#Importamos los módulos que vamos a necesitar

import pandas as pd
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn import metrics
import cartopy.feature as cfeature
from datetime import datetime
from haversine import haversine, Unit

#Calcular distancia en MN entre dos localizaciones

def distancia_puntos(a,b):

    pos_1 = [X.loc[a,"lat"], X.loc[a,"lon"]]
    pos_2 = [X.loc[b,"lat"], X.loc[b,"lon"]]

    distancia = haversine(pos_1, pos_2, unit=Unit.NAUTICAL_MILES)

    return distancia

#Diferencia de tiempos

def intervalo(a,b):

    tiempo_1 = X.loc[a,"time"]
    tiempo_convertido = tiempo_1[1:20]

    tiempo_2 = X.loc[b,"time"]
    tiempo_convertido_2 = tiempo_2[1:20]

    t1 = datetime.strptime(tiempo_convertido, '%Y-%m-%d %H:%M:%S').timestamp()
    t2 = datetime.strptime(tiempo_convertido_2, '%Y-%m-%d %H:%M:%S').timestamp()

    time = (t2 -t1)/3600

    return time

#Leemos el archivo csv

X = pd.read_csv(csv_elegido,
    header=1,
    names=["mmsi", "time", "lon", "lat"])
X = X.reset_index(drop=True)
X = X.sort_values(["mmsi", "time"], ascending = [True, True])

X.loc[0, "valid"] = True
moving = False

for i in range(1, len(X)):
    if X.loc[i, "mmsi"] == X.loc[i-1, "mmsi"]:
        time = intervalo(i-1, i)

        if time == 0:
            X.loc[i, "valid"] = False
            #Descartar posibles duplicados
```

```
        else:
            distancia = distancia_puntos(i-1,i)
            vel = distancia / time

            if vel < 1:
                if moving == True:
                    X.loc[i,"valid"] = True
                    moving = False

                else:
                    X.loc[i,"valid"] = False

            else:
                X.loc[i,"valid"] = False
                moving = True

    else:
        X.loc[i,"valid"] = True
        moving = False

X = X.drop(X[X["valid"] == False].index)
X = X.reset_index(drop=True)
X = X.drop(["mmsi","time","valid"], axis=1)
X

neighbors = NearestNeighbors(n_neighbors=minsamples_elegido)
neighbors_fit = neighbors.fit(X)
distances, indices = neighbors_fit.kneighbors(X)

#Se ordenan las distancias de forma ascendente y se grafica el resultado
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)

#Zoom en el "codo" de la gráfica. para ejecutar, volver a ejecutar las líneas
neighbor.
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.xlim(400, 500)
plt.ylim(0, 0.5)

#Ejecutamos el algoritmo DBSCAN

###Para USO CON DISTANCIA EUCLÍDEA usar el siguiente###
db = DBSCAN(eps=eps_elegido, min_samples=minsamples_elegido).fit(X)
labels = db.labels_

###Para USO CON DISTANCIA HAVERSINE usar el siguiente###
#Para usar Haversine Distance, tenemos que convertir eps a radianes también
#Las coordenadas se convierten a radianes automáticamente con np.radians(X)
añadido a DBSCAN
km_per_radian = 6371.0088
epsilon = eps_elegido / km_per_radian
db = DBSCAN(eps=epsilon, min_samples=minsamples_elegido, algorithm="ball_tree",
metric="haversine").fit(np.radians(X))
labels = db.labels_

#Número de clusters y outliers
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
```

```
print("Estimated number of clusters: ", n_clusters_)
print("Estimated number of noise points: ", n_noise_)
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(X, labels))

#Obtenemos el gráfico
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution="10m")
ax.add_feature(cfeature.RIVERS)
ax.add_feature(cfeature.LAND)
ax.gridlines()
ax.set_extent([min_lon, max_lon, min_lat, max_lat], ccrs.PlateCarree())
plt.scatter(X["lon"], X["lat"],
            c=labels,
            s=10,
            alpha=1,
            transform=ccrs.PlateCarree())

