Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Predicción de tipo de buque utilizando datos AIS y técnicas de inteligencia artificial

Grado en Ingeniería Mecánica

ALUMNO: Gonzalo Rodríguez Casajús

DIRECTORES: Belén Barragáns Martínez

Pablo Sendín Raña

CURSO ACADÉMICO: 2021-2022

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Predicción de tipo de buque utilizando datos AIS y técnicas de inteligencia artificial

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

Este TFG se enmarca dentro de un proyecto de investigación que el CUD-ENM está desarrollando a petición de la Armada Española con el objetivo de aplicar técnicas de inteligencia artificial para la mejora del conocimiento del entorno marítimo y, más concretamente, ayudar en la detección de anomalías en el comportamiento de los buques.

Analizados los flujos AIS que identifican a cada barco (tipo, zona, cinemática, etc.), se detecta que muchos buques no envían información del tipo de barco, campo clave en la identificación de anomalías de interés.

Para solucionar este problema, en este TFG se propone la aplicación de técnicas de aprendizaje automático supervisado con objeto de predecir el valor de dicho campo, teniendo en cuenta otro tipo de información que sí transmite el barco. Para ello, se entrenarán diferentes modelos, a partir de un conjunto de datos históricos de flujo marítimo mundial (previamente preprocesado y adaptado), empleando el algoritmo *Random Forest*. Se realizan variaciones al algoritmo y se emplean funciones de tratamiento de datos para tratar de mejorar los resultados.

Dichos modelos serán validados convenientemente, de modo que este TFG termina proponiendo el modelo construido a partir de determinados atributos que consiguen maximizar la calidad de la predicción.

PALABRAS CLAVE

Inteligencia artificial, *Machine Learning*, *Random Forest*, AIS, Tipo de buque

AGRADECIMIENTOS

En primer lugar, quiero agradecer a todo el personal de la Escuela Naval Militar que ha invertido parte de su tiempo y trabajo en mi formación y la de todos mis compañeros a lo largo de estos cinco años de escuela.

En especial agradecer a mis dos tutores, Belén y Pablo, para los cuales solo tengo buenas palabras y no me queda más que agradecerles por su impecable función a la hora de dirigir y tutorizar mi trabajo. Agradecerles siempre su paciencia y el apoyo prestado ante las diversas piedras que se han cruzado por el camino hasta la finalización de este trabajo, incluso algunas lanzadas desde países lejanos.

También agradecer al profesor Andrés Suárez por compartir su tiempo y experiencia previa en el ámbito del trabajo para la realización del trabajo.

A mi familia, por la formación que me han dado desde el día en que nací y que han hecho que me convierta en la persona que en la que me he convertido hoy en día. Por ese apoyo incondicional y ser siempre un lugar en el que ampararse.

Por último, me gustaría dirigir unas palabras a mis compañeros de la promoción 422-152, pues son ellos los que han hecho que estos cinco años de escuela sean tan especiales como realmente lo han sido, no me cabe duda de que sin ellos nada habría sido igual.

A todos ellos, gracias por estar ahí.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto y motivación

La introducción de tecnologías que emplean inteligencia artificial está cada vez más presente en múltiples ámbitos de aplicación y comienza a ser ya tecnología fundamental para la realización de numerosas tareas con unos resultados inimaginables hace unos pocos años.

Resulta muy extensa la lista de las compañías y organizaciones que han apostado por estas tecnologías y que trabajan en el desarrollo de las mismas. De hecho, la Armada no es una excepción. Su interés en este aspecto queda reflejado en las líneas generales de la Armada para 2022 [1], un documento redactado por el Almirante Jefe del Estado Mayor de la Armada (AJEMA), donde se exponen los retos a los que se enfrenta la Armada y las directrices que debe seguir para garantizar el éxito en su consecución. En este documento se recalca la importancia de la introducción de tecnologías de inteligencia artificial en las unidades y la establece como una de las principales vías de desarrollo para la Armada. Motivado por esta directriz nacen varios proyectos, como la introducción de técnicas de inteligencia artificial para automatizar el mantenimiento de los buques de nueva generación [2] o para mejorar el Conocimiento del Entorno Marítimo (CEM). Precisamente, en el marco de este último proyecto [3] se encuadra este TFG.

Es un proyecto cuyo desarrollo ha sido solicitado por parte de la Armada al Centro Universitario de la Defensa en la Escuela Naval Militar (CUD-ENM) con el propósito de desarrollar un demostrador tecnológico que haga uso de técnicas de inteligencia artificial para apoyar a la Armada en el desarrollo de los procedimientos operativos que se llevan a cabo para el CEM desde el Centro de Operaciones y Vigilancia de Acción Marítima (COVAM), que es la unidad dentro de la Armada encargada de la supervisión de los espacios marítimos de interés nacional. En concreto, el demostrador descrito utiliza el flujo de datos de los mensajes AIS (*Automatic Identification System*), que contienen información relativa a las características del buque, posicionamiento e información cinemática en tiempo real. El empleo de estos datos combinado con el empleo de técnicas de análisis de datos e inteligencia artificial se usará para la detección de anomalías en los patrones de comportamiento de los buques en la mar.

El concepto de anomalía en los patrones de comportamiento de los buques hace referencia a todas aquellas situaciones en las que las acciones de los buques en la mar no son coherentes con el resto del tráfico o con los datos históricos almacenados del mismo barco. En base a esta definición se puede entender como anomalía un amplio abanico de situaciones. Sin embargo, con el fin de acotar las anomalías cuyo conocimiento es de interés para la actividad de vigilancia marítima por la Armada desde el COVAM, la Armada ha elaborado un listado en el que se desarrollan las anomalías de interés cuya

detección se considera posible mediante el empleo de inteligencia artificial. Estas anomalías normalmente obedecen a la combinación de tres factores: tipo de buque, zona y cinemática.

Los datos AIS nos proporcionan la información necesaria para el conocimiento de estos tres factores. Sin embargo, pese a que uso del sistema se encuentra instalado a bordo de buques conforme a la normativa SOLAS [4] de la Organización Marítima Internacional (OMI) y, en el caso de España, es de obligado cumplimiento, muchos de los barcos a día de hoy no transmiten todos los campos o lo hacen de manera errónea. No hay que olvidar que gran parte de los campos que se recogen en los mensajes son cubiertos manualmente, lo cual nos lleva a pensar en la veracidad de los campos llegando incluso a encontrar numerosos casos en los que directamente los campos no están cubiertos. Centrándonos en el campo al que está sujeto este trabajo, el tipo de buque, aproximadamente un tercio de las tramas AIS no contienen información sobre este campo. Para reforzar esta premisa, en la Figura 1-1 se puede ver una representación de uno de los conjuntos de datos que ha sido utilizado en este estudio donde se muestran los barcos que emiten el tipo de buque de forma correcta (buque) frente a los que dejan el valor sin cubrir (no cubierto) o le dan un valor no válido (0).

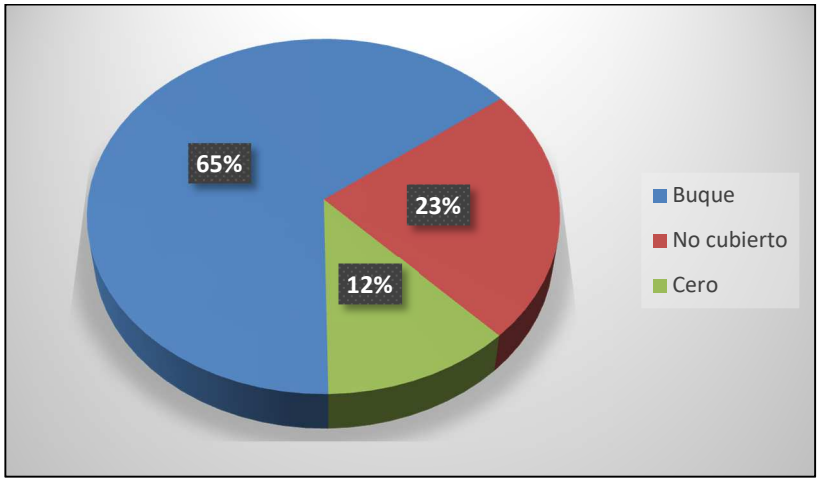

Figura 1-1 Representación de la información que aportan las tramas AIS en el campo *shiptype* (fuente: propia)

La ausencia del tipo de buque de diferentes contactos provoca que no se puedan reconocer las anomalías de interés para el COVAM pues, como ya se ha dicho anteriormente, las anomalías vienen definidas por tres factores, de los cuales uno es el tipo de buque. Además, si se aplica el sentido común, los buques con mayor probabilidad de estar realizando actividades ilícitas serán aquellos que traten de ocultar información. Es en esta tarea donde radica la importancia de este trabajo.

La motivación de este trabajo fin de grado es tratar de solucionar el problema que supone los datos AIS incompletos para la definición de anomalías y paliar así una de las necesidades dentro del proyecto de investigación en el uso de inteligencia artificial en el COVAM desarrollado por el CUD-ENM.

1.2 Objetivos

El objetivo principal de este trabajo fin de grado es la construcción de un modelo empleando técnicas de inteligencia artificial que sea capaz de predecir el tipo de buque de los diferentes contactos basándose para ello en los datos AIS que transmiten. El fin último del desarrollo de este modelo es determinar el tipo de buque de un determinado contacto, del cual se desconoce dicho campo, para que se puedan detectar los diferentes tipos de anomalías definidas por el COVAM.

Para ello se plantea el uso de técnicas de aprendizaje supervisado y la selección del algoritmo cuya definición mejor se ajuste a las necesidades expuestas y a los datos disponibles. Una vez seleccionado el algoritmo, se construirá el modelo, tratando de optimizar su rendimiento mediante la aplicación de variaciones en los parámetros de funcionamiento del propio algoritmo y mediante el empleo de diferentes técnicas de tratamiento de datos.

Constituye también un objetivo importante de este TFG el estudio de la validez de los datos dinámicos y estáticos para contribuir a la clasificación de los distintos tipos de buque, es decir, determinar si verdaderamente son datos que logran discernir un tipo de buque del resto.

Por último, se definirán diferentes experimentos que permitan ayudar a validar el modelo y seleccionar aquel que consigue optimizar la calidad de la predicción.

1.3 Estructura de la memoria

Tras hacer referencia al marco en el que se encuadra este trabajo, contextualizándolo y tras haber definido los objetivos que se esperan alcanzar, se pasa a explicar la estructura que seguirá la presente memoria con el fin de organizar la presentación de los trabajos realizados, así como los resultados obtenidos, de la forma más clara posible. El documento está dividido en cinco capítulos a los que a la finalización de los mismos se incluye la bibliografía que recoge todas las fuentes empleadas y los anexos que servirán de apoyo a lo largo del desarrollo del trabajo. Este TFG se organiza como sigue:

- Capítulo 1: Introducción y objetivos. En este primer capítulo se expone la importancia de la introducción de tecnologías de inteligencia artificial en el mundo actual y la postura de la Armada ante esta tendencia. Se explica el marco en el que se encuadra este trabajo. Por último, se plantean los objetivos a alcanzar en este trabajo y la organización de la memoria.
- Capítulo 2: Estado del arte. En este segundo capítulo se realizará una revisión del marco teórico en el que se engloba este trabajo. Los conceptos teóricos tratados en este capítulo se pueden resumir en los siguientes puntos:
    - Introducción al concepto de inteligencia artificial junto con una reseña histórica de la misma y su importancia en la actualidad.
    - Introducción al aprendizaje automático y los diferentes tipos que existen, así como sus campos de aplicación más populares.
    - Revisión de los algoritmos de inteligencia artificial más populares en la actualidad y explicación del funcionamiento de cada uno de ellos.
    - Introducción al Conocimiento del Entorno Marítimo (CEM) y el papel que ocupa el COVAM y la importancia de las tecnologías de inteligencia artificial para su desarrollo. Además, se presenta por primera vez el sistema AIS, una importante herramienta para la obtención de información para el COVAM y que será la fuente de la que se generarán los datos de entrada para el desarrollo de este trabajo.
    - Repaso de los trabajos previos realizados en el campo de estudio de este trabajo.
- Capítulo 3: Desarrollo del TFG. En este capítulo se describen las herramientas utilizadas y los datos disponibles para lograr el cumplimiento de los objetivos propuestos. Además, se justifica el empleo de *Random Forest* como algoritmo para el desarrollo del modelo, así como los pasos seguidos desde el procesamiento de los datos y la obtención de atributos a partir de los mismos que sirvan para alimentar el algoritmo. Por último, se explican las diferentes funciones que se han utilizado para tratar de mejorar el modelo y los experimentos realizados.
- Capítulo 4: Resultados obtenidos y validación. En este capítulo se exponen los resultados más relevantes de los experimentos realizados y se analizan con el fin de hallar el modelo que mejor resuelva el problema planteado.
- Capítulo 5: Conclusiones y líneas futuras. En este último capítulo se resumen las conclusiones alcanzadas tras el desarrollo del trabajo. También se plantean las posibles líneas futuras que traten de mejorar el trabajo o complementarlo con el fin de seguir avanzando en la consecución de los objetivos del proyecto.
- Finalmente se adjunta la bibliografía donde se especifican las fuentes utilizadas a lo largo del trabajo y una serie de anexos que tienen como propósito apoyar el desarrollo del trabajo para una mejor comprensión del mismo.

2 ESTADO DEL ARTE

El presente capitulo tratará de estudiar en qué consiste la inteligencia artificial (IA) y el desarrollo de esta reciente tecnología, para seguir con un estudio de las diferentes técnicas de *Machine Learning* abordando los diferentes tipos de aprendizaje como el aprendizaje supervisado, no supervisado y por refuerzo. Además, se expondrán los algoritmos de aprendizaje supervisado y no supervisado más utilizados para el desarrollo de máquinas inteligentes. Por otro lado, se tratará de explicar en qué consiste el Conocimiento del Entorno Marítimo, así como los principales factores que le afectan, que son la principal motivación de este trabajo. Por último, se expondrán una serie de proyectos previos relacionados que han abordado parcialmente objetivos similares al del presente Trabajo Fin de Grado.

2.1 La inteligencia artificial

2.1.1 Definición

Actualmente, encontrar una definición de inteligencia artificial es una tarea compleja. Son muchas las definiciones que se aproximan como válidas para hacer referencia a esta emergente tecnología cuya importancia está creciendo cada vez más. La dificultad a la hora de acuñar una definición para la inteligencia artificial radica en que es un concepto que depende de la propia definición de inteligencia, que hoy en día sigue teniendo múltiples interpretaciones.

La primera vez que se acuñaría este término sería por el miembro del departamento de Ciencias Informáticas de la Universidad de Standford John McCarthy durante la conferencia Dartmouth, a la que se referiría como la "*ciencia e ingenio de hacer máquinas inteligentes, específicamente, programas de cómputo inteligentes*" [5].

El problema radica en que esta definición seguía sujeta a la interpretación del significado de lo que se considera la inteligencia. Con el paso del tiempo y los avances en las ciencias computacionales, el concepto de inteligencia artificial abarcó un área aún más amplia pudiendo definir la misma como la habilidad de una máquina de presentar las mismas capacidades que los seres humanos, como el razonamiento, el aprendizaje, la creatividad y la capacidad de planear [6].

Actualmente, el criterio más aceptado a la hora de definir este concepto es el de los cuatro enfoques redactado por los científicos informáticos Stuart Russell y Peter Norvig en su libro *Artificial Intelligence: A Modern Approach* [7]. Sus autores agruparon las definiciones de ocho libros de texto, las cuales se ven reflejadas en la Tabla 1, y clasificaron la inteligencia artificial según los siguientes cuatro criterios: sistemas que piensan como humanos, sistemas que piensan racionalmente, sistemas que actúan como humanos y sistemas que actúan racionalmente.

Como se puede observar en la Tabla 1, los criterios de la parte superior hacen referencia al uso de la mente y el razonamiento, mientras que los de la inferior refieren a la forma de actuar, la conducta. Por otro lado, también se puede observar una división entre los criterios de la izquierda, que lo que buscan es imitar el comportamiento humano, y los de la derecha, que buscan ir más allá y profundizar en lo que es la inteligencia tratando de hacer máquinas racionales.

Sistemas que piensan como humanos	**Sistemas que piensan racionalmente**
«El nuevo y excitante esfuerzo de hacer que los computadores piensen… máquinas con mentes, en el más amplio sentido literal». (Haugeland, 1985) «[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje…» (Bellman, 1978)	«El estudio de las facultades mentales mediante el uso de modelos computacionales». (Charniak y McDermott, 1985) «El estudio de los cálculos que hacen posible percibir, razonar y actuar». (Winston, 1992)
Sistemas que actúan como humanos	**Sistemas que actúan racionalmente**
«El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia». (Kurzweil, 1990) «El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor». (Rich y Knight, 1991)	«La Inteligencia Computacional es el estudio del diseño de agentes inteligentes». (Poole et al., 1998) «IA… está relacionada con conductas inteligentes en artefactos». (Nilsson, 1998)

Tabla 1 Enfoques de Russell-Norvig para la inteligencia artificial (tomada de [7])

Asimismo, existe una clasificación grosso modo dentro del concepto de inteligencia artificial que la divide en dos grandes grupos: IA débil e IA fuerte. Hablamos de IA débil cuando nos referimos a sistemas que pueden realizar un determinado número de tareas y se les ha enseñado a realizar dichas actividades mediante algoritmos y aprendizaje guiado con técnicas de *Machine Learning* y *Deep Learning*, que mencionaremos más adelante. Por el contrario, se le da el nombre de IA fuerte a aquellos sistemas que pueden operar en muchos dominios diferentes, gracias a que las máquinas poseen una inteligencia similar a la capacidad cognitiva de los humanos. A día de hoy, toda IA de la que disponemos se engloba dentro de la IA débil y, como afirman los expertos en esta tecnología, aún estamos a muchos años de que esto cambie [8].

2.1.2 Contexto histórico

Las primeras tentativas hacia el mundo de la inteligencia artificial datan de siglos antes de su creación. Aristóteles fue el primero en definir un conjunto de reglas que describen una parte del funcionamiento de la mente para obtener conclusiones racionales a partir de ciertas premisas [9]. En tiempos análogos, Ctsebio de Alejandría crearía un regulador de flujo de agua que cambiaba su funcionamiento de forma racional, dando lugar a la primera máquina autocontrolada de manera racional, pero sin razonamiento [10]. Incluso el mismo Descartes llegó a plantear la idea de animal máquina [11], que consistiría en una máquina capaz de imitar o emular ciertos comportamientos humanos.

A pesar de todos estos acontecimientos no será hasta el año 1943, el año en el que se podrá establecer el nacimiento de la inteligencia artificial, pese a que sería años más tarde cuando se acuñaría este término.

El médico de la Universidad de Illinois Warren McCulloch y el matemático Warren Pitts construyeron el primer modelo matemático de una neurona del cerebro humano [12].

Resulta imposible hablar de inteligencia artificial sin mencionar a Alan Turing. Además de sus grandes aportaciones, como su máquina universal, sin las cuales no se habría conseguido llegar a desarrollar dichas tecnologías, trató de consolidar el campo que hasta entonces era tan abstracto de la inteligencia artificial en su artículo "*Computing Machinery and Intelligence*" [13]. En este trata de establecer si una máquina es inteligente o no en base a su conocido "*imitation game*", o renombrado como Test de Turing en su honor. Este acontecimiento marcaría al desarrollo de la inteligencia artificial en los años venideros, un resumen de los principales hitos se puede observar en la Figura 2-1. A principios de los años 80 la efectividad del Test de Turing se vería cuestionada con la publicación por parte del filósofo J. Searle de su experimento de "La habitación China" [14], el cual establecía que superar el Test de Turing no supone que una máquina posea inteligencia humana, solo que es capaz de simularla.

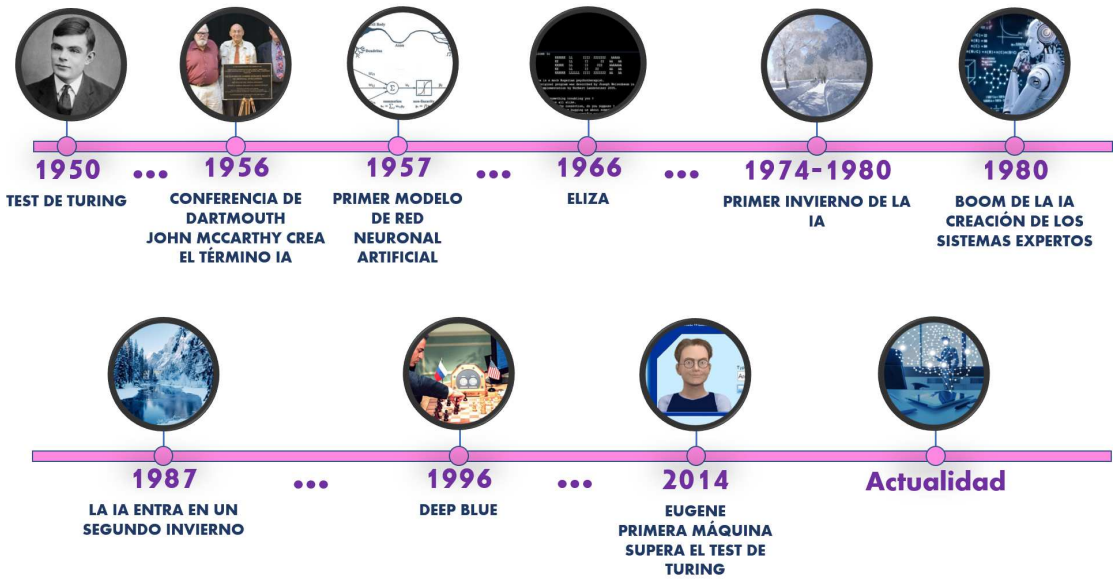

Figura 2-1 Eje cronológico de hitos de la IA (fuente: propia)

Sin embargo, no es hasta el año 1956, con la celebración de la conferencia de Dartmouth, cuando nace el campo de la inteligencia artificial tal y como la conocemos actualmente. En esta conferencia, John McCarthy establece por primera vez el término inteligencia artificial y le da una definición. Asimismo, se establecen las bases de esta rama y sus líneas futuras. Se acota por primera vez el campo que abarca la inteligencia artificial: "Este estudio procederá sobre la base de que todos los aspectos del aprendizaje o de rasgos de la inteligencia pueden, en principio, ser descritos de una forma tan precisa que se puede crear una máquina que los simule" [5].