#Gráfico sin outliers
Y = X
X = X.drop(X[X["labels"] == -1].index)
X = X.reset_index(drop=True)
Y["labels"] = labels
Y = Y.drop(Y[Y["labels"] == -1].index)
Y = Y.reset_index(drop=True)
labels_limpio = Y["labels"]
Y = Y.drop(["labels"], axis=1)

ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution="10m")
ax.add_feature(cfeature.RIVERS)
ax.add_feature(cfeature.LAND)
ax.gridlines()
ax.set_extent([min(X["lon"]), max(X["lon"]), 35.5, max(X["lat"])],
              ccrs.PlateCarree())

plt.scatter(Y["lon"], Y["lat"],
            c=labels_limpio,
            s=10,
            alpha=1,
            transform=ccrs.PlateCarree())

#Obtenemos el gráfico para los outliers
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution="10m")
ax.add_feature(cfeature.RIVERS)
ax.add_feature(cfeature.LAND)
ax.gridlines()
ax.set_extent([min_lon, max_lon, min_lat, max_lat], ccrs.PlateCarree())

for i in range(0, len(labels)):
    if labels[i] == -1:
        plt.scatter(X.lon[i], X.lat[i],
                    s=10,
                    alpha=1,
                    c="r",
                    transform=ccrs.PlateCarree())
```

NOTA: Sustituir los valores de “csv_elegido”, “eps_elegido”, “minsamples_elegido” y “[min_lon, max_lon, min_lat, max_lat]” por los valores que con los que se desee trabajar.

ANEXO II: CÓDIGO HDBSCAN

```
#Importamos los módulos que vamos a necesitar

import pandas as pd
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
from sklearn import metrics
import cartopy.feature as cfeature
from hdbscan import HDBSCAN
from datetime import datetime
from haversine import haversine, Unit

#Calcular distancia en MN entre dos localizaciones

def distancia_puntos(a,b):

    pos_1 = [X.loc[a,"lat"], X.loc[a,"lon"]]
    pos_2 = [X.loc[b,"lat"], X.loc[b,"lon"]]

    distancia = haversine(pos_1, pos_2, unit=Unit.NAUTICAL_MILES)

    return distancia

#Diferencia de tiempos

def intervalo(a,b):

    tiempo_1 = X.loc[a,"time"]
    tiempo_convertido = tiempo_1[1:20]

    tiempo_2 = X.loc[b,"time"]
    tiempo_convertido_2 = tiempo_2[1:20]

    t1 = datetime.strptime(tiempo_convertido, '%Y-%m-%d %H:%M:%S').timestamp()
    t2 = datetime.strptime(tiempo_convertido_2, '%Y-%m-%d %H:%M:%S').timestamp()

    time = (t2 -t1)/3600

    return time

#Leemos el archivo csv

X = pd.read_csv(csv_elegido,
                header=1,
                names=["mmsi","time","lon","lat"])
X = X.reset_index(drop=True)
X = X.sort_values(["mmsi","time"], ascending = [True, True])

X.loc[0,"valid"] = True
moving = False

for i in range(1,len(X)):
    if X.loc[i,"mmsi"] == X.loc[i-1,"mmsi"]:
        time = intervalo(i-1,i)

        if time == 0:
            X.loc[i,"valid"] = False
            #Descartar posibles duplicados
        else:
```

```
    distancia = distancia_puntos(i-1,i)
    vel = distancia / time

    if vel < 1:
        if moving == True:
            X.loc[i,"valid"] = True
            moving = False

        else:
            X.loc[i,"valid"] = False

    else:
        X.loc[i,"valid"] = False
        moving = True

else:
    X.loc[i,"valid"] = True
    moving = False

X = X.drop(X[X["valid"] == False].index)
X = X.reset_index(drop=True)
X = X.drop(["mmsi", "time", "valid"], axis=1)
X

neighbors = NearestNeighbors(n_neighbors=minsamples_elegido)
neighbors_fit = neighbors.fit(X)
distances, indices = neighbors_fit.kneighbors(X)

#Se ordenan las distancias de forma ascendente y se grafica el resultado
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)

#Zoom en el "codo" de la gráfica. para ejecutar, volver a ejecutar las líneas
neighbor.
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.xlim(400, 500)
plt.ylim(0, 0.5)