Tras la conferencia de Dartmouth, comienzan los años de oro de la IA. Las altas expectativas sobre esta nueva ciencia llevaron a los gobiernos e instituciones privadas a invertir grandes sumas de capital en todo lo que estuviera relacionado con la IA. Como consecuencia, la IA evoluciona rápidamente y se alcanzan importantes avances como la creación del primer modelo de red neuronal artificial por Frank Rosenblat en 1957 [15]. Además, se desarrollan increíbles programas como "STUDENT" [16] de Daniel Bobrow en 1964, que resolvía problemas matemáticos de instituto y preuniversitarios, o el primer *chatbot* "ELIZA" [17] diseñado por el MIT en el año 1966, que era una psicoterapeuta robótica con la que podías entablar conversaciones. En la Figura 2-2 se puede apreciar un ejemplo de funcionamiento del software "ELIZA".

Figura 2-2 Ejemplo de conversación con el software "ELIZA" [18]

Con el paso de los años, esta ciencia se encontró con varios problemas como la falta de capacidad de computación y de capacidad de almacenamiento de datos de los equipos y el aumento de la necesidad de tiempo y recursos que se producía de forma exponencial cuanto más se complicaba un problema. Estos inconvenientes ligados a que no se alcanzaron las altas expectativas que se proponían inicialmente, tuvieron como consecuencia que tanto inversores como científicos perdieran el interés en este campo, iniciándose así en el año 1974 la etapa conocida como el "invierno" de la inteligencia artificial.

No fue hasta los años 80 cuando se produce un boom en el campo de la IA. Esta tecnología vuelve a llamar la atención de los inversores privados gracias a la creación de los llamados Sistemas Expertos [12], que eran programas que almacenaban conceptos profesionales pudiendo resolver dudas de las empresas sin necesidad de la contratación de nuevos trabajadores, ahorrando grandes sumas de dinero a las empresas. Sin embargo, las expectativas demasiado altas y el optimismo exagerado impacientaron a los inversores. Este hecho, junto con la aparición de una burbuja económica propiciaron que la IA se sumiera en un segundo invierno en el año 1987, que se prolongará hasta finales del siglo XX [19].

La última etapa de la IA se extiende desde este momento hasta la actualidad. Se puede afirmar que la IA volvió a experimentar de nuevo un crecimiento gracias a los avances tecnológicos que son expuestos en el siguiente apartado. Durante este periodo se llevaron a cabo logros importantes tales como la creación de Deep Blue [20], un ordenador creado por IBM que fue capaz de batir al, por aquel entonces, campeón mundial de ajedrez Gari Kaspárov en 1996 (véase Figura 2-3). Y en el año 2014, el software "Eugene Goostman" [21] se convierte en la primera máquina en superar el Test de Turing.

Figura 2-3 Partida de Deep Blue contra Gari Kaspárov [20]

2.1.3 La inteligencia artificial en la actualidad

En la actualidad, los problemas que frenaron el desarrollo de la IA se han visto solucionados gracias a los grandes avances tecnológicos que se han dado en las últimas décadas. Como ya se mencionó previamente, los grandes problemas que frenaron el desarrollo de esta prometedora tecnología fueron la falta de grandes volúmenes de datos y la escasa potencia y capacidad de los equipos de la época.

El primer problema disminuyó parcialmente con la aparición de Internet. Con el nacimiento de Internet, comenzó el intercambio de datos constante, dando lugar a una ingente cantidad de datos de los que las máquinas podían disponer.

En cuanto al problema de la falta de potencia y memoria, una forma simple de observar por qué este problema ha perdido transcendencia con el paso de los años es mediante el ejemplo que se expone en la Figura 2-4. En ella podemos observar tres capturas distanciadas 30 años entre ellas siguiendo el orden de izquierda a derecha. En primer lugar, se aprecia un ordenador de hace aproximadamente 60 años con 5MB de memoria. En el centro se ve un ordenador comercial con 10MB de memoria. Lo interesante llega cuando se presta atención a la tercera, que se trata de un disco duro actual con 1TB de memoria. Es decir, a comienzos de la evolución histórica de los equipos informáticos se consiguió duplicar la capacidad de los equipos, pero años más tarde y en el mismo intervalo de tiempo se consigue que la cifra anterior se multiplique de forma abrumadora.

Figura 2-4 Ejemplo de evolución histórica de los equipos informáticos [22] [23]

La IA actualmente está abarcando un amplio campo de diferentes actividades realizadas por los seres humanos entre las que destacan las siguientes líneas de investigación y desarrollo: la robótica, la visión artificial, técnicas de aprendizaje y la gestión del conocimiento [24].

La utilización de la IA por parte de las empresas está pasando a ser un factor de vital importancia. Tanto en la industria, mediante la implementación de máquinas automatizadas que tratan de reproducir el trabajo que previamente realizaba un operador humano, hasta en la gestión de compañías para la predicción de valores, tratamiento de los datos o selección de mejores estrategias para llevar a cabo determinadas actividades.

La inteligencia artificial no solo se limita a las grandes empresas, sino que se está implementando de forma muy sutil en herramientas que usamos día a día, la mayoría sin ser apenas consciente de ello. La IA forma ya parte de nuestras vidas y podemos observarla en cuestiones tan cotidianas como el buscador de Google, recomendaciones de Youtube y Spotify, antivirus, videojuegos y detectores de Spam del correo electrónico [25]. No solo la usamos a diario, sino que la portamos siempre con nosotros en nuestros dispositivos móviles con el asistente inteligente Siri de la Figura 2-5 o Google Assistant Go. Estos son solo unos pocos ejemplos, pues en los tiempos que corren somos capaces de encontrar la IA en una infinidad de aplicaciones.

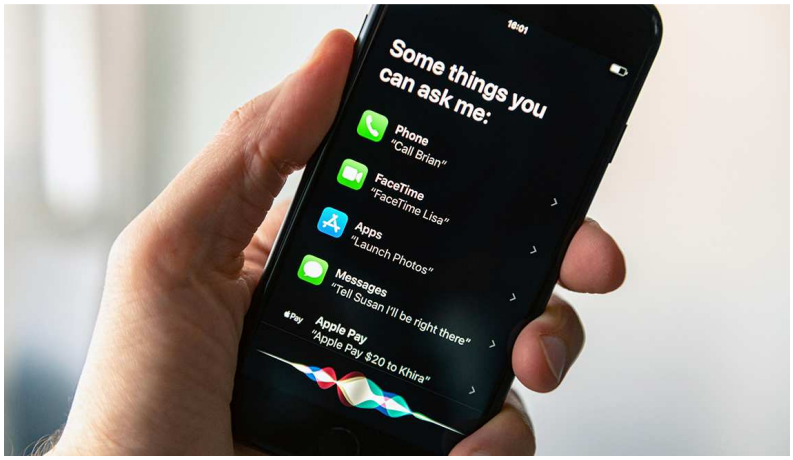

Figura 2-5 Siri, inteligencia artificial con funciones de asistente personal [26]

Ha llegado a ser tal el impacto de esta revolucionaria tecnología en la sociedad actual que son numerosas las obras tanto literarias como filmográficas en las que se representan avances desmesurados de la IA donde las máquinas parecen prácticamente seres humanos y, aunque todavía estamos muy lejos de alcanzar este tipo de aplicaciones, no hay duda de que la IA está creciendo a pasos agigantados y que actualmente se encuentra en su mejor momento en cuanto a desarrollo se refiere.

2.2 *Machine Learning*

El *Machine Learning* (o aprendizaje automático en su traducción al castellano) es la rama de la inteligencia artificial que se centra en la creación de sistemas a los que se les otorga capacidad de aprendizaje, entendiendo aprendizaje como la generalización del conocimiento a partir de experiencias. El propósito del *Machine Learning* es que una máquina sea capaz de llevar a cabo actividades o tareas para las que inicialmente no estaba programado, es decir, que las aprenda. Lo que diferencia el aprendizaje automático de cualquier otro tipo de máquina que no utiliza este tipo de técnicas, es que el *Machine Learning* permite realizar una tarea para la cual no estaba inicialmente programada.

Para que las máquinas sean capaces de aprender, las máquinas disponen de diversos algoritmos los cuales dotan a los ordenadores de la capacidad de identificar patrones en datos masivos (*Big Data*) y elaborar predicciones [27]. Lo que hace especial a este tipo de algoritmos es que son capaces de modificar su comportamiento en base a los datos de los que disponen, resultados de acciones pasadas o en lo que se les diga de esos resultados. Existen multitud de algoritmos que nos permiten llevar a cabo esta actividad. Sin embargo, en función de la forma de proceder del propio algoritmo, podemos establecer tres grandes grupos: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

2.2.1 Aprendizaje supervisado

El aprendizaje supervisado [28] es la rama del *Machine Learning* que se caracteriza por hacer uso de conjuntos de datos etiquetados con el fin de entrenar algoritmos que posteriormente sean capaces de clasificar datos de entrada o predecir resultados con precisión. Que los datos estén etiquetados quiere decir que son datos de entrada para los que se conoce la solución al problema que se plantea. Se trata de un proceso iterativo, es decir, que a medida que se van recibiendo datos de entrada en el modelo, este va aprendiendo y ajustando. Este proceso se conoce como entrenamiento. Un ejemplo sería un software destinado a distinguir en qué fotos aparece una persona concreta. El aprendizaje consistiría en pasarle una gran cantidad de fotos en las que aparezca la persona y otras en las que no, y se le dice en cuales aparece esa determinada persona y en cuáles no. A partir de esas fotos el algoritmo aprende y es capaz de determinar en qué fotos aparece la persona y en cuáles no.

Dentro de la rama del aprendizaje supervisado, existen dos tipos principales de aprendizaje: clasificación y regresión [29]. Pese a que ambos utilizan aprendizaje supervisado para entrenar algoritmos, existen ciertas diferencias en las tareas que realizan y el resultado que obtienen.

- Clasificación: se trata de algoritmos que son entrenados para clasificar los datos de entrada en variables discretas. Estos algoritmos pueden realizar tareas tanto de clasificación binaria, si solo clasifican entre dos variables como, por ejemplo, la detección de spam del correo electrónico ("spam" o "no spam"), como de clasificación multiclase, en caso de que existan más de dos variables para clasificar los datos de entrada, como por ejemplo un algoritmo que determine el tipo de un buque a partir de sus datos cinemáticos. También existe la posibilidad de crear algoritmos que asocien más de una clase a los datos de entrada, la llamada clasificación de etiquetas múltiples.
- Regresión: a diferencia de la clasificación, esta metodología trata de entrenar un algoritmo el cual estará destinado a predecir un valor de salida a partir de un rango continuo de valores posibles. Básicamente trata de encontrar una relación entre los datos de entrada y salida para ser capaz de predecir valores reales. A diferencia de la clasificación, los datos de salida no son discretos y la medición de la exactitud del algoritmo se calculará en función de la desviación del dato de salida obtenido y el esperado. Un ejemplo podría ser un algoritmo capaz de predecir la nota de corte de una oposición en función de las notas de selectividad de un año en concreto [30].

El aprendizaje automático supervisado sigue una serie de pasos para poder llevar a cabo su ejecución. A grandes rasgos, los pasos básicos del aprendizaje supervisado, que podemos ver de forma esquemática en la Figura 2-6, serían los siguientes [29]:

1. Selección de los datos de entrenamiento: determinar la naturaleza de estos.
2. Preprocesado de los datos: limpieza rigurosa de los datos para que el algoritmo los pueda usar correctamente.
3. Selección del algoritmo de aprendizaje supervisado a utilizar en el modelo: según la naturaleza de los datos y el resultado deseado, se seleccionará el algoritmo que se considere como el óptimo para la resolución de la tarea.
4. Entrenamiento del modelo, mediante los datos de entrada etiquetados para que el algoritmo se autoajuste y pueda ir ganando precisión mediante múltiples iteraciones de los datos de entrenamiento.
5. Prueba y evaluación del modelo: una vez esté ajustado, se prueba que el modelo funcione correctamente y se mide su precisión.

Este proceso finaliza una vez se obtiene un modelo que funcione. La degradación de los datos forma también parte del *Machine Learning*, por lo que, si queremos garantizar la precisión de nuestro modelo, debemos entrenarlo de forma periódica con datos actualizados.

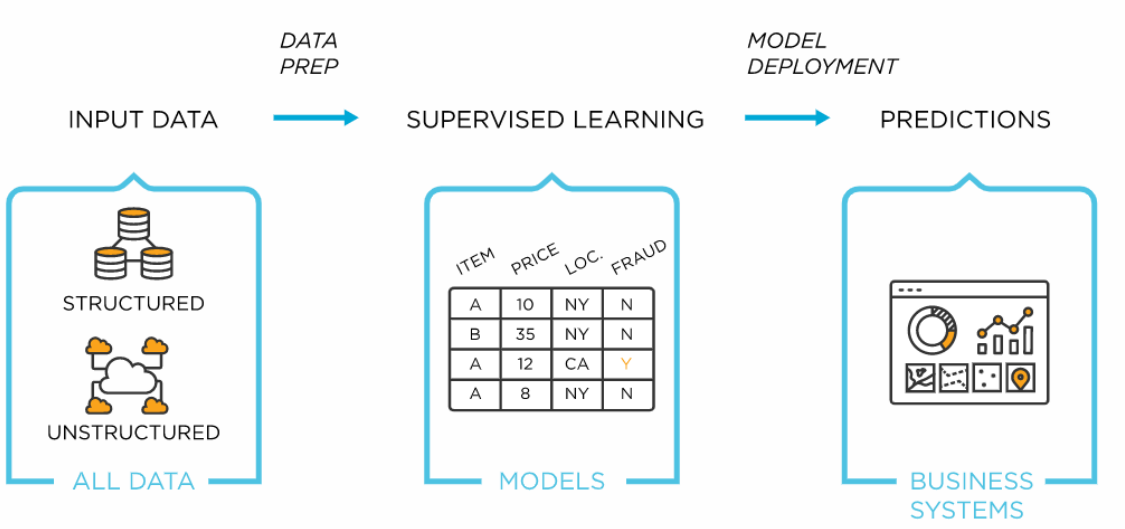

Figura 2-6 Esquema básico de las técnicas de aprendizaje supervisado [29]

El uso de este tipo de aprendizaje es muy extendido y lo podemos observar en aplicaciones de reconocimiento facial, reconocimiento de voz, reconocimiento de huellas digitales, coches automáticos y otras muchas más aplicaciones que se encuentran en dispositivos que usamos diariamente. El único inconveniente es que requiere de una gran cantidad de datos etiquetados. De hecho, muchas personas contribuyen diariamente con esta metodología, muy probablemente, sin ser consciente de ello. Se puede ver en los CAPTCHA como los de la Figura 2-7 que aparecen para acceder a determinadas páginas web. En estos se solicita al usuario que marque las fotos en las que aparecen coches, semáforos, señales o cualquier otro objeto. Pero lo que está haciendo el usuario realmente, además de servirle al sistema para comprobar que el usuario no es una máquina sino un humano, es etiquetar fotos que posteriormente servirán para entrenar algoritmos que reconozcan estos objetos y que, por ejemplo, más adelante puedan estar instalados en un vehículo autónomo [31].

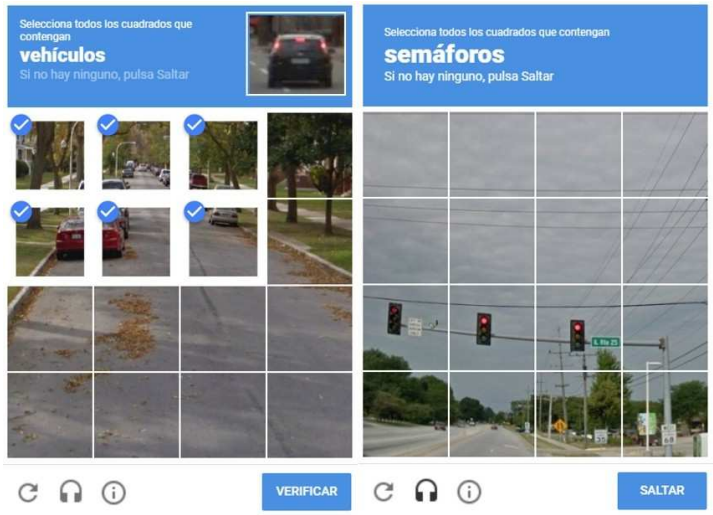

Figura 2-7 Ejemplo de CAPTCHA [32]

2.2.2 Aprendizaje no supervisado

El aprendizaje no supervisado hace referencia al conjunto de técnicas que existen en el campo del *Machine Learning* que tratan de inferir modelos a partir de unos datos de entrada de los que desconocemos la salida deseada a priori. Los algoritmos de aprendizaje no supervisado tratan de encontrar características similares entre los datos de entrada para realizar una clasificación correcta utilizando patrones y estructuras que comparten dichos valores y que, a simple vista, los humanos no

percibimos [33]. Al contrario que en el aprendizaje supervisado, los modelos que aplican el aprendizaje no supervisado se entrenan a partir de datos de entrada no etiquetados.

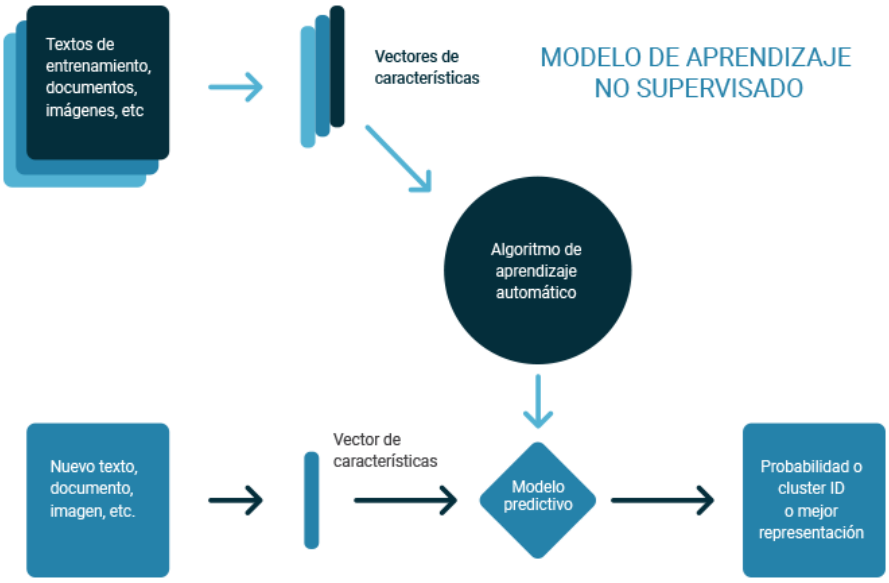

Figura 2-8 Diagrama de flujo del aprendizaje no supervisado [34]

El funcionamiento de esta metodología se puede explicar de manera superficial haciendo uso de la Figura 2-8. Como podemos observar, partimos de una base de datos que puede estar compuesta por textos, documentos e imágenes de los cuales se extraen los vectores de características que los definen. El algoritmo se entrena a partir de las características o patrones que ha observado en los datos de entrenamiento, obteniéndose finalmente un modelo cuando el algoritmo esté ajustado. De esta forma, cuando introduzcamos un nuevo texto, documento o imagen en nuestro modelo, se extraerán las características de este nuevo dato y se le asignará a un conjunto que posea características similares. En la Figura 2-9 se proporciona un ejemplo más práctico en el que los datos de entrenamiento son diferentes frutas. Además, podemos ver la secuencia recién descrita y cómo clasifica a la salida del modelo las frutas con características semejantes.

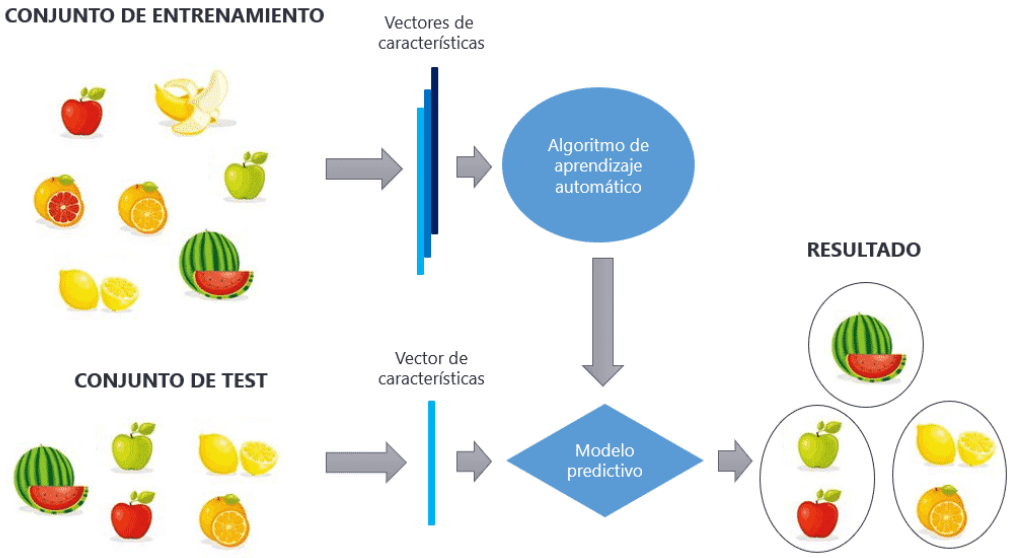

Figura 2-9 Ejemplo de aprendizaje no supervisado [35]

Este tipo de técnicas se suele emplear mayoritariamente para agrupar datos que recogen similitudes entre sí. Un claro ejemplo y que es a la vez uno de los usos más extendidos de esta rama del *Machine Learning* lo podemos en el *marketing*. Lo que realizan los algoritmos en este ámbito, a grosso modo, es agrupar a conjuntos de personas con gustos, características y hábitos similares para después enfocar en ellos la publicidad adecuada, en otras palabras, ofrecerle a cada persona productos en los cuales esta ciertamente interesado.

Los algoritmos de aprendizaje no supervisado son muy variados. Según los problemas que están destinados a solventar, podemos englobar los algoritmos, principalmente, en los siguientes tipos: agrupamiento, detección de anomalías, asociación y reducción de la dimensionalidad.

- Agrupamiento (*Clustering*): tienen la finalidad de clasificar en grupos los datos no etiquetados para formar subconjuntos de los mismos llamados *clusters*. Su aplicación está bastante extendida en el mundo real, como por ejemplo en análisis de redes sociales, segmentación de clientes, clasificación de especies y clasificación de enfermedades. Existen multitud de algoritmos que se utilizan para realizar la tarea de *clustering*, los cuales estudiaremos más en detalle más adelante. Un ejemplo que ayuda a la comprensión de su funcionamiento: si los datos de entrada fueran fotos de gatos y perros, el algoritmo agruparía las fotos en dos *clusters*, uno para las fotos de perros y otro para las de gatos.
- Detección de anomalías: consiste en utilizar algoritmos de aprendizaje no supervisado para analizar, reconocer y buscar patrones que permitan detectar anomalías, entiendo por anomalía una desviación de una regla o un defecto de funcionamiento [36].
- Asociación: este método trata de ordenar los datos en base a atributos que comparten. El funcionamiento de estos algoritmos consiste en encontrar las relaciones que hay entre los objetos, sin necesidad de que exista parecido entre ellos. Pese a que pueda parecer un concepto similar al agrupamiento, la forma de trabajar por parte del algoritmo los diferencia. En el ejemplo anterior de las fotos de perros, a diferencia de la clasificación, que agruparía a todos los perros juntos, mediante asociación lo que haría el algoritmo es, por ejemplo, asociar las correas con los perros.
- Reducción de dimensionalidad: la dimensionalidad es el conjunto de variables que tienen los datos. Estas características están representadas como atributos y el objetivo es la reducción de número de estos. En muchas ocasiones, la información de estos atributos está relacionada, dando como consecuencia una redundancia que añade ruido a los datos, es decir, añade información inservible que puede disminuir el rendimiento de nuestro modelo. Este tipo de técnicas asumen que una gran cantidad de datos es redundante y eliminan dimensiones o las combinan, según corresponda, con el fin de que la compresión de datos resulte más sencilla y tenga un menor coste computacional al existir menos atributos. Dentro de este tipo de problemas podemos encontrar dos categorías principales: la selección de variables, que consiste en la selección de un subconjunto de características del *dataset* inicial, y la extracción de características, que trata de generar atributos derivados de los iniciales con la intención de generar unas características informativas y poco redundantes.

2.2.3 Aprendizaje semisupervisado

Como se ha expuesto en los apartados previos, los algoritmos de aprendizaje supervisado utilizan datos etiquetados para desarrollar su entrenamiento, mientras que los algoritmos de aprendizaje no supervisado utilizan datos no etiquetados. Las técnicas de aprendizaje semisupervisado hacen uso tanto de datos etiquetados como datos no etiquetados.

El concepto nace a raíz de intentar solventar las deficiencias que presentaban tanto el aprendizaje supervisado como el no supervisado. El aprendizaje supervisado es más preciso, pero presenta el problema de que su coste computacional suele ser muy elevado, principalmente por el etiquetado de los datos que requiere del esfuerzo de trabajadores con experiencia y formación en el dominio, y los otros costes que pueden llegar a ser muy elevados como la limitación de tiempo y los recursos financieros

[37]. Por otro lado, el aprendizaje no supervisado no ofrece tanta precisión como el supervisado, pero es una metodología más sencilla y su coste tanto computacional como temporal es menor al usar datos no etiquetados. En este escenario, el aprendizaje semisupervisado aprovecha la precisión de los algoritmos de aprendizaje supervisado y trata de mejorar su coste computacional con el uso de datos no etiquetados.

Existen muchos algoritmos de aprendizaje semisupervisado pero, de forma genérica, el funcionamiento de los mismos se basa en el concepto que se aprecia en la Figura 2-10. Inicialmente, se dispone de un *dataset* formado por un conjunto de datos etiquetados y otro conjunto de datos no etiquetados, en el que el primero de los conjuntos representa la minoría de los datos. Los datos etiquetados se utilizan para entrenar un modelo mediante técnicas de aprendizaje supervisado. Este modelo se utiliza para predecir las etiquetas de los datos no etiquetados. Los nuevos datos etiquetados junto con el conjunto de datos etiquetados que existían en el *dataset* inicial conforman un nuevo *dataset* que se utilizará para reentrenar un modelo final mediante técnicas de aprendizaje supervisado [28].

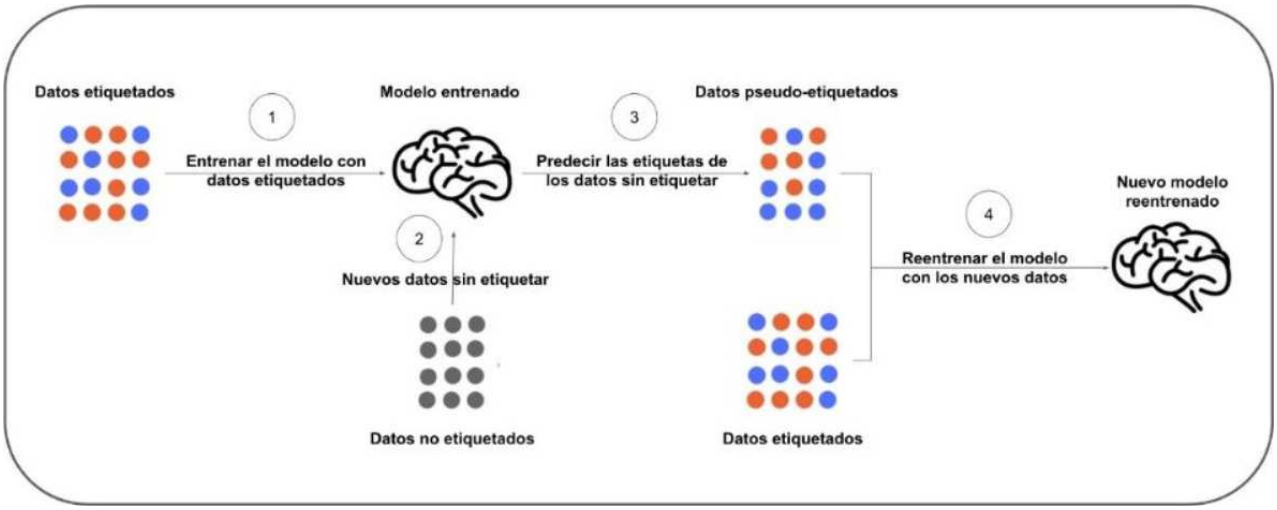

Figura 2-10 Esquema del proceso de aprendizaje semisupervisado [38]

Los modelos de aprendizaje semisupervisado son cada vez más usados en la actualidad, especialmente en tareas en las que el análisis de los datos resulta una tarea complicada y que suponen un elevado coste de recursos, como por ejemplo el análisis de archivos de audio.

2.2.4 Aprendizaje por refuerzo

El aprendizaje por refuerzo es una técnica dentro del campo del *Machine Learning* que se caracteriza por el empleo de la técnica de prueba y error por parte del algoritmo para desarrollar su entrenamiento. Lo que busca este tipo de aprendizaje es que la máquina sea capaz de tomar una secuencia de decisiones que le lleven a realizar la tarea para la que se ha entrenado con éxito.

Para dar una definición más precisa de aprendizaje reforzado y describir su funcionamiento, es necesario previamente definir un conjunto de términos de uso frecuente en esta rama del *Machine Learning*:

- Agente: es el programa que se va a entrenar y que posteriormente será el encargado de la toma de decisiones para realizar la tarea requerida.
- Entorno: es el ambiente, real o virtual, en el que opera el programa.
- Acción: son los movimientos realizados por el agente dentro del entorno, provocando un cambio en el estado del entorno.
- Estado: es la situación en la que se encuentra el entorno cada vez que cambia como consecuencia de que el agente realice una acción.

- Recompensa: es la retroalimentación que le proporciona el entorno al agente para valorar la acción que haya realizado. Es la encargada de hacer saber al algoritmo que la decisión tomada es correcta o errónea. La recompensa puede ser positiva o negativa.

Por ejemplo, si quisiéramos asociar los elementos citados con anterioridad a una situación real, como una máquina con capacidad de jugar al ajedrez, el agente sería el programa que mueve las fichas; el entorno consistiría en el tablero de ajedrez; la acción sería el conjunto de los posibles movimientos de las fichas que el agente puede realizar; el estado sería la situación en la que se encuentra el tablero y las fichas en un momento determinado. El estado cambia cada vez que se mueva una ficha; y la recompensa sería ganar o perder la partida, que se asociaría a recompensa positiva y negativa, respectivamente.

Una vez aclarados estos conceptos, ya se puede dar una explicación más técnica de aprendizaje por refuerzo y su funcionamiento.

En el aprendizaje por refuerzo un agente aprende por prueba y error, y en base a la experiencia, aprende a realizar la tarea. Por lo tanto, podemos definir el aprendizaje por refuerzo como un tipo de método de aprendizaje automático en el que el agente interactúa con el entorno, por medio de acciones, y aprende a actuar en él a través de la experiencia [39]. En función del estado, el algoritmo tomará una serie de decisiones y, en caso de que estas le hagan completar la tarea con éxito o no, el algoritmo adquirirá una recompensa positiva o negativa. Además, el diseñador le puede imponer al agente un conjunto de reglas que debe cumplir durante la realización de la tarea.

La clave del entrenamiento de este tipo de máquinas reside en la experiencia, no trabaja con datos etiquetados como hemos visto en otras técnicas de Machine *Learning*. Para que la experiencia funcione, es necesario que exista una retroalimentación fluida entre el agente y el entorno, tal y como se ve en la Figura 2-11, en la que el agente realiza las acciones en el entorno y, al realizarlas, el entorno le devuelve una recompensa, ya sea positiva o negativa, y su estado, para que el agente pueda tomar la decisión de cuál será su siguiente acción para realizar la tarea.

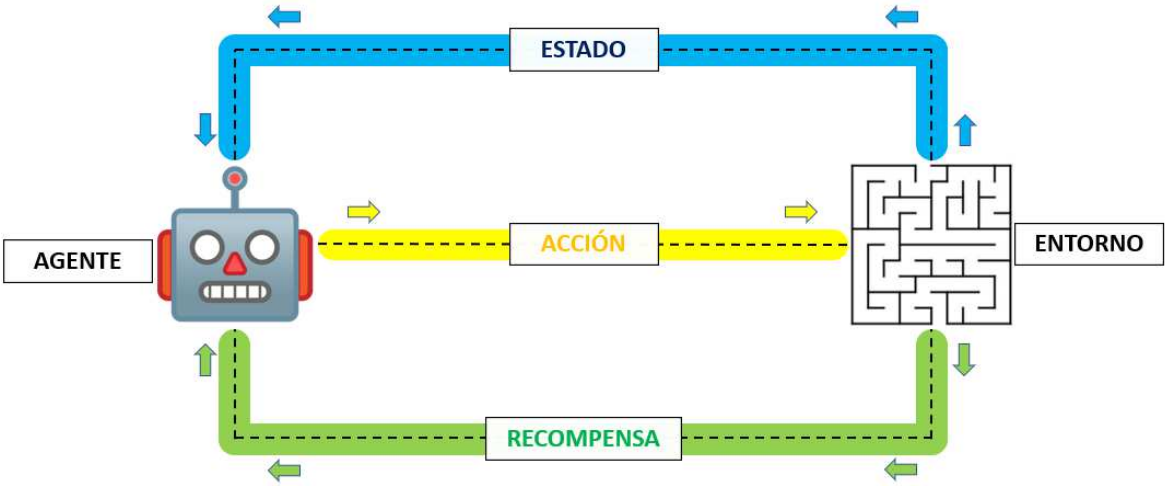

Figura 2-11 Esquema básico del aprendizaje por refuerzo (fuente: propia)

El empleo de este tipo de técnicas se utiliza en tareas muy variadas. Entre estas, podemos destacarla como la metodología para enseñar a jugar a una máquina o las aplicaciones científicas. Gracias a este tipo de aprendizaje se ha conseguido crear máquinas capaces de jugar a juegos y, en ciertos casos, llegar a superar a los humanos en los mismos.

El aprendizaje por refuerzo es sin duda una tecnología de vanguardia y que tiene un gran potencial con vistas al futuro. Pese a que no se trata de un algoritmo que se pueda aplicar a todo tipo de problemas, su importancia es cada vez mayor debido a los grandes avances obtenidos. Actualmente el aprendizaje por refuerzo es la forma que más se aproxima para dotar de creatividad a una máquina, pues trata de buscar nuevas formas de realizar la tarea hasta que haya una adecuada [40]. Sin embargo, el hecho que

ha supuesto un mayor impulso para este tipo de aprendizaje es la aplicación de técnicas de *Deep Learning* combinadas con el propio aprendizaje por refuerzo, utilizando una red neuronal para almacenar la experiencia obtenida por la máquina durante su funcionamiento. Nace así lo que se denomina el aprendizaje por refuerzo profundo, consiguiéndose con su aplicación la creación de máquinas y algoritmos con capacidades sin precedentes entre los que podemos destacar *AlphaGO*, una inteligencia artificial, que se puede apreciar en la Figura 2-12, desarrollada por Google capaz de ganar a jugadores profesionales de GO, un juego popular chino que destaca por ser sumamente complicado [41].

Figura 2-12 Inteligencia artificial *AlphaGo* jugando contra un jugador profesional de Go [42]

2.2.5 Deep Learning

Deep Learning es un subconjunto del *Machine Learning* que utiliza grandes cantidades de datos y numerosos algoritmos organizados en capas para el empleo de técnicas que doten a los ordenadores de la capacidad de aprender de forma independiente y realizar tareas propias de los seres humanos, como por ejemplo la identificación de imágenes, el reconocimiento del lenguaje e incluso la capacidad de predecir variables [43]. En la Figura 2-13 se puede apreciar cómo se encuadra esta tecnología en el marco de la inteligencia artificial, además de ver que se trata de una tecnología cuyo apogeo se ha alcanzado de forma reciente.

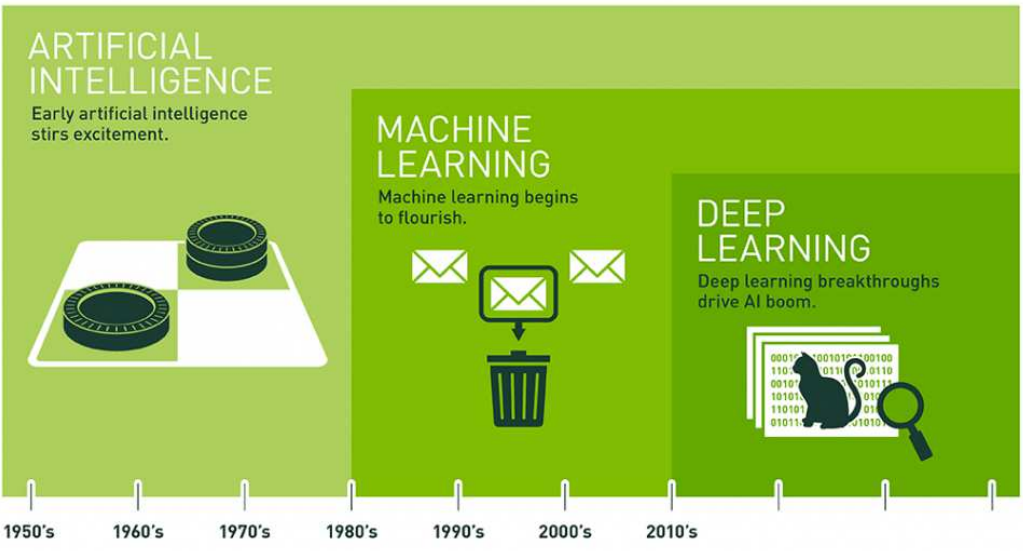

Figura 2-13 *Deep Learning* y *Machine Learning* como subconjuntos de la inteligencia artificial [44]

El concepto que caracteriza al *Deep Learning* con respecto de cualquier otro de tipo de *Machine Learning* es el uso de las redes neuronales.

Las redes neuronales son probablemente la técnica más potente que existe actualmente en el campo del *Machine Learning*. Las redes neuronales consisten en muchas pequeñas funciones matemáticas, cada una de ellas llamada neurona, que se combinan y coordinan entre sí, pasándose resultados de las funciones que desempeñan unas a otras, formando una red. Cuando estas redes son grandes y están formadas por muchas capas adquieren la denominación de redes profundas.

Las redes neuronales profundas tratan de simular la estructura del cerebro humano y su forma de aprender. Están formadas por múltiples capas de neuronas interconectadas en las que las capas más bajas aprenden los conceptos más básicos y conforme se va avanzando por las capas superiores encontramos conceptos más complejos que son aprendidos por el modelo con la ayuda de los conocimientos adquiridos en las capas inferiores. De esta forma, las redes neuronales son capaces de realizar un aprendizaje de forma jerarquizada, es decir, que la información se aprende por niveles. Los niveles más bajos tratan de aprender conceptos simples, como por ejemplo qué es una rueda o un tornillo, y las capas superiores se enfocan en aprender conceptos más abstractos a partir de lo que han aprendido las capas inferiores, en nuestro ejemplo podría ser un coche. Cuantas más capas tenga la red neuronal, más abstractos serán los conceptos que puede llegar a aprender [45].

A nivel global, existen dos tipos de capas: las capas visibles, donde se produce la entrada y salida de datos y que los humanos podemos comprender, y las capas ocultas, que son aquellas que contienen valores no observables y mediante la interconexión de las cuales se produce el aprendizaje. Asimismo, estas capas se organizan en tres niveles [46] a la hora de funcionar como una red neuronal tal como se ve en la Figura 2-14:

- La capa de entrada: es la parte del modelo por la que se ingieren los datos para su posterior procesamiento. Las neuronas de esta capa reciben datos o señales procedentes del entorno.
- Las capas ocultas: estas capas no tienen conexiones directas con el entorno. Estas capas son las que permiten a la red neuronal aprender. Las capas ocultas adicionales pueden favorecer la optimización y obtención de resultados más precisos.
- La capa de salida: está compuesta por las neuronas que proporcionan la salida de la red neuronal.

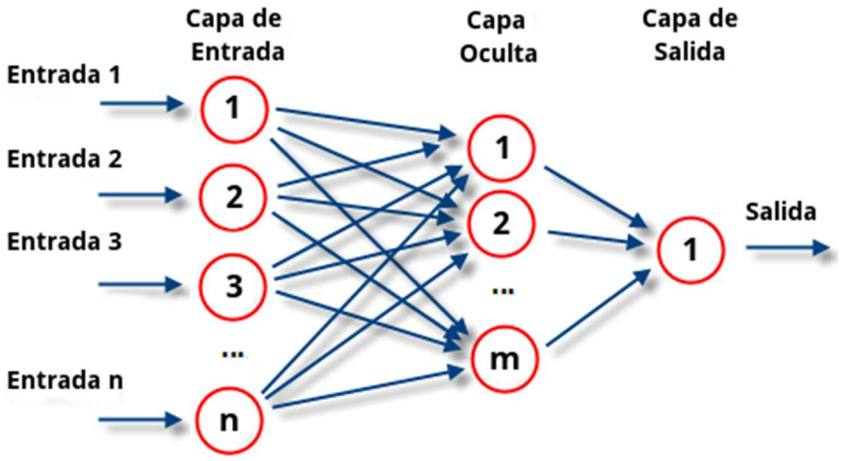

Figura 2-14 Esquema simplificado de las capas de una red neuronal [47]

Las aplicaciones de los algoritmos de *Deep Learning* abarcan un campo sumamente amplio. Entre estas aplicaciones destacan la visión artificial, que proporciona la capacidad a las máquinas de reconocer objetos, caracteres, rostros e imágenes del mundo real; previsiones de mercado; comprensión del lenguaje, como por ejemplo para la traducción de textos automáticos; asistentes virtuales, como Siri, Cortana o Alexa; robótica, para que los robots sean capaces de aprender a realizar tareas humanas y

comportarse como ellos; para tareas sanitarias, facilitando detección de enfermedades y diagnósticos por ordenador; en noticias y publicidad, para ofrecer al usuario anuncios y noticias de acuerdo con sus intereses; y servicios de recomendación de contenido multimedia como Netflix, Spotify o YouTube, para ofrecer contenido de acuerdo con los gustos del usuario [43].

Además, actualmente nos encontramos inmersos en la era de la información. La llegada de la digitalización, la bajada de precio de los dispositivos de almacenamiento y un cambio de mentalidad a la hora del almacenamiento de datos, han tenido como consecuencia la aparición del fenómeno *Big Data*. Las técnicas de *Deep Learning* resultan muy útiles para trabajar con la gran cantidad de datos que se manejan hoy en día, dando un nuevo resurgir al campo del *Machine Learning*, y, por ende, al campo de la inteligencia artificial.

2.2.6 Campos de aplicación

Como se ha ido exponiendo a lo largo del presente documento, las aplicaciones del *Machine Learning* son tan amplias que actualmente podemos encontrarla en una gran diversidad de campos. A continuación, se expondrán algunos de estos campos, enfocándose en los que tienen más transcendencia en las áreas que afectan a nuestra vida cotidiana [48].

2.2.6.1 Educación

Cuando se aplica el *Machine Learning* a este campo, se realiza principalmente para llevar un control del aprendizaje del alumnado de forma automática. Existen algoritmos capaces de evaluar las lecciones impartidas a los alumnos, mediante los cuales se puede realizar corrección de tareas u observación de los temas de mayor o menor dificultad para el alumnado con el fin de ayudar al profesorado en la programación de las lecciones.