# Ejecutamos el algoritmo HDBSCAN

###Para USO CON DISTANCIA EUCLÍDEA usar el siguiente###
hdb = HDBSCAN(min_cluster_size = minclustersize_final).fit(X)
labels = hdb.labels_

####USAR SOLO PARA HDBSCAN CON DISTANCIA HAVERSINE###
#Para usar Haversine Distance, tenemos que convertir las coordenadas a radianes
con el atributo np.radians(X) añadido a HDBSCAN
hdb = HDBSCAN(min_cluster_size=minclustersize_final, min_samples=minsamples_final,
metric="haversine").fit(np.radians(X))
labels = hdb.labels_

####USAR SOLO PARA HDBSCAN con SELECTING EPSILON###
km_per_radian = 6371.0088
epsilon = eps_elegido / km_per_radian
hdb = HDBSCAN(min_cluster_size=minclustersize_elegido, metric="haversine",
min_samples=minsamples_final, cluster_selection_epsilon =
epsilon).fit(np.radians(X))
labels = hdb.labels_
```

```
#Número de clusters y outliers
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimated number of clusters: ", n_clusters_)
print("Estimated number of noise points: ", n_noise_)
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(X, labels))

#Obtenemos el gráfico
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution="10m")
ax.add_feature(cfeature.RIVERS)
ax.add_feature(cfeature.LAND)
ax.gridlines()
ax.set_extent([min_lon, max_lon, min_lat, max_lat], ccrs.PlateCarree())
plt.scatter(X["lon"], X["lat"],
            c=labels,
            s=10,
            alpha=1,
            transform=ccrs.PlateCarree())

#Gráfico sin outliers
Y = X
X = X.drop(X[X["labels"] == -1].index)
X = X.reset_index(drop=True)
Y["labels"] = labels
Y = Y.drop(Y[Y["labels"] == -1].index)
Y = Y.reset_index(drop=True)
labels_limpio = Y["labels"]
Y = Y.drop(["labels"], axis=1)

ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution="10m")
ax.add_feature(cfeature.RIVERS)
ax.add_feature(cfeature.LAND)
ax.gridlines()
ax.set_extent([min(X["lon"]), max(X["lon"]), 35.5, max(X["lat"])],
              ccrs.PlateCarree())

plt.scatter(Y["lon"], Y["lat"],
            c=labels_limpio,
            s=10,
            alpha=1,
            transform=ccrs.PlateCarree())

#Obtenemos el gráfico para los outliers
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines(resolution="10m")
ax.add_feature(cfeature.RIVERS)
ax.add_feature(cfeature.LAND)
ax.gridlines()
ax.set_extent([min_lon, max_lon, min_lat, max_lat], ccrs.PlateCarree())

for i in range(0, len(labels)):
    if labels[i] == -1:
        plt.scatter(X.lon[i], X.lat[i],
                    s=10,
                    alpha=1,
                    c="r",
                    transform=ccrs.PlateCarree())
```

NOTA: Sustituir los valores de “csv_elegido”, “eps_elegido”, “minclustersize_elegido” “minsamples_elegido” y “[min_lon, max_lon, min_lat, max_lat]” por los valores que con los que se desee trabajar.

ANEXO III: CÁLCULO DE LA VELOCIDAD REAL DE LOS BUQUES

```
#Importamos los módulos que vamos a necesitar

import pandas as pd
from datetime import datetime
from haversine import haversine, Unit
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

X = pd.read_csv("volcado4.csv",
               header=1,
               names=["mmsi", "time", "lon", "lat", "sog"])
X = X.reset_index(drop=True)
X = X.sort_values(["mmsi", "time"], ascending = [True, True])
X