2.2.6.2 Motores de búsqueda

En este ámbito el *Machine Learning* está aplicado a muchas más funciones de las que uno se puede imaginar a simple vista. En primer lugar, si uno tiene en mente el buscador de Google, este despliega un conjunto de opciones conforme escribimos para tratar de ahorrar tiempo prediciendo lo que vamos a escribir. En esta tarea, el *Machine Learning* trata de hacer que estas opciones sean cada vez más precisas. Además, cuando buscamos imágenes, son técnicas de *Machine Learning* con las que se entrenan los algoritmos que asocian lo que escribe el usuario que quiere buscar con la imagen que busca. De esta forma, si una persona escribe “coche” en el buscador, este será capaz de entender lo que es un coche y mostrarnos imágenes de estos. También se han potenciado los asistentes de voz, como Siri o Cortana, e incluso se han llegado a crear aplicaciones capaces de identificar los objetos de una imagen para realizar su búsqueda, como *Google Lens* [49]. Podemos ver ejemplos diferentes usos de esta aplicación en la Figura 2-15

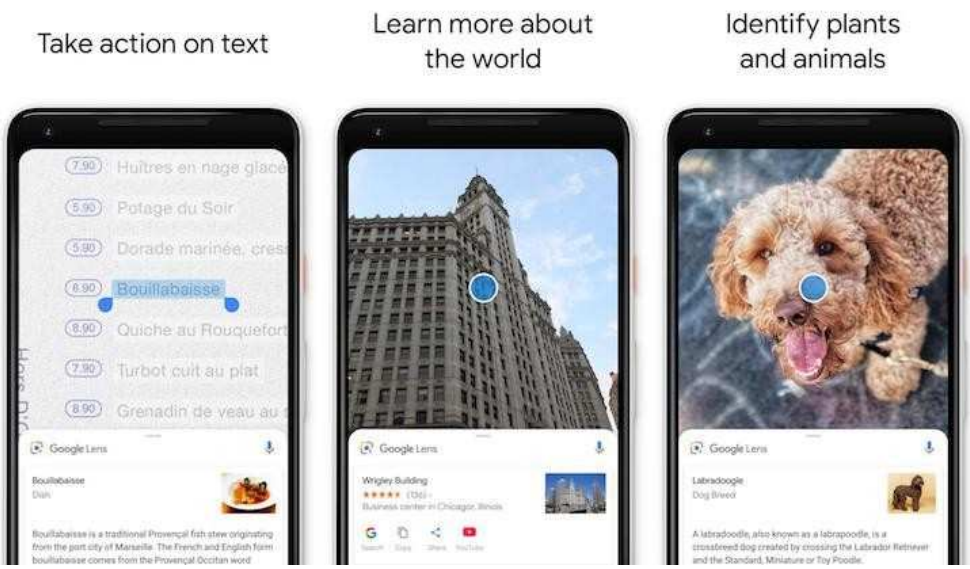

Figura 2-15 *Google Lens*, técnicas de *Machine Learning* aplicadas a un motor de búsqueda [50]

2.2.6.3 Marketing

Es uno de los campos que ya se ha mencionado anteriormente. Gracias a estas aplicaciones las empresas son capaces de personalizar la publicidad y las sugerencias que se muestran en nuestros dispositivos para adaptarla a nuestros gustos en base a las búsquedas que realizamos, compras o contenido multimedia que consumimos.

2.2.6.4 Medicina

En esta rama la integración de la inteligencia artificial mediante *Machine Learning* se aplica sobre todo a mejorar la eficiencia a la hora del diagnóstico de enfermedades y su tratamiento, reduciendo costes y evitando posibles errores humanos. Actualmente, ya existen programas y *chatbots* que atienden a los pacientes y son capaces de diagnosticarlos y hacerles un seguimiento.

Sin embargo, su aplicación va mucho más allá, siendo posible la creación de programas que realicen una detección de células cancerígenas de nuestro cuerpo antes de que se desarrolle el cáncer. Con lo que se podría decir que gracias a las técnicas de *Machine Learning* se ha conseguido prolongar la vida de muchas personas.

Además, también se han desarrollado Inteligencias Artificiales capaces de predecir la aparición de brotes epidémicos y su impacto en el mundo en base a datos históricos. La predicción de la gravedad de los brotes es de vital importancia en países del tercer mundo, donde los medios disponibles para combatirlos escasean.

2.2.6.5 Industria

Posiblemente, sea el campo en que más se ha popularizado el uso de tecnologías entrenadas a partir de técnicas de aprendizaje automático. Precisamente, en el libro escrito por el empresario alemán Klaus Schaub "*The Fourth Industrial Revolution*" [51] se establece que uno de los hechos que marca el inicio de la cuarta revolución industrial es la aparición de las fábricas inteligentes.

Las técnicas de inteligencia artificial y, más en concreto, de *Machine Learning* se han convertido en recursos indispensables para el desarrollo del término conocido actualmente como Industria 4.0. La increíble cantidad de datos que maneja este concepto hace que la IA y *Big Data* se conviertan en herramientas casi indispensables para las empresas.

Además, si nos referimos a la aplicación en las fábricas, actualmente podemos observar cómo cada vez son más las fábricas que automatizan sus procesos. Los trabajadores humanos traen consigo sueldo,

cansancio y riesgo al llevar a cabo actividades peligrosas. Las máquinas suponen para las empresas ahorro del coste de la mano de obra, no les hace falta descansar, son más precisas y pueden realizar actividades que para los humanos resultarían peligrosas.

2.2.6.6 Finanzas

Las finanzas son uno de los primeros campos en los que se puede apreciar la aparición de estas tecnologías. Esto se debe a que son sistemas que requieren de plataformas intuitivas para realizar predicciones y estudios del movimiento de los mercados. Además, cobran también importancia a la hora de revisar a los usuarios y verificar y controlar trámites y transacciones.

2.2.6.7 Robótica

En este campo es donde se han logrado avances de lo más sorprendente, y es que con el paso de los años y las investigaciones en estas tecnologías se ha acabado consiguiendo que las máquinas sean capaces de acometer tareas que, hasta no hace muchos años, resultaba imposible pensar que las máquinas pudiesen realizar.

Existen muchos ejemplos conocidos por todo el mundo. Asimo, al que podemos ver en el centro de la Figura 2-16, es un robot creado por Honda en el año 2000 capaz de caminar, manipular objetos y mantener el equilibrio. Además, responde a estímulos exteriores comportándose como si de una persona se tratara. Otra conocida empresa es Boston Dynamics, la cual lleva ya años desarrollando robots que realizan actividades de lo más variopintas como bailar, acrobacias o manipular objetos, como por ejemplo el robot que se puede ver en la parte derecha de la Figura 2-16.

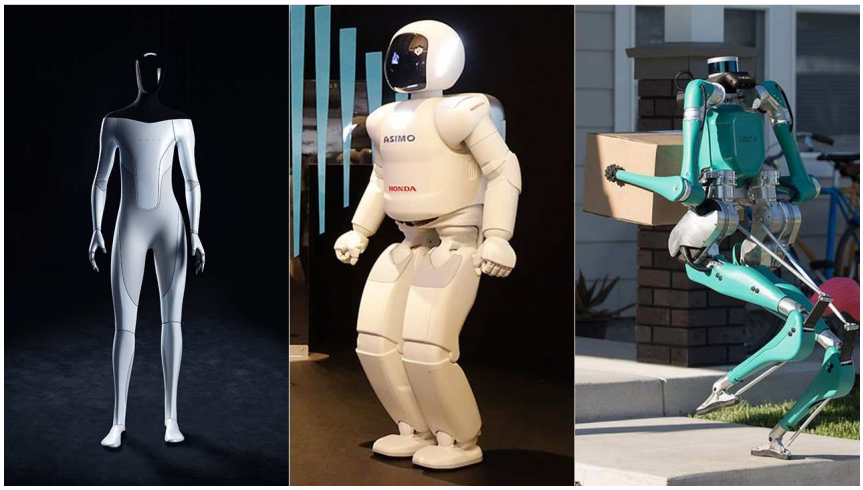

Figura 2-16 Ejemplos de IA aplicados a la robótica [52]

El *Machine Learning* aplicado a la robótica es una tecnología en la que se están invirtiendo grandes cantidades de dinero y realizando investigaciones por parte de las empresas de todo el mundo. El futuro de este campo es de lo más prometedor, y ya podemos ir observando cómo se acerca este futuro en inventos como el Tesla Bot desarrollado por Tesla y que se puede ver en la parte izquierda de la Figura 2-16, el cual todavía está en su fase de desarrollo y no se espera su lanzamiento hasta 2023, pero ya se muestra como un humanoide de características muy similares a las humanas.

2.3 Algoritmos de aprendizaje supervisado

Habiéndose ya expuesto los principales tipos de aprendizaje, su funcionamiento y las utilidades que tienen en el mundo real, en este apartado se procede a explicar de forma resumida algunos de los algoritmos de aprendizaje supervisado más usados en el campo del *Machine Learning*.

2.3.1 Árboles de decisión

Los árboles de decisión son una técnica de aprendizaje supervisado que se puede utilizar para problemas tanto de regresión como de clasificación. Sin embargo, estas técnicas se emplean mayoritariamente para tareas de clasificación. Esta metodología puede asimilarse a un clasificador que sigue una estructura similar a un diagrama de flujo para la toma de decisiones. Previamente a explicar en qué consiste este algoritmo, se deben tener claros ciertos conceptos básicos:

- Nodo raíz: es el primer nodo del árbol de decisión. Representa el conjunto de datos al completo que posteriormente se dividirá en dos o más grupos homogéneos conforme los datos avancen por el árbol.
- Nodo hoja: nodos los cuales, una vez alcanzan los datos, no generan más divisiones en el conjunto. Son los nodos de salida del árbol.
- División: es el proceso de dividir el nodo raíz o cualquier otro nodo en base a una condición establecida.
- Rama/Subárbol: se denomina rama o subárbol al árbol resultante de dividir otro árbol.
- Nodo principal/secundario: se denomina nodo principal al nodo del cual nacen el resto de los nodos, que son los nodos secundarios. El nodo principal sería el nodo raíz, mientras que los nodos secundarios serían nodos hoja o nodos en los que se vuelve a llevar a cabo una división de los datos, los denominados nodos de decisión.

Básicamente, en lo que consiste un árbol de decisión es en la entrada de un conjunto de datos a través del nodo raíz. Cuando el árbol recibe un nuevo dato de entrada, este será conducido a través de una rama u otra en función de las características del dato de entrada hasta alcanzar un nodo hoja [53]. El nodo hoja alcanzado será el que determine la salida. Un ejemplo de árbol de decisión lo podemos ver en la Figura 2-17. Lo que se observa es un árbol de decisión que en función de las condiciones meteorológicas de un día en concreto determina si se saldrá o no a navegar. En el nodo raíz los datos de entrada serían los días, cada uno con su oleaje y viento determinado. Lo que hace el árbol es establecer una condición sobre estas características para dividir los datos, como en el ejemplo que el oleaje sea mayor o menor de 3 metros. De esta división puede acabar en un cuadro hoja (cuadrado verde en el ejemplo) el cual proporcionará una salida u otro nodo de decisión donde repetirá la secuencia descrita hasta alcanzar un nodo hoja.

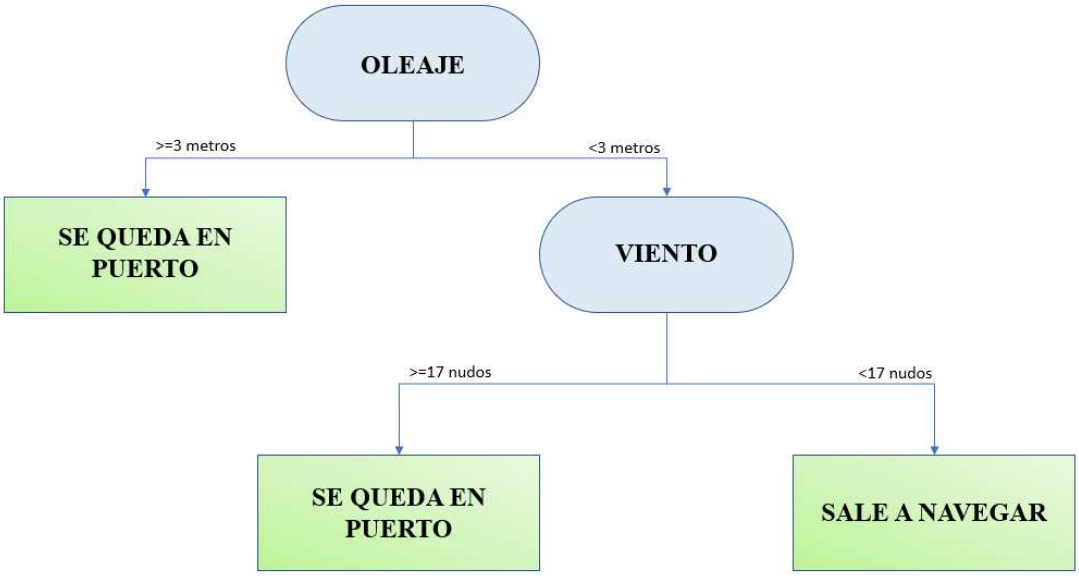

Figura 2-17 Ejemplo de árbol de decisión (fuente: propia)

En muchas ocasiones, nos puede interesar eliminar ramas del árbol de decisión porque resultan redundantes o empeoran el funcionamiento del árbol. Para ello existe el proceso denominado poda, que consiste en eliminar las ramas del árbol no deseadas.

La ventaja de los árboles de decisión es que tratan de imitar cómo razona el pensamiento humano ante una decisión, por lo que es una metodología que resulta fácil de comprender. Lo complicado a la hora de implementar esta técnica es generar el propio árbol y, más concretamente, tomar la decisión de qué características o atributos utilizaremos para realizar la división de los datos, los que serían en nuestro ejemplo de la Figura 2-17 el oleaje o el viento. Para esta elección se dispone de dos métodos principalmente: Ganancia de información y Gini.

- Ganancia de información: este método se basa en el cálculo de la entropía de los subconjuntos resultantes de una división en función de un atributo. La entropía es una medida que representa la aleatoriedad de los datos. Existen fórmulas matemáticas que utilizan la entropía de los subconjuntos resultantes de una división para medir la ganancia de la información de cada división. Las divisiones con una ganancia de la información mayor o, en otras palabras, las que generen nodos más puros, irán en la parte alta del árbol. Lo que se busca es que se formen subconjuntos diferenciados de elementos similares entre sí, pero diferentes con respecto al resto de subconjuntos.
- Gini: es un índice entre 0 y 1, que se calcula en base a la probabilidad de extraer dos elementos de un conjunto y que sean de la misma clase. Si lo son, el índice es igual a 1. Se deben preferir atributos que posean un índice de Gini bajo frente a uno alto. De esta forma los atributos con menor índice de Gini estarán más cerca del nodo raíz. Este método presenta el problema de que solo es válido para decisiones binarias.

2.3.2 Random Forest

Los árboles de decisión poseen ciertas limitaciones que hacen que sean muy buenos algoritmos para los datos de entrenamiento, pero cuando intentamos emplear el algoritmo ya entrenado con otros datos genéricos, pierden precisión. Esto se debe a que los árboles de decisión tienden a sobre ajustarse (*overfit*). *Random Forest* surge para solucionar todas estas deficiencias, reduciendo el sobreajuste y aumentando la precisión [54]. El sobreajuste consiste en la incapacidad de un modelo de generalizar patrones sobre los datos debido al empleo de excesivos datos anómalos de entrenamiento, lo que provoca que modelos que en primera instancia están bien entrenados no sean capaces de clasificar los nuevos datos de entrada.

Random Forest es una técnica de aprendizaje supervisado que se utiliza ampliamente en problemas de clasificación y de regresión. *Random Forest* pertenece a una familia de algoritmos llamados métodos de ensamble, que se caracterizan por construir varios modelos de *Machine Learning* al mismo tiempo para mejorar la precisión. Está muy relacionado con los árboles de decisión, por lo que para entender cómo funciona *Random Forest* es necesario tener claro el concepto de árbol de decisión que se ha expuesto en el apartado anterior.

Este algoritmo consiste en la creación de *n* árboles de decisión aleatorios no podados. Estos árboles se crean mediante la selección de forma aleatoria de conjuntos de datos de entrenamiento que son extraídos de un *dataset* que contenga todos los datos. Estos conjuntos se denominan *Bootstrap Dataset* y por cada uno de ellos se generará un árbol de decisión diferente. Los *Bootstrap Dataset* contienen datos que desconocemos, incluso pueden contener datos repetidos. Este proceso de creación de árboles mediante la agrupación de datos aleatorios se denomina *bagging.* Además, el usuario selecciona las características que quiere que utilice el modelo para la construcción de los árboles que determinarán las divisiones. De esta forma se consigue que los árboles generados sean aleatorios unos de otros. Por otro lado, se reservará un conjunto de datos del *dataset* original que serán los denominados datos de test. Estos datos no estarán presentes en ningún *Bootstrap Dataset* y nos servirán para evaluar el modelo una vez esté entrenado.

Una vez se ha creado y entrenado el número de árboles que se desea, el funcionamiento de este algoritmo para realizar tareas de clasificación y regresión a la llegada de nuevos datos es muy simple. Se introduce el dato de entrada en cada uno de los árboles generados durante la fase de entrenamiento. Cada árbol se ejecutará internamente para dar una salida, tal y como se puede ver en el ejemplo de la Figura 2-18. La predicción asociada al valor de entrada será el valor más votado por todos los árboles, en caso de tratarse de un problema de clasificación, y un valor promedio de salida de cada árbol, en caso de tratarse de un problema de regresión [55]. En nuestro ejemplo, al tratarse de un problema de clasificación, la salida de ejecutar *Random Forest* sería 1.

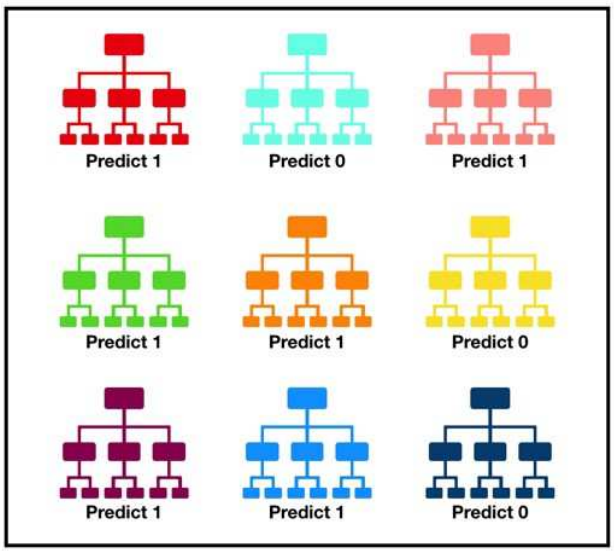

Figura 2-18 Ejemplo esquemático del funcionamiento de *Random Forest* [56]

Random Forest tiene la ventaja frente a otras técnicas de aprendizaje supervisado de que es capaz de realizar tareas de clasificación y regresión de una forma fácil de comprender, como los árboles de decisión, pero de una forma más precisa y siendo capaz de trabajar con grandes volúmenes de datos. El único inconveniente de *Random Forest* con respecto de los árboles de decisión es que requiere de un mayor coste computacional y tiempo de ejecución [57]. En la Tabla 2 se puede ver una comparación de ambos algoritmos:

Árboles de decisión	***Random Forest***
1.Los árboles de decisión normalmente sufren el problema de sobreajuste si se les permite crecer sin ningún control.	1.Los bosques aleatorios se crean a partir de subconjuntos de datos y el resultado final se basa en una clasificación promedio o mayoritaria y, por lo tanto, se soluciona el problema del sobreajuste.
2.Un solo árbol de decisión es más rápido en el cálculo.	2.Es comparativamente más lento.
3.Cuando un conjunto de datos con características se toma como entrada de un árbol de decisión, formulará un conjunto de reglas para hacer predicciones.	3.El bosque aleatorio selecciona observaciones al azar, construye un árbol de decisión y se toma el resultado promedio. No utiliza ningún conjunto de fórmulas.

Tabla 2 Comparativa entre *Random Forest* y árboles de decisión [57]

2.3.3 Gradient Boosting

Gradient Boostig es una técnica de aprendizaje supervisado que se puede utilizar para resolver tanto problemas de clasificación como de regresión. Al igual que *Random Forest*, esta metodología se engloba dentro de los conocidos como métodos de ensamble.

Gradient Boosting trata de combinar modelos débiles para obtener un modelo más fuerte y preciso. Utiliza árboles de decisión al igual que *Random Forest*, pero se diferencia de este en el procedimiento de construcción de los árboles y en la metodología de interpretar los resultados.

Para la construcción de árboles, el concepto de *Gradient Boosting* consiste en, a partir de las debilidades de un árbol, crear árboles que sean capaces de lidiar con esas debilidades. En primer lugar, se construye un árbol y se evalúa mediante la función conocida como *loss function*. Existen múltiples maneras de calcular la *loss function* que, en función de la escogida, se procederá a aplicar el algoritmo de forma diferente. Básicamente, la *loss function* nos ofrece un parámetro que nos permite conocer cuan bien funciona el árbol. Cuanto menor sea el parámetro, más preciso será el modelo. A continuación, se debe construir un árbol que junto con su anterior sea capaz de disminuir la *loss function* que se tenía anteriormente. Matemáticamente, la forma de calcular este nuevo árbol viene dado por el gradiente de la *loss function* con respecto a la salida del modelo anterior [58].

A la hora de la interpretación de las salidas de cada árbol, cada predicción será evaluada y ponderada y a continuación se procede a la decisión final del modelo.

Con respecto a *Random Forest*, *Gradient Boosting* ofrece modelos más precisos y que son altamente personalizables, pero *Random Forest* se ajusta mejor ante los errores. Además, *Gradient Boosting* tiende a sobreajustarse rápidamente, por lo que hay que prestar especial atención a la hora de aplicar este tipo de algoritmos.

2.3.4 Regresión lineal

La regresión lineal es una técnica de aprendizaje supervisado que se basa en un método de regresión estadístico en el que existe una relación entre dos variables, una dependiente (eje Y) y una independiente (eje X). El valor de entrada al algoritmo será el independiente y el de salida el dependiente. En caso de haber un único valor de entrada, estaremos ante regresión lineal simple. Si, por el contrario, hay más de uno, se denominará regresión lineal múltiple [59].

El algoritmo trata de establecer el valor de salida (Y) de los datos de entrada (X) a partir de la ecuación de una recta que represente al conjunto de los datos. En la Figura 2-19 podemos ver la representación de los datos en puntos rojos y la recta deseada en verde.

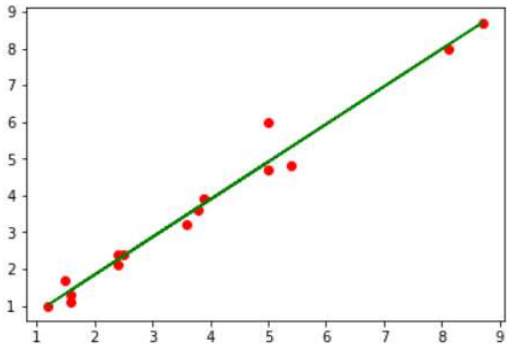

Figura 2-19 Recta de regresión lineal [54]

Para hallar la ecuación de la recta (Ecuación 2-1) debemos hallar los valores *w* y *b*. Para hallar estos valores habrá que encontrar la recta que tenga el menor error cuadrático medio (ECM) que viene dado por la Ecuación 2-2. Igualando esta ecuación a la ecuación de la recta obtendremos la Ecuación 2-3 que,

mediante iteraciones, nos permitirá obtener el valor de *w* y *b* que hacen que el error cuadrático medio tenga el menor valor posible, obteniéndose así la ecuación de la recta que buscamos.

$$y = wx + b$$

Ecuación 2-1 Ecuación de la recta

$$ECM = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Ecuación 2-2 Error cuadrático medio

$$ECM = \frac{1}{N}\sum_{i=1}^{N}(y_i - \omega x_i - b)^2$$

Ecuación 2-3 Resultado de igualar ecuación de la recta y ECM

2.3.5 Regresión polinomial

Es un caso específico dentro de la regresión lineal. Se utilizan cuando una recta no es capaz de ajustarse fielmente a los datos. En este caso, en lugar de utilizar la ecuación de la recta, se utilizará el polinomio del grado que se desee. A mayor grado del polinomio, más se ajustará la recta a los datos, pero será más fácil sufrir sobreajuste [29].

2.3.6 Regresión logística

La regresión logística es una técnica estadística que se utiliza en el aprendizaje automático supervisado para la resolución de problemas de clasificación binaria. También es capaz de predecir la probabilidad de que ocurra un nuevo evento.

El algoritmo se entrena a partir de los datos etiquetados. Dado que este componente está pensado para problemas de dos clases, la columna de etiqueta o clase debe contener exactamente dos valores [60]. El proceso de entrenamiento de una función logística se puede realizar maximizando la probabilidad de que los puntos de un conjunto de datos se clasifiquen correctamente, lo que se conoce como estimación de máxima verosimilitud. La estimación de máxima verisimilitud es un enfoque genérico para la estimación de parámetros en modelos estadísticos. La maximización se puede realizar utilizando diferentes métodos de optimización como el descenso del gradiente [61].

La función que relaciona la variable dependiente con la independiente se denomina función logística (Ecuación 2-4) y su representación gráfica se muestra en la Figura 2-20. La función logística está acotada entre 0 y 1. De esta forma, al introducir el valor de X en la ecuación resultante, obtendremos si pertenece a una clase o a otra dependiendo si el valor de p es mayor o menor de 0,5. Además el valor p también se puede interpretar como la probabilidad de un suceso.

$$p = \frac{1}{1 + e^{-x}}$$

Ecuación 2-4 Función logística

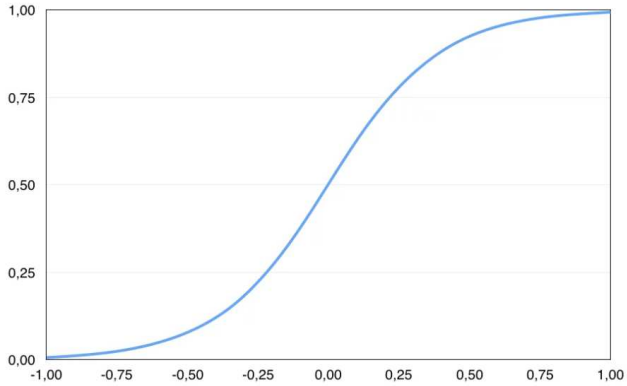

Figura 2-20 Representación de la función logística [61]

A diferencia de la regresión lineal que proporciona una salida continua, la regresión logística proporciona una salida discreta. Pese a ser una técnica relativamente sencilla, tiene un gran uso en la industria o en aplicaciones como la detección de *spam* del correo electrónico.

2.3.7 Métodos Bayesianos

Los Métodos Bayesianos o algoritmos de Naïve Bayes son algoritmos de aprendizaje supervisado que se basan en la técnica de clasificación estadística del teorema de Bayes.

A este tipo de modelos se les conoce como "inocentes" porque parten de la premisa de que las características diferenciadoras de los datos son independientes entre sí. Por ejemplo, si quisiera clasificar si un café es de buena calidad, este método supondría que la cantidad de leche que tenga el café no afecta a la cantidad de azúcar que deba llevar el café para que este sea de buena calidad.

Esta técnica crea un clasificador probabilístico que determina a qué clase pertenece un dato en función de la probabilidad de eventos previos, en función de la probabilidad condicional, utilizando la ley de Bayes (Ecuación 2-5), donde $P(A)$ y $P(R)$ son la probabilidad de que ocurran A y R respectivamente, $P(R|A)$ es la probabilidad de que se dé R dado A y $P(A|R)$ es la probabilidad de que se dé A dado R, que se conoce como la probabilidad posterior. Para su aplicación, organiza los datos en una tabla de frecuencias y calcula la probabilidad de los diferentes eventos. A continuación, aplica la ecuación Naïve Bayes (Ecuación 2-5) para calcular la probabilidad posterior de cada clase y la que tenga el mayor valor será la predicción.

$$P(A|R) = \frac{P(R|A)P(A)}{P(R)}$$

Ecuación 2-5 Ley de Bayes

El algoritmo de Naïve Bayes es muy usado para problemas de clasificación, aportándonos ventajas como su simplicidad y rapidez a la hora de ejecutarlo lo que nos permite realizar clasificaciones tanto binarias como multiclase. Por otro lado, tiene el inconveniente de que al tratarse de modelos inocentes no son capaces de asimilar la relación entre características, obteniendo una visión distorsionada de la realidad en la que sí existe dicha relación [62].

2.3.8 Support Vector Machine (SVM*)*

Support Vector Machine, o Máquina de Vector Soporte en castellano, es un algoritmo de aprendizaje supervisado que sirve tanto para clasificación como para regresión. Sin embargo, generalmente solo se utiliza para resolver problemas de clasificación. Son algoritmos sencillos que ofrecen soluciones precisas con relativamente poca carga computacional.

Para aplicar SVM, primero debemos representar los datos en un espacio n-dimensional, donde *n* será el número de características de los datos, tal y como se ve en la representación de la izquierda de la

Figura 2-21. No hay que olvidar que los datos están etiquetados por lo que sabemos a qué clase pertenecen. Una vez realizada la representación, lo que busca el algoritmo es hallar un hiperplano que divida los datos por clases.

Como podemos observar en la representación del centro de la Figura 2-21, existe más de un hiperplano que divide los datos por clases. Para entender cuál es el hiperplano que se escogerá, primero debemos explicar el concepto de margen y de vector soporte. Se puede apreciar un ejemplo de los mismos en la representación de la derecha de la Figura 2-21. Se define como vector soporte de una clase como el dato más cercano al hiperplano de la clase. Habrá tantos vectores soportes como clases existan, un vector por clase. El margen es la suma de cada una de las distancias entre cada vector soporte y el hiperplano. De esta forma, el hiperplano que utilizará el algoritmo será aquel que maximice el margen tal y como se ve en la representación de la derecha de la Figura 2-21.

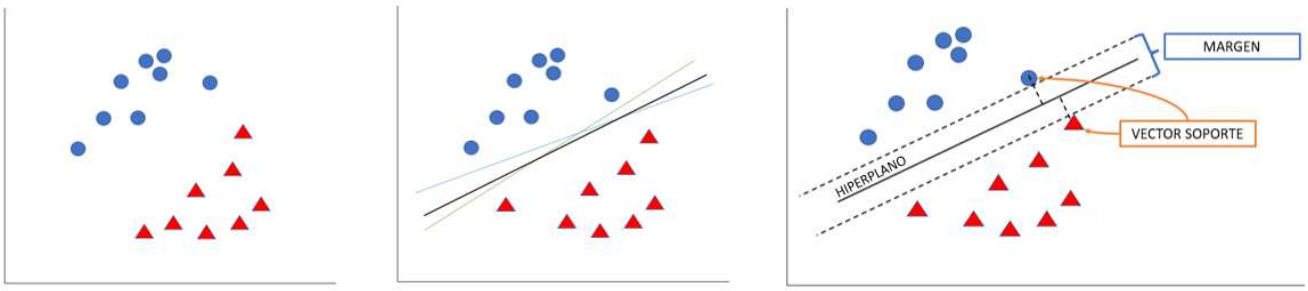

Figura 2-21 Fases de entrenamiento de un algoritmo SVM (fuente: propia)

Una vez establecido el hiperplano, el algoritmo clasificará los nuevos datos de entrada en función de la zona delimitada por el hiperplano en la que quede representado este nuevo punto, obteniendo la misma clase que los datos de la región [63].

Existen situaciones en las que definir una recta que separe los datos por clases es complicado. Para solucionar este problema se utiliza el truco de Kernel. El truco de Kernel consiste en dotar a los datos de una nueva dimensión, de tal forma que con la nueva disposición de los datos nos resulte más fácil separar los datos por clases. Se muestra un ejemplo en la Figura 2-22. Una vez hallado el hiperplano, se representarán de nuevo los datos y el hiperplano con las dimensiones originales.

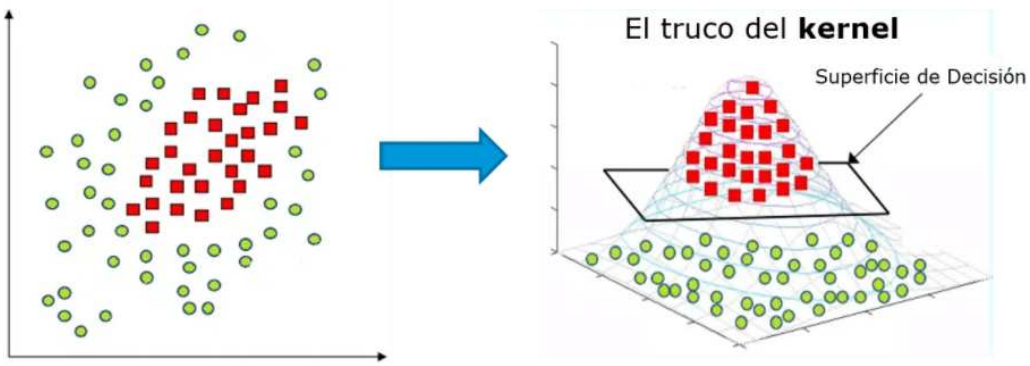

Figura 2-22 Ejemplo de aplicación del truco de Kernel [64]

2.3.9 Redes neuronales

Como ya se expuso previamente, las redes neuronales son una técnica de *Machine Learning* que dan nombre a toda una rama del aprendizaje automatizado que se denomina *Deep Learning*. Puesto que estas ya se han tratado en profundidad, nos limitaremos a remarcar que se engloba dentro de las técnicas de aprendizaje supervisado y que se trata de una de las técnicas más potentes y con mayor proyección dentro del *Machine Learning*. Además, es frecuente combinar las redes neuronales con otros algoritmos con el fin de mejorar su rendimiento.

2.3.10 K-Nearest Neighbor (KNN)

K-Nearest Neighbor, o K-Vecinos Más Próximos en castellano, es un algoritmo de aprendizaje supervisado muy sencillo y que se utiliza, en mayor medida, para clasificar datos en base a la clase de los datos similares.

El funcionamiento de este algoritmo consiste en medir la distancia entre los nuevos datos de entrada a clasificar y los datos de entrenamiento que son datos ya etiquetados. Los datos se ordenan de menor a mayor en función de la distancia al dato que se quiere clasificar y el algoritmo seleccionará los *k* valores de menor distancia para realizar la clasificación. El algoritmo observa las clases de los vecinos que ha seleccionado y clasificará el nuevo dato de entrada dentro de la clase a la que pertenezcan la mayoría de los vecinos seleccionados.

En la Figura 2-23 podemos ver un ejemplo de cómo funcionaria el algoritmo con un valor de entrada representado por una estrella, dos posibles clasificaciones diferentes (azul o rojo) y un valor de *K*=3. Los datos de entrenamiento están representados con círculos y su color determina la etiqueta. Como se puede observar, en este caso los tres datos vecinos más cercanos se corresponden con los de dentro del círculo. El dato se clasificará con la etiqueta "azul" puesto que la mayoría de los datos del círculo son azules.

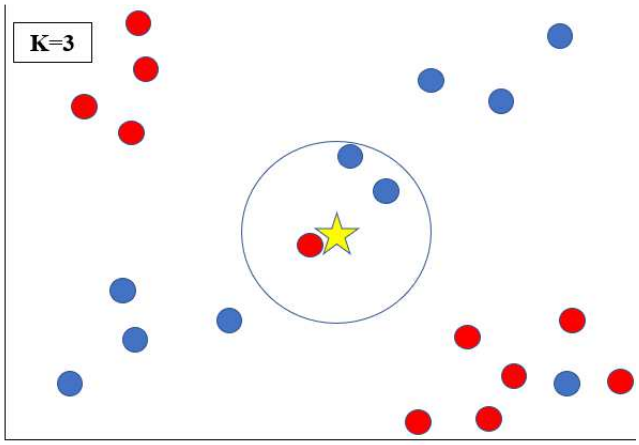

Figura 2-23 Ejemplo de KNN para K=3 (fuente: propia)

A diferencia de los algoritmos que hemos visto hasta el momento, KNN es un algoritmo basado en instancias, es decir, no utiliza los datos etiquetados para entrenar un modelo, sino que hace uso de los datos de entrenamiento en el mismo momento que entran los datos a clasificar. Este tipo de técnicas de aprendizaje también se conoce como *lazy learning methods* [65].

Como se puede apreciar, a la hora de ejecutar este algoritmo, será necesario realizar el cálculo de dos parámetros para su correcto funcionamiento: la distancia de los datos etiquetados a cada nuevo dato a clasificar y el valor de *K*.

Existen muchas técnicas para medir la distancia entre los datos. Sin embargo, la más utilizada es la distancia euclídea, que es la que se obtiene a partir de las coordenadas de los datos (vease Ecuación 2-6), en la que P1 y P2 son los puntos de los que se quiere saber la separación siendo sus coordenadas (x_1, y_1) y (x_2, y_2) respectivamente.

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 2-6 Distancia euclídea

Determinar el valor de *K* es una tarea más compleja. Un valor de menor de *K* realizará clasificaciones menos estables pues se basará en menos datos. Por otro lado, un valor de *K* mayor realizará clasificaciones más estables pues dispone de más datos y, como consecuencia, es más probable que la

clasificación sea más precisa. Por ejemplo, en la Figura 2-23 si hubiésemos utilizado *K*=1, la etiqueta del nuevo dato habría sido diferente que para *K*=3.

Aunque parezca que cuanto mayor sea el valor de *K* mejor será la predicción, no sucede así. Cuando *K* obtiene un valor demasiado elevado, llega un momento en el que se empiezan a apreciar numerosos errores, pues está trabajando con datos que ya están demasiado lejos. Existen muchas técnicas para determinar el valor de *K*, pero una de las más utilizadas se basa en ensayos de prueba y error en los que se va aumentando el valor de *K*. En el momento en que el error aumenta en lugar de disminuir, fijamos *K* en el valor anterior a que el error aumentara. Además, es recomendable usar un valor de *K* impar para que no se produzca un empate entre las clases de los datos vecinos [66].

2.4 Algoritmos de aprendizaje no supervisado

En este apartado, se procede a exponer de manera resumida las principales técnicas de aprendizaje no supervisado. Cabe recordar que este tipo de técnicas se caracterizan por utilizar datos no etiquetados para el aprendizaje del modelo.

2.4.1 K-Medias

El algoritmo *K-Means*, o K-medias en castellano, es uno de los algoritmos de aprendizaje no supervisado destinado a la clusterización que permite agrupar un conjunto de datos de forma simple en *k* grupos.

La metodología de este algoritmo es muy simple y se puede resumir en cinco pasos, que se pueden ver además en la Figura 2-24:

1. Selección del número de *clusters* (*k*): el valor de *k* óptimo se puede establecer de diversas maneras, la más famosa y utilizada es el *elbow method* que consiste en la representación gráfica de la varianza para cada *k*, y seleccionar el valor de *k* en el que se aprecie visualmente un cambio significativo, es decir, un codo en la gráfica.
2. Selección de los centroides iniciales: los centroides son los puntos que determinan donde está el centro de cada *cluster*, desde los cuales mediremos las distancias al resto de los datos. Los centroides iniciales, en la mayoría de los casos, son escogidos aleatoriamente.
3. Asignar los *clusters*: se recorren todos los datos y se calcula la distancia de los mismos a cada centroide. Los datos serán asignados al *cluster* del centroide que le es más cercano. La distancia utilizada suele ser la distancia euclídea que fue explicada previamente. Una vez realizado este paso, existen funciones que nos permiten evaluar la calidad de los *clusters* para determinar si es necesario realizar más iteraciones.
4. Recalcular los centroides: en caso de requerirse más iteraciones, se calculan los nuevos centroides. Existen muchas opciones para hacerlo, siendo la más común situarlos en el centro de los datos de cada *cluster*.
5. Repetición de los pasos 3 y 4: Una vez reasignados los centroides, se repiten estos pasos hasta que se considere que los resultados obtenidos son los deseados.

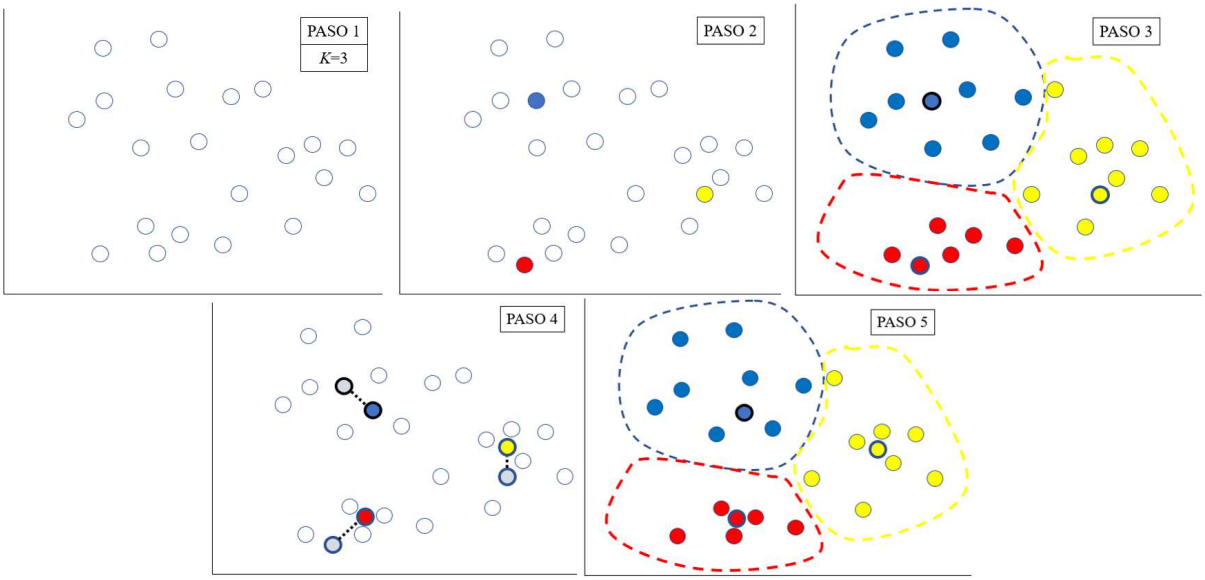

Figura 2-24 Ejemplo esquemático paso a paso de ejecución de K-medias (fuente: propia)

Este algoritmo presenta problemas cuando los datos son muy cercanos a más de un centroide y para grupos de diferentes formas y tamaños, por lo que se deberá realizar un estudio previo de los datos para determinar si se trata del algoritmo adecuado para realizar cada tarea [67].

2.4.2 Clusterización jerárquica

La clusterización jerárquica es un algoritmo de aprendizaje no supervisado que trata de realizar el agrupamiento de los datos en base a la distancia entre ellos, tratando de crear *clusters* formados por datos similares entre si.

El funcionamiento del algoritmo es muy sencillo y fácil de comprender: en primer lugar, calcula la distancia entre todos los datos. Una vez calculadas las distancias, agrupará en un mismo *cluster* los datos más cercanos. A continuación, hará lo mismo con el siguiente más cercano. Este proceso se repite hasta que se alcance el número de *clusters* deseado o hasta que la siguiente distancia entre puntos a agrupar sea mayor que un valor prefijado por el usuario.

Una herramienta muy útil para el desarrollo de este algoritmo es el dendograma [68]. Este consiste en una representación gráfica de la ejecución del algoritmo en la que en el eje X estarán los datos a agrupar y en eje Y la distancia. La forma de proceder es la misma que se ha explicado anteriormente. Los datos se van uniendo en el eje X y la línea que los une tendrá una altura igual que la que los separará. Cuando ya hemos realizado el proceso obtendremos el dendograma. Se puede ver un ejemplo en la Figura 2-25, donde a la derecha se observa el dendograma y a la izquierda los datos de donde proviene. Es una herramienta muy útil pues si queremos dejar de agrupar por distancia máxima trazaremos una horizontal, como por ejemplo la línea descontinua amarilla de la Figura 2-25, en la distancia que se quiera establecer y se obtendrán los grupos. Para el número de *clusters* se puede usar el dendograma o parar de agrupar en cuanto se tengan los grupos deseados.

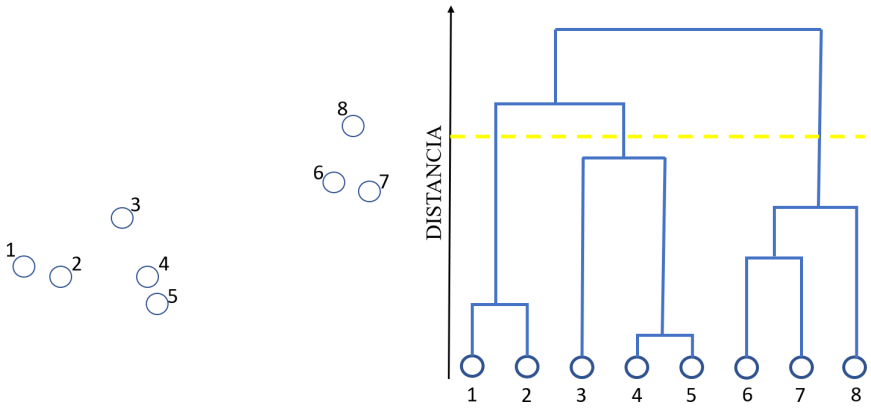

Figura 2-25 Ejemplo de dendograma (fuente: propia)

2.4.3 DBSCAN

DBSCAN, *Density-Based Spatial Clustering of Applications with Noise*, es un algoritmo de clusterización dentro de la familia de los algoritmos de aprendizaje no supervisado que funciona en base a la densidad del conjunto de datos. En este algoritmo, los datos serán representados por puntos y sus coordenadas vendrán dadas por características concretas de los datos [67].