#Calcular distancia en MN entre dos localizaciones

def distancia_puntos(a,b):

    pos_1 = [X.loc[a,"lat"], X.loc[a,"lon"]]
    pos_2 = [X.loc[b,"lat"], X.loc[b,"lon"]]

    distancia = haversine(pos_1, pos_2, unit=Unit.NAUTICAL_MILES)

    return distancia

    #Diferencia de tiempos

def intervalo(a,b):

    tiempo_1 = X.loc[a,"time"]
    tiempo_convertido = tiempo_1[1:20]

    tiempo_2 = X.loc[b,"time"]
    tiempo_convertido_2 = tiempo_2[1:20]

    t1 = datetime.strptime(tiempo_convertido, '%Y-%m-%d %H:%M:%S').timestamp()
    t2 = datetime.strptime(tiempo_convertido_2, '%Y-%m-%d %H:%M:%S').timestamp()

    time = (t2 -t1)/3600

    return time

    #Cálculo de la velocidad en función de las posiciones y el tiempo
    transcurrido

X.loc[0,"speed"] = 0.0
for i in range(1,len(X)):
    if X.loc[i,"mmsi"] == X.loc[i-1,"mmsi"]:
        distancia = distancia_puntos(i-1,i)
        time = intervalo(i-1,i)

        if time == 0:
            X.drop(index=i)
            #Eliminar posibles duplicados
        else:
```

```
        vel = distancia / time
        X.loc[i,"speed"] = vel

    else:
        X.loc[i,"speed"] = 0.0
        #Asumimos que en el primer mensaje emitido por cada barco, su velocidad es
nula
X

#Extremadamente extraño que un buque vaya a más de 50 nudos, por lo que suponemos
error en el mensaje
X.drop(X.loc[X["speed"]>50].index, inplace=True)
X.drop(X.loc[X["sog"]>50].index, inplace=True)

X

velocidades = X.drop(["mmsi","time","lat","lon"], axis=1)
velocidades
velocidades.describe()

plt.figure()
sns.lmplot(x="sog", y="speed", data=velocidades)
```

ANEXO IV: FILTRO PARA EXTRAER BUQUES PARADOS

```
#Importamos los módulos que vamos a necesitar

import pandas as pd
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn import metrics
import cartopy.feature as cfeature
from datetime import datetime
from haversine import haversine, Unit

#Ahora importaremos el archivo csv que contiene los datos

X = pd.read_csv(csv_elegido,
                header=1,
                names=["mmsi", "time", "lon", "lat", "sog"]).sort_values(["mmsi", "time"],
                                ascending = [True, True])
print(X)

#Calcular distancia en MN entre dos localizaciones

def distancia_puntos(a,b):

    pos_1 = [X.loc[a,"lat"], X.loc[a,"lon"]]
    pos_2 = [X.loc[b,"lat"], X.loc[b,"lon"]]

    distancia = haversine(pos_1, pos_2, unit=Unit.NAUTICAL_MILES)

    return distancia

#Diferencia de tiempos

def intervalo(a,b):

    tiempo_1 = X.loc[a,"time"]
    tiempo_convertido = tiempo_1[1:20]

    tiempo_2 = X.loc[b,"time"]
    tiempo_convertido_2 = tiempo_2[1:20]

    t1 = datetime.strptime(tiempo_convertido, '%Y-%m-%d %H:%M:%S').timestamp()
    t2 = datetime.strptime(tiempo_convertido_2, '%Y-%m-%d %H:%M:%S').timestamp()

    time = (t2 -t1)/3600

    return time

#Filtro para encontrar buques parados, y que estos no formen un clúster por sí
mismos al emitir mensajes AIS en la misma zona

X.loc[0,"valid"] = False
moving = True

for i in range(1,len(X)):
    if X.loc[i,"mmsi"] == X.loc[i-1,"mmsi"]:
        time = intervalo(i-1,i)
```

```
if time == 0:
    X.loc[i,"valid"] = False
    #Descartar posibles duplicados
else:
    distancia = distancia_puntos(i-1,i)
    vel = distancia / time

    if vel < 1:
        if moving:
            X.loc[i,"valid"] = True
            moving = False

        else:
            X.loc[i,"valid"] = False

    else:
        X.loc[i,"valid"] = False
        moving = True

else:
    X.loc[i,"valid"] = True
    moving = False

X = X.drop(X[X["valid"] == False].index)
X = X.reset_index(drop=True)
X = X.drop(["mmsi","time","sog","valid"], axis=1)
X
```