Para el funcionamiento de este algoritmo el usuario debe establecer primero el valor de dos hiperparámetros:

- Epsilon (ϵ): distancia a partir de la cual se determina si un punto es cercano a otro. Si la distancia entre dos puntos es menor de ϵ, serán considerados vecinos.
- Puntos mínimos: son los vecinos mínimos que debe tener un punto para ser considerado punto central.

En DBSCAN los datos se dividen en tres tipos: central, borde y ruido. Un ejemplo de estos para unos hiperparámetros dados es el que se puede ver en la Figura 2-26. Debemos conocer previamente en qué consisten estos puntos para poder entender el funcionamiento del algoritmo:

- Puntos centrales: son los puntos que tienen una cantidad de vecinos que supera el número de puntos mínimos preestablecido.
- Puntos de borde: son los puntos que tienen una cantidad de vecinos menor que el valor de puntos mínimos preestablecido.
- Puntos de ruido: son los puntos que no se engloban en ninguna de las dos categorías anteriores.

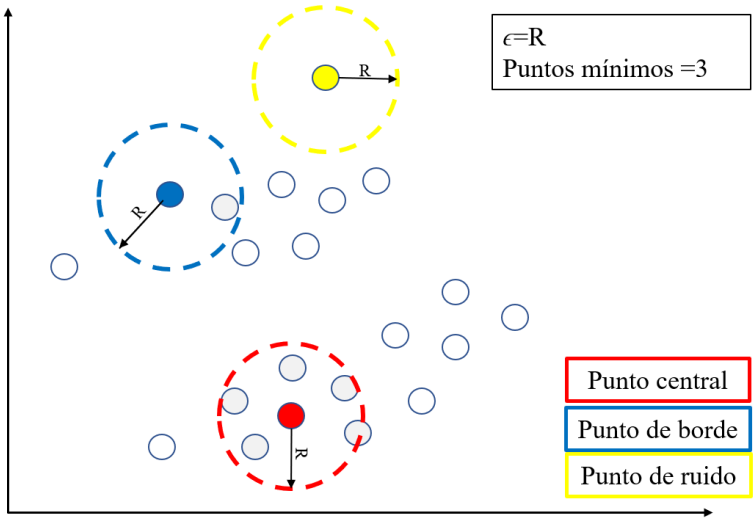

Figura 2-26 Clasificación de puntos en algoritmo DBSCAN (fuente: propia)

Lo primero que realiza el algoritmo es identificar todos los puntos clasificándolos en los tres tipos que se han mencionado anteriormente. A continuación, el algoritmo toma un punto al azar y, en caso de ser un punto central, establecerá una nueva clase. El algoritmo recorre todos sus vecinos y les asigna su misma clase. Los vecinos de este punto realizarán lo mismo, siempre y cuando sean puntos centrales, los vecinos de los puntos de borde no tienen por qué pertenecer a su misma clase, es decir, para pertenecer a una clase los puntos tienen que ser vecinos de un punto central de esa misma clase. Una vez realizado este proceso, el algoritmo escogerá al azar otro punto y así sucesivamente hasta clasificar todos los datos posibles, pues habrá datos que no cumplan los criterios para pertenecer a un *cluster* (puntos de ruido o puntos de borde sin puntos centrales vecinos) [69].

2.4.4 PCA

Principal Component Analysis (PCA) [70] es un algoritmo de reducción de la dimensionalidad que se encuadra dentro de las técnicas de aprendizaje no supervisado. Este algoritmo se utiliza para conjuntos de datos que poseen un elevado número de variables, hasta tal punto que manejar todas estas variables hace que el empleo de los datos sea una tarea muy compleja. El propósito de PCA es reducir el número de estas variables del conjunto de datos, pero manteniendo la máxima información posible. Se trata de buscar componentes principales que sean capaces de representar los datos de la forma más fehacientemente posible. Los componentes principales son resultado de la combinación de las variables iniciales.

El funcionamiento de este algoritmo, sin entrar en profundidad en detalles matemáticos, se puede resumir en cinco pasos:

1. Estandarización: se trata de transformar los datos a escalas que sean comparables ente sí.
2. Cálculo de la matriz de covarianza: busca determinar si existe alguna relación entre las variables de los datos de entrada. Para ello construye la matriz covarianza, que no es más que una matriz simétrica que tiene como datos de entrada las covarianzas asociadas con todos los pares posibles de las variables iniciales.
3. Cálculo de los autovectores y autovalores de la matriz de covarianza para encontrar los componentes principales: los autovectores de la matriz covarianza son en realidad las direcciones de los ejes donde hay más varianza y, por lo tanto, más información. A estos autovectores los llamaremos componentes principales.
4. Vector de características: se trata de seleccionar las componentes principales a mantener y a descartar. El vector de características es simplemente una matriz cuyas columnas son los autovectores que decidimos mantener.

5. Reformular los datos a lo largo de los ejes de los componentes principales: consiste en reorientar los datos de sus ejes originales a los representados por las componentes principales mediante el uso del vector de características.

2.4.5 ICA

Independent Component Analysis (ICA), o análisis de componentes independientes en castellano, es una técnica de *Machine Learning* para resolver problemas de reducción de la dimensionalidad que se encuadra dentro de los algoritmos de aprendizaje no supervisado. A diferencia de PCA, que trata de buscar correlación entre las variables, ICA trata de buscar independencia, siendo capaz de detectar subcomponentes independientes en un conjunto de datos y de una manera mucho más precisa.

Un clásico ejemplo para la comprensión del funcionamiento de este algoritmo es "el problema del cóctel". Imagínese dos personas en un cóctel hablando entre ellas incapaz de escucharse la una a la otra debido al ruido en la sala. Este algoritmo participaría en analizar todas estas entradas de ruido junto con la voz de los individuos. El algoritmo estudiaría todas las entradas y trataría de diferenciar la voz de los individuos del ruido ambiente [71].

Básicamente la metodología utilizada por ICA se basa en técnicas estadísticas simples que tratan de estudiar la independencia de las variables de entrada en base a dos suposiciones clave: la independencia estadística entre ellas y que posean una naturaleza no gaussiana, es decir, que posean una distribución anormal.

2.5 Conocimiento del entorno marítimo

El Conocimiento del Entorno Marítimo (CEM) se define como la acción de fusionar y analizar la información que se recibe de un número elevado de fuentes, obteniéndose una imagen precisa de todo lo que sucede en los espacios marítimos de interés nacional. Como resultado de este proceso continuo de sintetizado y análisis de la información, se obtiene una representación precisa, casi en tiempo real, de las actividades que transcurren en nuestras aguas. Esta representación se denomina *Recognized Maritime Picture* (RMP) [72], apreciable un ejemplo en la Figura 2-27.

Figura 2-27 Ejemplo no clasificado de RMP [73]

A diferencia de tiempos pasados, el entorno marítimo no se limita al mar territorial, zona contigua y Zona Económica Exclusiva (ZEE), sino que ha evolucionado a un concepto mucho más amplio. El

espacio marítimo de interés se ha ido incrementando hasta alcanzar una definición que se centra en dos factores: uno espacial, allí donde se encuentren los intereses, y otro temporal, en cada momento. Por lo tanto, el espacio marítimo de interés nacional actualmente es un concepto cambiante y muy amplio [74].

Mas de 200.000 buques navegan al año por las líneas de comunicaciones marítimas de soberanía española. La protección de estas líneas de tráfico marítimo es fundamental, no solo para la economía nacional, también para la internacional. El mar es un medio utilizado por diferentes organizaciones para llevar a cabo todo tipo de actividades fraudulentas, entre las que se pueden destacar acciones terroristas, tráfico de armas de destrucción masiva, expolio del patrimonio subacuático nacional y la inmigración ilegal que sigue siendo un problema latente en Europa. La Armada Española tiene la misión de proteger y vigilar estos espacios de interés nacional.

Bajo esta premisa, surge la necesidad de ejercer vigilancia y control sobre estos espacios, y el conocimiento del entorno marítimo juega un papel fundamental a la hora de realizar esta tarea. Un buen conocimiento del entorno marítimo se ha convertido a día de hoy en una herramienta indispensable para la planificación y conducción de las acciones de seguridad marítima llevadas a cabo por las Fuerzas Armadas a través de la Fuerza de Acción Marítima (FAM).

La actual capacidad de obtención del Conocimiento del Entorno Marítimo descansa en el Centro de Operaciones y Vigilancia de la Acción Marítima (COVAM). El COVAM es la herramienta de la FAM para recoger y analizar la información procedente de las distintas fuentes para conseguir un Conocimiento del Entorno Marítimo lo más completo posible.

2.5.1 COVAM

El Almirante de Acción Marítima (ALMART) tiene como uno de sus principales cometidos ser Comandante del Mando Operativo Marítimo (CMOM), integrándose en la estructura operativa de las Fuerzas Armadas. La responsabilidad de este organismo es el planeamiento, conducción y seguimiento de las operaciones de vigilancia y seguridad de los espacios marítimos de responsabilidad y soberanía e interés nacional. Para poder ejercer esta misión, el CMOM se coordina con diferentes autoridades y organismos tanto civiles como militares y de ámbito nacional e internacional.

El ALMART es también el responsable del mantenimiento del Conocimiento del Entorno Marítimo en los espacios de interés nacional, y recaerá sobre el mismo la responsabilidad de activar, coordinar y generar, a nivel nacional, el Sistema de Cooperación y Orientación al Tráfico Marítimo (NCAGS).

Para auxiliar al ALMART y al CMOM en el ejercicio de sus cometidos, se cuenta con el COVAM, ubicado en Cartagena y encuadrado en el Estado Mayor de la Fuerza de Acción Marítima. Este Estado Mayor dispone de los sistemas de Mando y Control y enlace para poder efectuar la conducción y seguimiento de las operaciones, el mantenimiento adecuado del Conocimiento del Entorno Marítimo y coordinarse con las diferentes organizaciones militares y civiles, nacionales e internacionales.

El COVAM, junto con las unidades de la Fuerza de Acción Marítima (FAM), vela por la seguridad de las aguas de interés nacional como son las propias aguas nacionales, Golfo de Guinea, Golfo Pérsico y Océano Índico. En la Figura 2-28 se pueden ver estas zonas de interés. Para el desarrollo de estas misiones, el COVAM tiene como objetivo proporcionar la RMP a todas las unidades de la Armada, estaciones en tierra y agencias civiles que así lo soliciten. Además, el centro de operaciones del COVAM opera permanentemente (24/7) para cumplir muchos otros cometidos entre los que destacan:

- Llevar a cabo el seguimiento de las misiones ordenadas por ALMART/CMOM y las que estén realizando unidades de la FAM, así como ejercer la conducción operativa de las mismas cuando le sean asignadas y tomar las medidas necesarias ante incidencias o asuntos de carácter operativo que requieran de acción inmediata.
- Conseguir el conocimiento de la situación táctica del entorno marítimo (*Maritime Situational Awareness* - MSA) a partir de la construcción del RMP a través de los diferentes sistemas

de control de tráfico de los que dispone el COVAM, tanto en zonas de interés nacional como en zonas de operaciones.

- Establecer relaciones, aportando información, con otras autoridades y unidades de la Armada, así como su equivalente en armadas de otros países aliados y organizaciones y autoridades de carácter civil, sobre todo en este último caso sirviendo como punto de contacto entre la Armada y la comunidad marítima.
- Ejercer la acción de mando del ALMART/CMOM permanentemente y apoyarlo en cualquier actividad que así lo requiera.

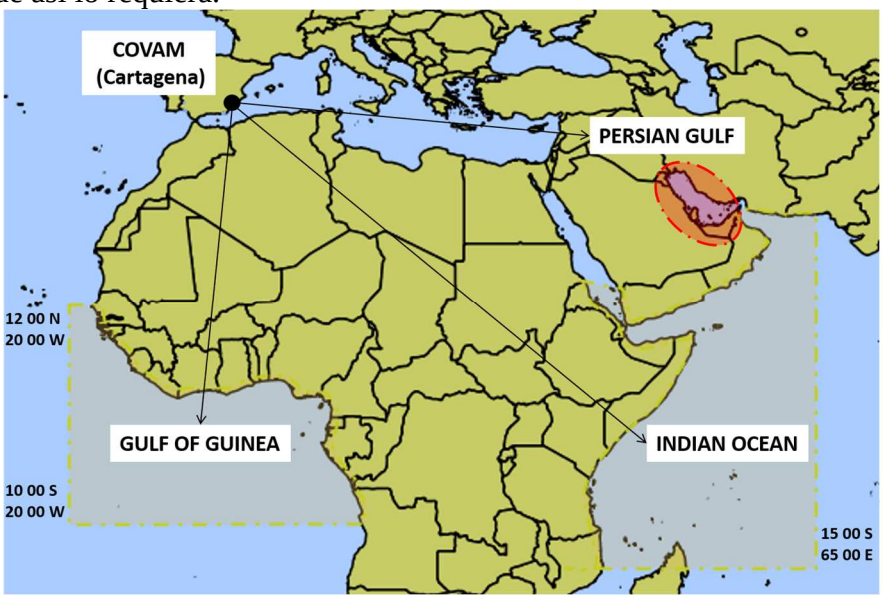

Figura 2-28 Zonas de interés del COVAM [72]

Para poder ejercer todos sus cometidos, el COVAM cuenta con un total de 35 personas que le permiten cubrir los puestos necesarios de tal forma que se pueda garantizar el funcionamiento de este de forma permanente.

Además de esta plantilla, el COVAM cuenta con personal que se encarga de realizar el planeamiento y preparación de actividades a la que se van a destacar las unidades, análisis del CEM y mantenimiento de bases de datos y sistemas del COVAM. Esta base de datos se nutre de diferentes fuentes de información, pero una de las más importantes son los datos AIS.

2.5.2 Datos AIS

El *Automatic Identification System* (AIS), Sistema de Identificación Automática en castellano, es un sistema que se desarrolló en sus orígenes como una ayuda para evitar colisiones en la mar, proporcionando a las embarcaciones que se vieran entre sí y aportando información sobre el entorno que les rodea. El AIS transmite mediante un transpondedor información a todas las embarcaciones que dispongan de este sistema y siempre que se encuentren dentro del alcance del equipo. También ofrece a las autoridades costeras una gran ayuda para el control del tráfico marítimo y tareas de mantenimiento de la seguridad en la mar [75].

Esta tecnología pasó a ser obligatoria para todos los buques de pasajeros y buques de más de 300 toneladas brutas en el año 2002 a raíz de las exigencias de la Organización Marítima Internacional con el convenio internacional para la seguridad de la vida humana en la mar (SOLAS) [4].

Existen dos tipos de transpondedores AIS [76]:

- AIS Clase A: Este tipo de transpondedor AIS solo es obligatorio en las embarcaciones de más de 300 toneladas brutas en tránsitos intercontinentales. La información transmitida y

recibida por este tipo de sistema es más completa en comparación con el transpondedor AIS Clase B, aunque son más caros y pesados.

- AIS Clase B: diseñado para embarcaciones no englobadas en el convenio SOLAS, principalmente embarcaciones de recreo. Es un sistema que aporta menos información, pero es más ligero y económico.

Además, existe la opción de que el equipo disponga de un único receptor AIS, es decir, el sistema recibe información del resto de transpondedores, pero no transmite la información propia.

El transpondedor AIS proporciona a los buques información de interés para mantener la seguridad en la navegación. Los sistemas AIS envían información codificada mediante texto plano que necesita ser decodificada. Estas tramas se decodifican, se procesan y se incorpora nueva información de otras fuentes para poder ser procesada por las consolas de presentación de datos. Los datos transmitidos se pueden organizar en cuatro grupos: datos estáticos, datos dinámicos, datos relacionados con el viaje y mensajes breves relacionados con la seguridad. Los mensajes transmitidos son los siguientes [77]:

- Información estática (transmitida cada 6 minutos):
  - Número MMSI: El número de identificación del servicio móvil marítimo es una serie de 9 dígitos que identifica inequívocamente a cada estación transmisora.
  - Número IMO: Numero de 7 dígitos asignados para cada buque de forma única, con el propósito de identificarlos y auxiliar en la consecución de la seguridad marítima.
  - Nombre y distintivo de llamada del buque.
  - Eslora y manga del buque.
  - Tipo de buque: El tipo de buque clasifica a cada embarcación dentro de unas clases determinadas que se pueden ver en el Anexo II: Códigos numéricos identificadores del tipo de buque en los datos . También se relaciona con cada tipo un número de dos dígitos que identifica este valor. Este valor tiene especial importancia dentro de las Fuerzas Armadas pues en función del tipo de buque se establecen las acciones a tomar por las unidades de la Armada para la conducción de operaciones de seguridad marítima.
  - Ubicación de la antena de fijación de posición.
- Información dinámica (depende del rumbo y velocidad):
  - Posición GPS del barco con indicación de precisión.
  - Marca de tiempo de la posición: en hora UTC.
  - Rumbo sobre fondo: rumbo efectivo del buque.
  - Velocidad sobre fondo: velocidad efectiva del buque.
- Información relacionada con el viaje (transmitida cada 6 minutos o cuando se realizan cambios):
  - Calado.
  - Cargamento.
  - Plan de ruta (*Waypoint*): estableciendo último puerto de salida y siguiente puerto de recalada, así como la fecha y hora en la que se realizó la salida del último puerto y en la que recalará en el siguiente.
- Mensajes breves relacionados con la seguridad: Texto de formato libre que tiene por objetivo transmitir un mensaje breve hacia las unidades y estaciones cercanas para contribuir a la seguridad de los buques en la mar.

2.5.3 Importancia de la IA para el CEM en la Armada

La Armada Española está viviendo un cambio a pasos agigantados hacia el fenómeno conocido como Armada 4.0. Este concepto surge por la necesidad de actualizarse y mantenerse al nivel tecnológico del resto de marinas del mundo. Esta revolución tecnológica pasa sobre todo por la

integración de conceptos como, por ejemplo, *Internet of Things* (IoT), técnicas de inteligencia artificial y *Machine Learning* en el trabajo diario de las unidades.

Las técnicas de inteligencia artificial ya se han empezado a integrar de forma experimental en buques de la armada, sobre todo para mantenimientos y facilitar el manejo de la gran cantidad de datos de los que disponen. Un claro ejemplo lo podemos ver en [2] con el proyecto *soprene* desarrollado por Indra, que consiste en el uso de inteligencia artificial para la detección de comportamientos anómalos y averías en los sistemas de los buques.

El COVAM cuenta con infraestructuras y equipos, que se pueden ver en la Figura 2-29, que le sirven de apoyo para realizar las misiones que le son encomendadas. Por ello, la adaptación a las nuevas tecnologías era inherente al nacimiento de la Armada 4.0. Por este motivo, la Armada ya confirmó el desarrollo de un programa para la modernización del COVAM en el que se han invertido 1,7 millones de euros, con fin de mediados de 2022, este programa será gestionado por la Dirección General de Armamento y Material (DGAM). El objetivo de este programa es incorporar mejoras técnicas, de formación y de infraestructura del centro, así como en la estandarización y modernización [78].

Figura 2-29 Centro de Operaciones y Vigilancia de la Acción Marítima (COVAM) [78]

Además de la modernización de la infraestructura y los equipos, el Almirante Jefe de Estado Mayor de la Defensa (AJEMA) remarca la importancia de la introducción de la inteligencia artificial en el CEM diciendo que esto supondrá **"**un paso adelante a la hora de interconectar un gran número de agencias, tanto nacionales como internacionales, que tienen información del tráfico marítimo. Facilitará el conocimiento del entorno marítimo ya que, a la hora de analizar datos, cuanto más ingente es la cantidad, menos puede el ser humano ser capaz de priorizar y tomar decisiones, por lo que la introducción de la inteligencia artificial va a ser un paso adelante para tomar decisiones adecuadas que ayudarán al ALMART" [79].

El interés de la Armada en la aplicación de esta nueva tecnología se puede ver por ejemplo en el proyecto que el COVAM encargado al Centro Universitario de la Defensa para desarrollar un demostrador tecnológico que sea capaz de detectar anomalías en los patrones de comportamiento de los buques en la mar mediante el uso de técnicas de inteligencia artificial, auxiliando al COVAM en su misión de vigilancia marítima de las zonas marítimas de interés. Este proyecto es en el que se encuadra el presente Trabajo Fin Grado.

2.6 Trabajos previos en el campo de estudio

A continuación, se muestra una breve reseña de algunos ejemplos de avances que se han realizado previamente en este campo de estudio y que han servido para enfocar ciertos aspectos de este trabajo.

2.6.1 Clasificación de buques basada en la agrupación del comportamiento de los buques a partir de datos AIS

Este trabajo que se puede ver en [80], es un proyecto desarrollado por miembros de la Universidad Tecnológica de Delft en los Países Bajos.

Este trabajo utiliza los datos AIS registrados de los buques que transitan por la zona portuaria de Rotterdam para entrenar un algoritmo con el fin de alcanzar dos objetivos: determinar el comportamiento de los buques en la zona portuaria para su posterior clasificación y la clasificación de estos buques en función de sus características recogidas en los datos AIS.

Para el desarrollo de los clasificadores, este proyecto utiliza los datos AIS y una combinación de técnicas de aprendizaje supervisado y no supervisado, lo cual le permite no solo clasificar los buques sino también determinar su comportamiento futuro en base a sus características. Además, como resultado del estudio optimiza el número de atributos para la clasificación utilizando únicamente datos estáticos, en concreto, la eslora y la manga de un buque.

2.6.2 Clasificación de buques basada en Random Forest utilizando información estática de los datos AIS

Este trabajo, que se puede apreciar en [81], está desarrollado por miembros de la Universidad Nacional de Tecnología para la Defensa de China.

La motivación de este trabajo es muy similar a la del presente TFG. Básicamente de lo que trata es de desarrollar una clasificación mediante técnicas de inteligencia artificial para completar los mensajes AIS en que se presenten campos vacíos, más concretamente el tipo de buque, prediciendo su valor a través de un algoritmo. Para ello la metodología empleada es *Random Forest* y los datos a partir de los cuales realiza el entrenamiento de la máquina, y que posteriormente utilizará para determinar el tipo de buque, son los datos estáticos de los mensajes AIS, consiguiéndose una metodología de trabajo bastante eficaz.

2.6.3 Caracterización del tráfico marítimo con AIS

Consiste en un proyecto desarrollado por el CMRE (*Center of Maritime Research & Experimentation)* de la OTAN que se puede encontrar en [82]. El informe se desarrolla para la caracterización del tráfico de superficie a partir de datos AIS y está enmarcado en los proyectos de vigilancia marítima de la OTAN.

El objetivo de este proyecto es el desarrollo de una máquina mediante uso de técnicas de inteligencia artificial para la detección de anomalías, enfocándose para ello en el estudio de los datos cinemáticos. Se utilizan tanto fuentes históricas como datos AIS actuales que juntos constituyen una base de datos que permite la creación de mapas de densidad a partir de parámetros de los mensajes AIS. Estos mapas se procesan mediante técnicas de *Machine Learning* dando como resultado un algoritmo que será capaz de alertar en caso de que un buque lleve a cabo comportamientos anómalos en base a los datos AIS que este transmite.

3 DESARROLLO DEL TFG

Después de haberse realizado un análisis sobre las posibles técnicas de *Machine Learning* junto con las ventajas que estas nos ofrecen y los ámbitos en los que su aplicación resulta de más utilidad, en el presente apartado se expondrá la solución propuesta utilizando el algoritmo de aprendizaje supervisado *Random Forest*, junto con la justificación del empleo de dicho algoritmo, así como el desarrollo del empleo de esta metodología.

A su vez se expondrán el entorno de trabajo seleccionado para este proyecto (*Jupyter Lab* [83]) y del lenguaje de programación empleado para el desarrollo del mismo (*Python* [84]). Estos constituirán la herramienta fundamental para realizar el análisis y tratamiento de los datos AIS, los cuales también serán analizados en el presente apartado de la memoria.

Por último, se enumerarán todos los experimentos realizados, desde los que se realizan con el objetivo de obtener una familiarización con los datos, su tratamiento previo para su empleo, la creación de atributos para que el algoritmo pueda trabajar y la metodología de *Random Forest*. Todo este proceso da como resultado tres *scripts* (*PrettyStatic.ipynb*, *PrettyDinamic.ipynb* y *PrettyMix.ipynb*) y los resultados de su ejecución serán evaluados en el siguiente capítulo.

3.1 Entorno de trabajo y software empleado

Como ya se ha explicado en una breve introducción previamente, la aplicación web seleccionada para el tratamiento de los datos y empleo de técnicas de aprendizaje automático ha sido *Jupyter Lab* y el lenguaje de programación empleado para la elaboración del código ha sido el lenguaje *Python*. A continuación, se justificará la selección de ambos y se mostrarán los primeros pasos en su empleo dentro del desarrollo del trabajo.

3.1.1 Jupyter Lab

Jupyter Lab es una aplicación web diseñada para llevar a cabo el desarrollo interactivo de cuadernos, datos y código. Gracias a la flexibilidad proporcionada por esta interfaz, los usuarios son capaces de configurar y organizar flujos de trabajo en ciencia y análisis de datos, computación científica, periodismo computacional y aprendizaje automático [85].

Jupyter Lab es el sucesor oficial de *Jupyter Notebooks*, por lo que nos ofrece aún más ventajas que su precursor permitiendo más opciones de personalización e interacción y haciendo más sencilla la implementación de extensiones. Su principal ventaja es que el código se organiza en celdas independientes, como las que se pueden ver en la Figura 3-1, haciendo más sencilla la tarea de la detección de errores y pudiendo ejecutar programas de forma parcial, es decir, es posible probar bloques de código de forma independiente. Además, gracias a que en la aplicación existen muchos núcleos, la

programación no se limita al lenguaje *Python*, aportando flexibilidad a la hora de desarrollar el código y análisis. *Jupyter Lab* soporta más de cincuenta lenguajes de programación. Aunque estas son las principales ventajas de hacer uso de la aplicación, también dispone de las siguientes virtudes [86]:

- Código abierto: su código fuente se encuentra a disposición de todos los usuarios de forma libre, lo que permite mayores contribuciones para actualizaciones y mejoras.
- *Software* gratuito.
- Funciona en el navegador: no requiere de instalación de aplicaciones en los equipos, permite trabajar de forma completamente *online* desde el navegador.
- Diferentes opciones a la hora de exportar y compartir los resultados: la aplicación ofrece a sus usuarios exportar y compartir sus códigos en diferentes formatos, como HTML, PDF, *Markdown*, *script* de *Python* o fichero *Notebook*.
- Permite colaboración (*JupyterHub*): los usuarios pueden realizar proyectos de forma conjunta gracias al empleo de *JupyterHub*, que es un servidor que permite el acceso simultáneo de varios usuarios a sus cuadernos de *Jupyter Lab*, ejecutando un servidor *Jupyter* de forma independiente para cada uno de ellos.

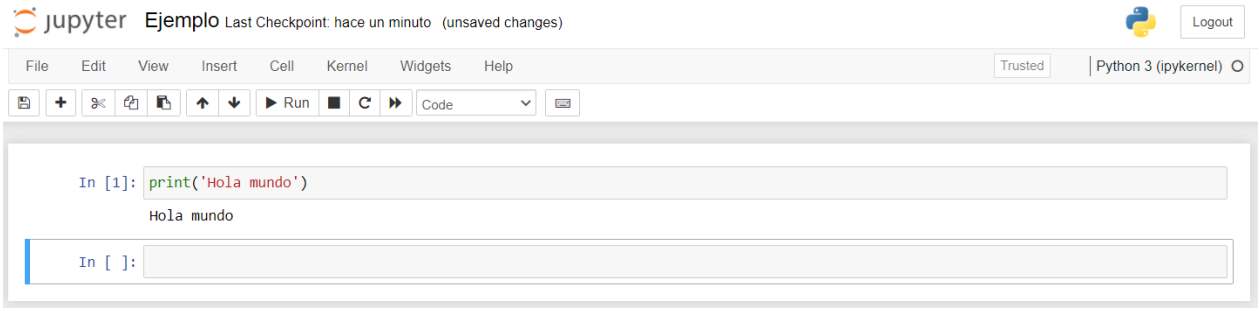

Figura 3-1 Interfaz para escritura de código de *Jupyter Lab* (fuente: propia)

Por estos motivos, *Jupyter Lab* ha sido escogido como la aplicación a utilizar para el desarrollo de este trabajo, siendo clave su capacidad de ejecución por bloques para simplificar la tarea de cambios en el código a la hora de realizar experimentos, su facilidad de obtención y ejecución, y la facilidad de importar las librerías y funciones necesarias, sobre todo a la hora de realizar tareas de aprendizaje automático.

Para el desarrollo del proyecto, más concretamente, se ha hecho uso de esta aplicación a través de un servidor virtualizado en un equipo proporcionado por el CUD-ENM.

3.1.2 Lenguaje de programación Python

Dentro de todos los posibles lenguajes de programación de los que *Jupyter* nos permite hacer uso para el desarrollo de nuestro código, la opción escogida ha sido *Python*. El lenguaje *Python* es un lenguaje de programación que actualmente es el de mayor crecimiento a nivel mundial gracias, entre otras cosas, a la extensa comunidad de la que dispone y a su facilidad a la hora de realizar la interpretación y la escritura de código debido a su alta similitud con el lenguaje humano [87].

Las principales razones para hacer uso de este lenguaje de programación son, en primer lugar, que la aplicación *Jupyter Lab*, a pesar de soportar más de cincuenta lenguajes de programación como ya se mencionó previamente, está diseñado específicamente para la programación en *Python* y, en segundo lugar, *Python* cuenta con prácticamente todos los paquetes necesarios para realizar el procesado de los datos y el desarrollo de tareas de aprendizaje automático.

3.1.3 SQLite3

SQLite [89] es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de forma rápida, sencilla, potente y eficaz en equipos con pocas capacidades de *hardware*, como por ejemplo dispositivos móviles. *SQLite* es el motor de base de datos más utilizado en el mundo.

Actualmente se puede encontrar integrado en todos los teléfonos móviles y en la mayoría de los ordenadores.

3.1.3.1 SQL

Structured Query Language (SQL) [89], o Lenguaje de Consulta Estructurado, es un lenguaje de programación que permite interactuar con bases de datos relacionales, en este caso *SQLite3*, en las que la información se organiza en tablas. Se trata de un lenguaje sencillo. Además, lleva treinta años siendo el lenguaje más utilizado a la hora de interactuar con base de datos.

El motivo de utilizar SQL se debe al gran volumen de datos con el que se está trabajando. El uso de bases de datos relacionales permite realizar consultas a los datos con tiempos de acceso mucho menores a los tiempos de acceso a archivos CSV (*Comma Separated Values*).

3.1.4 Instalación de librerías a utilizar

A continuación, en la Tabla 3, se listan las librerías utilizadas para desarrollo del trabajo, haciendo una breve referencia a su propósito.

Librería	Tarea
Pandas	Procesamiento de datos
Numpy	Cálculo numérico y análisis de datos de gran volumen
SQLite3	Provee las herramientas necesarias para trabajar con bases de datos SQLite3
Sklearn	Aprendizaje automático de propósito general
Matplotlib	Visualización de los datos
Seaborn	Visualización de los datos

Tabla 3 Librerías empleadas

3.2 Justificación del algoritmo empleado: *Random Forest*

Como ya se explicó en la introducción del capítulo, el algoritmo escogido para desarrollar el sistema de inteligencia artificial que realice la determinación del tipo de buque ha sido *Random Forest*. En el presente apartado se pretende justificar la selección de dicho algoritmo y exponer las ventajas que nos aporta a la hora desarrollar la tarea requerida.

La primera decisión que hay que tomar a la hora de seleccionar el algoritmo a utilizar, es determinar si se van a utilizar técnicas de aprendizaje supervisado o de aprendizaje no supervisado. No hay que olvidar que las técnicas de aprendizaje supervisado requieren de la existencia de datos etiquetados, mientras que los de aprendizaje no supervisado no requieren de los mismos.

Para responder a la pregunta planteada debemos observar nuestro problema. El fin de nuestro modelo es clasificar nuevos datos de entrada entre unas posibles salidas bien definidas. Los algoritmos de aprendizaje supervisado son perfectos para este tipo de tareas y además resultan mucho más sencillos de programar que los algoritmos de aprendizaje no supervisado. Sin embargo, los algoritmos de aprendizaje supervisado presentan el problema de que requieren de un gran volumen de datos etiquetados para entrenarlos. En nuestro caso ya disponemos de una base de datos, considerablemente grande, cuyos datos ya se encuentran etiquetados (el campo de *Shiptype* dentro de los mensajes AIS cubierto), por lo que la mejor opción es la de utilizar técnicas de aprendizaje supervisado.

Una vez decidido que se emplearán técnicas de aprendizaje supervisado, solo nos queda decidir qué algoritmo vamos a utilizar. Lo que buscamos es un algoritmo que sea capaz de manejar una gran cantidad

de datos y que sea capaz de realizar predicciones con precisión. Como ya se vio en el capítulo 2 de la memoria, las técnicas de aprendizaje supervisado más idóneas para esta tarea podrían ser SVM, *Gradient Boosting* o *Random Forest*. Finalmente se ha decidido optar por *Random Forest* debido a su capacidad de operar con una gran cantidad de datos, simplicidad y que se trata de una herramienta rápida, permitiendo así ejecutar varios modelos. Además, se trata de una herramienta flexible que nos permitirá dar diferentes enfoques al problema.

Random Forest ha sido utilizado en trabajos similares [81] obteniéndose resultados muy buenos que, a pesar de que puedan existir algoritmos que desempeñen mejor la tarea, acaba siendo un algoritmo más efectivo en cuanto a resultados obtenidos en base al esfuerzo aplicado.

3.3 Estudio de los datos

Como ya se explicó en el capítulo anterior, para entrenar un algoritmo un requisito fundamental es disponer de una base de datos para alimentar una inteligencia artificial que sea capaz de predecir el tipo de buque para las tramas AIS que no dispongan de dicho campo cubierto, ya sea mediante la detección de patrones o similitudes entre los objetos de la base de datos. Como ya se ha aclarado previamente, utilizaremos técnicas de aprendizaje supervisado, por lo que para el desarrollo de nuestro algoritmo utilizaremos datos etiquetados, es decir, los datos identifican en cada uno de los casos de qué tipo de buque se trata, que es la variable que se quiere predecir.

En el caso de nuestro trabajo, las bases de datos empleadas han sido generadas a partir de los mensajes AIS que han ido transmitiendo los diferentes buques y son recogidos por el COVAM para crear conjuntos de datos de los mensajes recibidos durante una determinada franja temporal. Es importante recordar que existen diferentes mensajes AIS transmitidos por parte del buque. Dentro de todos estos tipos de mensajes hay dos tipos que resultan de interés para el desarrollo del trabajo: los mensajes que recogen datos dinámicos del viaje (mensajes de tipos 1, 2 y 3) y los que recogen datos dimensionales de la plataforma transmisora (mensajes de tipo 5).

3.3.1 Datos dinámicos

El primero de los tipos de mensajes (tipos 1, 2 y 3) recoge datos relativos al estado dinámico del buque, es decir, a las variables que no son permanentes y varían en función del tiempo para cada buque. Este tipo de mensajes se transmiten con alta periodicidad, por norma general de dos a diez segundos entre mensajes, para poder llevar a cabo un seguimiento del movimiento del buque principalmente con el propósito de aportar seguridad a la navegación. En el presente trabajo se utilizarán como una fuente de información más que nos servirá para el cálculo de atributos que aporten una mayor cantidad de variables a la hora de llevar a cabo el aprendizaje automático.

A partir de todos estos mensajes, se ha generado un archivo CSV que recoge información de las tramas AIS transmitidas de todo el tráfico marítimo mundial para una franja temporal de catorce días que sigue la estructura que se puede ver en el Anexo III: Campos del archivo CSV de datos dinámicos original. Además, este archivo se ha enriquecido con otros datos que pueden resultar de interés, como por ejemplo la celda H3, la microzona o la macrozona. Este archivo supone el punto de partida en lo referente al uso de datos dinámicos para el entrenamiento del algoritmo. De todos los campos disponibles, se han seleccionado los que se estima que pueden ser de utilidad para el aprendizaje automático, es decir, los que nos permitan diferenciar los distintos contactos o permitan observar patrones en las características de estos. Estos campos son los siguientes:

1. ***MMSI*:** El número de identificación del servicio móvil marítimo es una serie de 9 dígitos que identifica inequívocamente a cada estación transmisora.
2. ***IMO*:** Número de 7 dígitos asignados para cada buque de forma única, con el propósito de identificarlo y auxiliar en la consecución de la seguridad marítima.
3. ***Timestamp*:** Es la hora de recepción del mensaje AIS por parte del sistema receptor.
4. ***Latitude*:** Latitud geográfica.

5. ***Longitude*:** Longitud geográfica.
6. ***COG* (*Course Over Ground*):** Rumbo del contacto.
7. ***SOG* (*Speed Over Ground*):** Velocidad del contacto.
8. ***Shiptype:*** Código de dos dígitos que define el tipo de buque del emisor del mensaje.

Una vez estudiado el origen y la metodología de los datos, se pasa a realizar un primer estudio de los datos disponibles del tráfico marítimo mundial para una franja temporal de catorce días. En primer lugar, este tipo de mensajes AIS es transmitido también por las plataformas de señalización marítima y otras unidades que transitan por encima de la superficie, pero no son considerados buques. Estos casos cuentan con un número a partir del cien en el campo *shiptype* de los mensajes AIS, entre cien y ciento treinta para los elementos de señalización y de ciento treinta y uno a doscientos cincuenta y cinco para el resto, que nos permite ver fácilmente de qué se trata. Para observar esta clasificación se ha definido el diccionario *platform* en *Python*:

```
platform = dict.fromkeys(list(range(0,100)), 'ship')
platform.update(dict.fromkeys(list(range(100,131)), 'beacon'))
platform.update(dict.fromkeys(list(range(131,256)), 'above'))
platform.update({0:'zero', -1:'nan'})
```

Reemplazando los valores del campo *shiptype* por su homólogo en el diccionario, se ha podido llevar a cabo una representación de los datos en base a la clasificación descrita que se puede ver en la Figura 3-2.

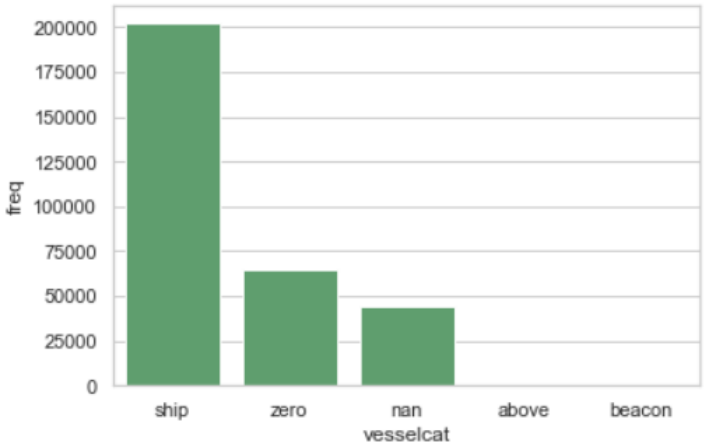

Figura 3-2 Representación del campo *shiptype* en el *dataset* inicial

Como podemos observar en la Figura 3-2, en una primera aproximación de los datos en bruto, la mayoría de los mensajes AIS tiene cubierto el campo *shiptype* con un valor establecido dentro de los parámetros que lo distinguen como un buque. Sin embargo, gran parte de los mensajes llegan con un valor de cero o sin especificarse, por lo que estos datos no podrán ser utilizados para el entrenamiento. Por otro lado, se aprecia que el resto de las categorías es despreciable.

La Figura 3-2 nos permite además darnos cuenta de la importancia del desarrollo de un clasificador automático del tipo de buque, de una muestra de más de trescientos mil datos AIS más de un tercio de los datos no tienen el campo del tipo de buque cubierto, aproximadamente un treinta y siete por ciento. Es precisamente la existencia de estos mensajes lo que motiva la realización del presente TFG.

Si nos centramos en los datos que transmiten correctamente como buque y descartamos los que no tienen el campo cubierto o le asignan un valor erróneo, podemos observar de una forma desglosada de qué tipo de buque se trata en particular. Para ello se ha definido un diccionario llamado *shiptype* a partir del código que aparece en el campo *shiptype*. El código referente a dicho diccionario sería el siguiente:

```
shiptype = dict.fromkeys(list(range(1,20)), 'reserved')
shiptype.update(dict.fromkeys(list(range(20,30)),'Wing In Ground'))
shiptype.update({30:'fishing',31:'tug',32:'tug',33:'dredger',34:'diving
ops',35:'military',36:'sailing',37:'pleasure craft'})
shiptype.update(dict.fromkeys(list(range(38,40)),'reserved'))
shiptype.update(dict.fromkeys(list(range(40,50)),'High speed craft'))
shiptype.update({50:'pilot',51:'SAR',52:'tug',53:'port tender',54:'Anti-
pollution equipment',55:'Law Enforcement'})
shiptype.update(dict.fromkeys(list(range(56,59)),'Spare-Local Vessel'))
shiptype.update({59:'noncombatant'})
shiptype.update(dict.fromkeys(list(range(60,70)),'Passenger'))
shiptype.update(dict.fromkeys(list(range(70,80)),'Cargo'))
shiptype.update(dict.fromkeys(list(range(80,90)),'Tanker'))
shiptype.update(dict.fromkeys(list(range(90,100)),'Other type'))
shiptype.update({0:'Not available', -1:'NaN'})
```

En este diccionario se recogen todos los tipos que los mensajes AIS identifican. El diccionario se ha generado en base al ya mencionado anteriormente Anexo II: Códigos numéricos identificadores del tipo de buque en los datos AIS. Gracias a este diccionario podemos estudiar la cantidad de buques de cada clase disponibles en nuestros datos para la alimentación del algoritmo de *Machine Learning*. Al igual que con el diccionario *platform*, gracias a la función de *Python* se ha podido realizar una representación de todos estos tipos que se muestra en la Figura 3-3. Esta representación ya excluye los datos que contienen valores no validos o que según nuestro diccionario serían considerados *NaN* o *Not available*.

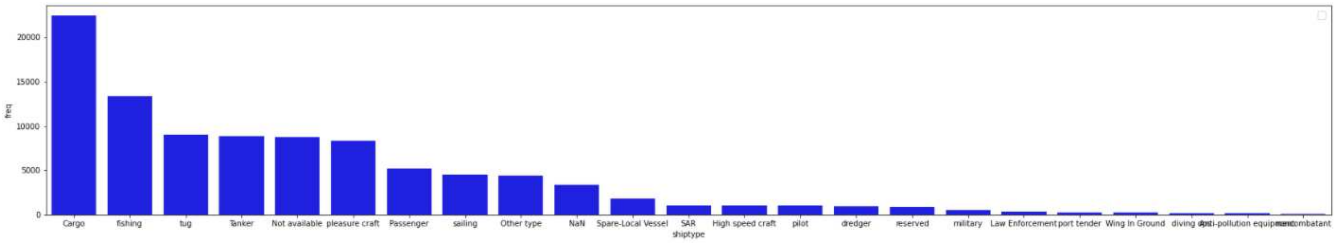

Figura 3-3 Representación del campo *shiptype* una vez reemplazado mediante el diccionario *shiptype*

Los resultados de la Figura 3-3 ya tienen en consideración la posibilidad de que existan varios mensajes con un mismo MMSI o IMO, solo se representan los campos que cuentan con un MMSI único. Con respecto al IMO también se seleccionan solo los que poseen IMO único, pero también se recoge los que transmiten sin IMO y poseen MMSI único, puesto que existe una gran cantidad de buques que no transmite IMO. Por norma general, los datos provenientes de buques que transmiten ambos campos (MMSI e IMO) son de mejor calidad que el resto. En la Figura 3-4 se puede observar una comparativa del total de los datos con MMSI único (en azul) con respecto a cuantos de los mismos transmiten con MMSI e IMO único y distinto de cero (en amarillo). La Figura 3-4 solo muestra los tipos de buque con mayor población, añadiendo los buques militares.

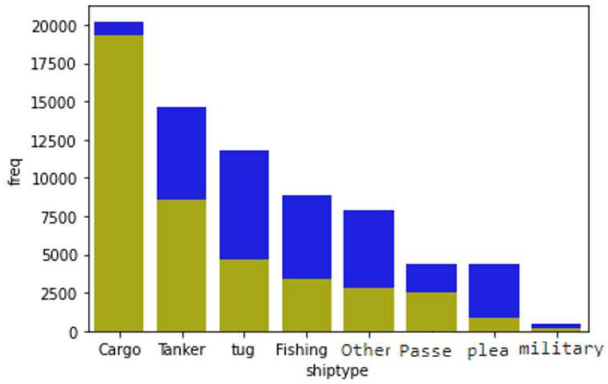

Figura 3-4 Representación de tramas AIS con MMSI único frente a las que tienen MMSI+IMO único

Pese a que los mensajes que tienen MMSI único son más numerosos, la calidad de los datos con IMO único es mayor. Además, cuantos más datos se quieran procesar, más tiempo se tardará en realizar la consulta y requerirá de más memoria y capacidad de cómputo. Es por ello por lo que en el caso de los datos estáticos se ha primado la calidad con respecto a la cantidad, como veremos más adelante.

En la Figura 3-5 podemos ver un resumen del estudio de los datos realizado a lo largo del apartado. Para entrenar el algoritmo nos interesa que los datos sean lo más fiables posible y tratar de reducir el número de datos que puedan añadir ruido a nuestro algoritmo.

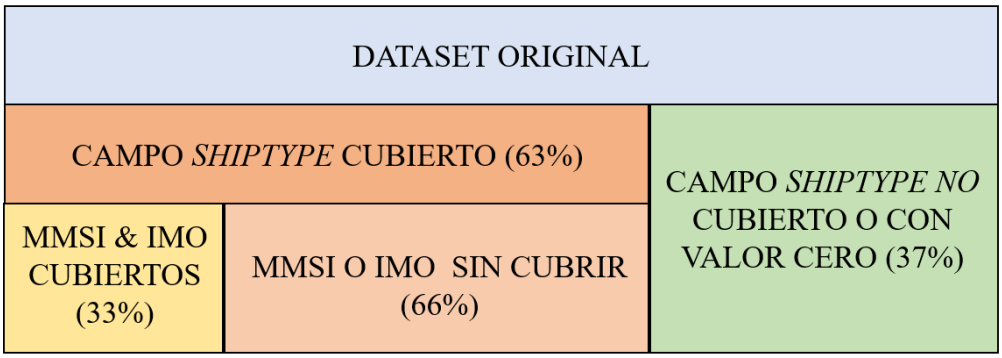

Figura 3-5 Resumen del estudio del *dataset* de datos dinámicos generado a partir de mensajes AIS tipo 1, 2 y 3 (fuente: propia)

3.3.2 Datos estáticos

El segundo tipo de mensajes AIS (tipo 5) que se ha empleado para la extracción de datos para el desarrollo del modelo es el que recoge datos dimensionales de la plataforma emisora, es decir, datos que no varían con el tiempo y que son propios de cada buque desde su puesta en funcionamiento. A diferencia de los mensajes de datos dinámicos, los mensajes AIS que transmiten datos estáticos se transmiten con menos frecuencia, en concreto, por norma general, cada seis minutos. Esto se debe precisamente a que, como ya hemos mencionado, son datos que no varían, por lo que su necesidad a la hora de actualizarlo repetidamente es relativamente innecesaria.

Al igual que con los datos dinámicos, se ha generado un script en *Python* para la lectura, decodificación y escritura de la segunda base de datos de entrenamiento que será utilizada para proporcionar un enfoque diferente al problema utilizando datos estáticos en lugar de dinámicos. Al igual que el conjunto de datos anterior, este *dataset* ha sido generado a partir de la información recibida en las tramas AIS transmitidas por el tráfico marítimo mundial en una franja temporal de catorce días. Los campos disponibles en este caso son los siguientes:

1. ***MMSI*:** Mismo significado que en la anterior base de datos.

2. ***to_bow*:** en los mensajes AIS se establece como el campo *A*. Es la distancia desde el transpondedor AIS hasta la proa del buque en metros.
3. ***to_stern*:** en los mensajes AIS se establece como el campo *B*. Es la distancia desde el transpondedor AIS hasta la popa del buque en metros.
4. ***to_port*:** en los mensajes AIS se establece como el campo *C*. Es la distancia desde el transpondedor AIS hasta la banda de babor en metros.
5. ***to_starboard*:** en los mensajes AIS se establece como el campo *D*. Es la distancia desde el transpondedor AIS hasta la banda de estribor en metros.
6. ***Draught*:** calado del buque en metros.
7. ***Shiptype*:** En este caso, a diferencia de los datos dinámicos, el *dataset* original contiene el tipo de buque escrito explícitamente en lugar del código numérico.

Para facilitar la comprensión de los campos *A, B, C* y *D* se aporta una representación gráfica de los mismos en la Figura 3-6.

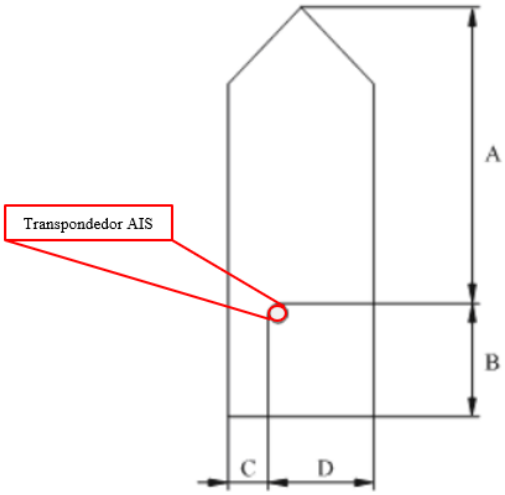

Figura 3-6 Representación gráfica de los campos *A, B, C* y *D* [90]

En este caso, solo disponemos del MMSI para diferenciar un buque de otro o para la eliminación de tramas duplicadas. La menor periodicidad de los mensajes de tipo 5 hace que las tramas duplicadas sean menores. Para nuestro conjunto de datos a utilizar se ha realizado un análisis de los datos totales disponibles para determinar su idoneidad a la hora de trabajar con ellos. En la Figura 3-7 se puede ver el resultado de dicho análisis resumido en una representación gráfica que compara el conjunto total de los datos (en azul) y las tramas dentro de estos datos que no tienen MMSI duplicado o valores no válidos (campos no cubiertos o igual a cero) para entrenar nuestro algoritmo (en rojo). Los tipos de buque que se tienen en consideración son únicamente los que resultan de interés para los objetivos de este trabajo.

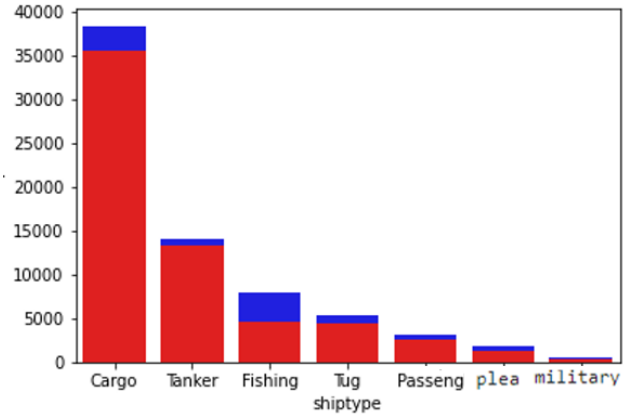

Figura 3-7 Representación de tramas AIS de tipo 5 comparando el total con las tramas óptimas para su uso

Como podemos observar en la Figura 3-7, la pérdida de datos es prácticamente despreciable, por lo que asumiremos dichas pérdidas obteniendo como resultado el conjunto de datos que alimentará el algoritmo.

3.3.3 Combinación de datos estáticos y dinámicos

Tras la observación de la existencia de dos fuentes diferentes de información, se plantea la posibilidad de crear un conjunto de datos que contenga tanto datos estáticos como dinámicos. Para ello, se tienen como punto de partida los dos conjuntos de datos con los que estamos trabajando (datos estáticos y datos dinámicos) y se buscan elementos que permitan asociar las tramas de ambos conjuntos a un mismo barco, es decir, buscar un elemento común que sea capaz de relacionarlas.

Si se tienen en cuenta los campos que poseen cada uno de los conjuntos de datos, podemos observar que el único elemento común entre los mismos es el MMSI. Puesto que el campo MMSI está cubierto con un código teóricamente único para cada buque, podemos decir que los datos con un mismo MMSI en ambos conjuntos de datos (datos estáticos y datos dinámicos) se refieren al del mismo buque.

Una vez decidido que utilizaremos el MMSI solo nos queda combinar los conjuntos de datos. Para ello se ha hecho uso de una función que posee SQL llamada *INNER JOIN* que nos permite realizar la unión de dos conjuntos de datos en los cuales existe un elemento en común. La salida sería el conjunto de datos en el que se recogen todos los datos que existen para un determinado valor clave, que es el elemento común (MMSI), tanto de una tabla como de la otra.

Para poder realizar esta acción, primero ha habido que guardar los datos estáticos (tabla *BestStatic*) y los datos dinámicos (tabla *BestDinamic*) en la misma base de datos SQL. Posteriormente, se emplea la siguiente sentencia para realizar la unión:

```
combined=pd.read_sql("""SELECT * FROM BestDinamic INNER JOIN BestStatic USING
(mmsi)""", con=mixevi)
```

El resultado de la sentencia se ha guardado en otra tabla SQL para agilizar las consultas, obteniéndose el conjunto de datos que se puede ver representado en la Figura 3-8 en función del valor del campo *shiptype*. Se puede observar que los datos referentes a un mismo buque parecen ser suficientes como para entrenar a un modelo que se nutra de los dos conjuntos de datos disponibles (datos estáticos y datos dinámicos).

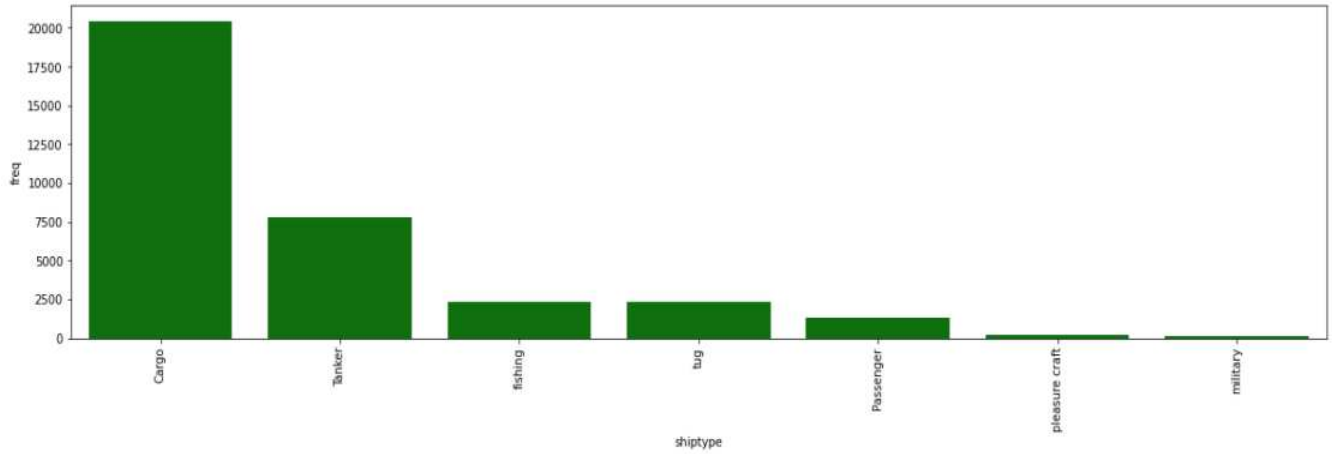

Figura 3-8 MMSI únicos de cada tipo de buque después de combinar datos estáticos y dinámicos

3.4 Preprocesado de los datos

Una vez descritos las herramientas y los datos que vamos a utilizar para el desarrollo del clasificador, a continuación, se irán explicando los pasos a seguir hasta la obtención del modelo y su optimización.

En este apartado se explicará el procedimiento seguido para el filtrado preliminar de los datos con el fin de obtener un conjunto de datos optimizado y que permita facilitar y agilizar las consultas al mismo.

Con preprocesado se hace referencia a la adaptación de los datos disponibles para que puedan ser utilizados para entrenar nuestro algoritmo correctamente. Estas tareas consisten sobre todo en la eliminación de las tramas AIS del conjunto de datos inicial que, en primera instancia, no resultan de utilidad por estar incompletas o por contar con valores ilógicos, por ejemplo, que la velocidad de un barco sea superior a doscientos nudos, así como en la selección de los datos que resultan de interés dentro del conjunto original, con el objetivo reducir la cantidad de datos con los que se trabaja a los necesarios. De lo contrario, estaríamos continuamente obligando al algoritmo a procesar datos que no va a utilizar en ningún momento.

En ambos conjuntos de datos la forma de proceder ha sido muy similar. En primer lugar, debido al gran volumen de datos que se está manejando, hubo que plantear un bucle que fragmentara los datos y los procesara en bloques. Una vez fragmentado, para cada subconjunto realizamos una selección de los campos que vamos a utilizar para la creación de atributos o distinción de los datos. Además, y únicamente para los datos dinámicos, se pasa el campo *shiptype* por el diccionario *shiptype* descrito anteriormente para adaptar dicho campo para su manejo. A continuación, se eliminan los datos nulos o con valores que no pueden ser utilizados para cálculos numéricos. Por último, seleccionamos los tipos de buque que vamos a predecir, que son aquellos que aparecen en un documento emitido por el COVAM que recoge las anomalías que les resultan de interés en sus labores de vigilancia marítima y, por lo tanto, son los que se quiere conocer. Finalmente, los datos que hayan superado el cribado serán guardados en una tabla dentro de la base de datos SQL para facilitar su posterior manejo.

Puesto que el código empleado tanto para los datos estáticos como dinámicos presenta una misma estructura y es prácticamente similar a excepción de ciertos nombres de campos y archivos, se muestra a continuación únicamente el código empleado para los datos dinámicos con el fin de ilustrar la explicación anterior.

```
orig=['TIMESTAMP','MMSI''IMO','COG','SOG','LATITUDE','LONGITUDE','LOA','SHIPTYP
E AIS RAW']
subs=['tim','imo','cog','sog','lat','lon','loa','shiptype']


reader=pd.read_csv('../Gonzalo/DataDinamic.csv',sep=';', chunksize=1000000)
for df in reader:
    chunk=df[orig]
    chunk.columns = subs
    chunk=chunk.replace({'shiptype': shiptype})
    chunk=chunk[chunk.replace([np.inf,-np.inf],np.nan).notnull().all(axis=1)
    chunk=chunk.dropna()
    df = chunk[chunk.shiptype.isin(['fishing','tug','military','pleasure
    craft','Passenger','Cargo','Tanker','Other type',])]
    df.to_sql('dinamic', con=conn, if_exists='append', index=False)
```

Por último, y puesto que queremos datos de la mayor calidad posible, se descartarán los buques que puedan generar dudas con respecto a la fiabilidad de los datos o que no aporten información relevante. Este es el caso de los buques que compartan MMSI para el conjunto de datos combinados, puesto que si hay más de un barco con el mismo MMSI no tenemos forma de demostrar de cual se trata en realidad. Además, en el conjunto de datos dinámicos se eliminan aquellos datos que mantengan constante todos los parámetros en todos los datos que transmiten (buques parados o a velocidad y rumbo constante) porque no nos aportan información que les distinga con respecto a otros.

3.5 Creación de atributos

Una vez se ha realizado un primer cribado de los datos disponibles, el siguiente paso a seguir en el desarrollo de nuestro algoritmo es la creación de los atributos que el algoritmo empleará para discernir un tipo del buque del resto. Puesto que disponemos de dos fuentes de datos diferentes, el resultado de este proceso dará como resultado dos clases de atributos distintas: atributos estáticos y atributos dinámicos.

3.5.1 Atributos estáticos

Los atributos estáticos son aquellos que se han creado a partir de los mensajes de tipo 5. Estos atributos son los que podemos ver listados y descritos en la Tabla 4.

Atributo	Descripción	Fórmula
len	Eslora del contacto. Resultante de la suma de *A* y *B*.	$A + B$
wid	Manga del contacto. Resultante de la suma de *C* y *D*.	$C + D$
ldivw	Cociente entre eslora y manga.	$len/_{\omega id}$
ldivd	Cociente entre eslora y calado.	$len/_{draught}$
wdivd	Cociente entre manga y calado.	$wid/_{draught}$
area	Área aproximada de la planta del contacto. Resultante de multiplicar la eslora por la manga.	$len \times \omega id$
grith	Suma de la eslora y la manga.	$len + wid$
aml	Área longitudinal sumergida aproximada. Resultante de multiplicar la eslora por el calado.	$len \times draught$
amt	Área transversal sumergida aproximada. Resultante de multiplicar la manga por el calado.	$wid \times draught$
vs	Volumen sumergido aproximado. Resultante de multiplicar la eslora por la manga y por el calado	$len \times wid \times draught$
aol	Tanto por uno que supone la distancia *A* sobre la eslora total del buque. Resultante de dividir *A* entre la eslora.	$A/_{len}$

Tabla 4 Atributos estáticos creados a partir de datos AIS

A estos atributos se le suman los parámetros *A, B, C, D* y calado, que también nos pueden servir para distinguir un tipo de buque de otro.

Para el cálculo de atributos y su anexión al conjunto de los datos se ha definido la función *statics*. El código en el que se define esta función es el que se muestra a continuación:

```
def statics(df):
    df['len']=df.a+df.b# Eslora
    df['wid']=df.c+df.d# Manga
    df['ldivw']= df.len/df.wid# Cociente entre eslora y manga
    df['ldivd']= df.len/df.draught# Cociente entre eslora y calado
    df['wdivd']= df.wid/df.draught# Cociente entre manga y calado
    df['area']= df.len*df.wid# Area de la cubierta aproximada= eslora x manga
    df['grith']= df.len+df.wid# Suma de eslora y manga
    df['aml']=df.len*df.draught# area longitudinal sumergida
    df['amt']=df.wid*df.draught# area transversal sumergida
    df['vs']=df.len*df.draught*df.wid # volumen sumergido
    df['aol']=df.a/df.len# proporción que supone a sobre la eslora total
    return df
```

3.5.2 Atributos dinámicos

Los atributos dinámicos son aquellos que se han creado a partir de los mensajes de los tipos 1, 2 y 3. Los atributos dinámicos se crean a partir del rumbo, la velocidad y la posición que trasmite cada buque en los mensajes AIS. Además, estos mensajes llevan consigo un sello temporal, el cual también está recogido en uno de sus campos, que nos permitirá realizar un estudio del movimiento de cada buque y así crear atributos dinámicos.

La creación de atributos dinámicos consiste en el cálculo de la cinemática de los contactos en función de mensajes sucesivos que permiten describir su movimiento. Es por ello que antes de realizar cualquier cálculo los datos se agrupan por su IMO y se ordenan temporalmente. Posteriormente, con todas las variables cinemáticas calculadas para cada una de las tramas se aplicarán estadísticos cuyo valor resultante serán los atributos a emplear.

Las variables cinemáticas calculadas son las que se pueden apreciar en la Tabla 5 (en la misma se obvian las conversiones de las variables a las unidades correspondientes).

Variable	Descripción	Fórmula
dtime	Distancia temporal entre mensajes sucesivos.	Δtim
nmi	Distancia espacial entre mensajes sucesivos.	$\sqrt{(\Delta lat)^2 + (\Delta lon)^2}$
sog_hat	Velocidad calculada a partir de la distancia espacial entre mensajes.	$nmi/dtime$
sog_err	Error de la *sog_hat* en función de la velocidad transmitida en el mensaje AIS.	$(sog\_hat - sog)/sog$
acc_hat	Aceleración calculada a partir de la distancia espacial entre mensajes.	$\Delta sog\_hat/dtime$
jerk_hat	Sobreaceleración calculada a partir de la distancia espacial entre mensajes.	$\Delta acc\_hat/dtime$
acc	Aceleración calculada a partir de la distancia temporal entre mensajes.	$\Delta sog/dtime$

jerk	Sobreaceleración calculada a partir de la distancia temporal entre mensajes.	$\Delta acc/_{dtime}$
omega	Velocidad de caída de rumbo	$\Delta cog/_{dtime}$
alpha	Aceleración de caída de rumbo	$\Delta omega/_{dtime}$
zeta	Sobreaceleración de caída de rumbo	$\Delta zeta/_{dtime}$

Tabla 5 Variables cinemáticas calculadas

Para el cálculo de las variables cinemáticas y su anexión al conjunto de los datos se ha definido la función *cinematics*. El código en el que se define esta función es el que se muestra a continuación:

```
def cinematics(df):
    #Tiempo
    df = df.sort_values(by='tim')
    df.tim = pd.to_datetime(df.tim,unit='ms')
    df['dtime'] = df.tim.diff().dt.seconds
    hours = df.dtime/3600

    # Variables lineales
    dy = np.radians(df.lat.diff())*6373/1.852
    dx = np.radians(df.lon.diff())*6373/1.852
    df['nmi'] = np.sqrt(dx**2 + dy**2)
    df['sog_hat'] = df.nmi/hours
    df['sog_err'] = (df.sog_hat-df.sog)/(df.sog+0.1)*100
    df['acc_hat'] = df.sog_hat.diff()/hours
    df['jerk_hat'] = df.acc_hat.diff()/hours
    df['acc'] = df.sog.diff()/hours
    df['jerk'] = df.acc.diff()/hours

    # Variables angulares
    deg_norm = df.cog.diff()%360
    deg_diff = pd.concat([360-deg_norm, deg_norm], axis=1).min(axis=1)
    df['omega'] = np.radians(deg_diff)/hours
    df['alpha'] = df.omega.diff()/hours
    df['zeta'] = df.alpha.diff()/hours
    return df
```

Una vez hemos calculado todas las variables cinemáticas para cada uno de los buques que existen en nuestro conjunto de datos, nos surge el problema de que al calcularse las variables para cada mensaje solo teniendo en cuenta el mismo y el inmediatamente anterior, existirán diferentes valores de la cinemática para cada contacto. La solución ya se ha adelantado anteriormente, que consiste en aplicar estadísticos a las diferentes variables cinemáticas de cada buque para obtener así un único valor representativo de cada una de las variables cinemáticas por buque.

Los estadísticos empleados son los que se pueden apreciar en la Tabla 6. La forma de calcular estos estadísticos es mediante funciones disponibles en las librerías nombradas al principio del capítulo.

Estadístico	Descripción
avg	Media.
sdv	Desviación estándar.
iod	Cociente de la varianza entre la media.
q50	Mediana.
skw	Cálculo de la asimetría del conjunto de datos.
krt	Valor que representa la curtosis de los datos.

Tabla 6 Estadísticos que conformarán los atributos dinámicos

Para la aplicación de estadísticos, la creación de atributos y la anexión de los atributos calculados al conjunto de los datos, se ha definido la función *aggregate.* A continuación, se muestra la función, pero por simplificación solo se muestra para su aplicación en el campo *sog*, puesto que para el resto de los campos la forma de proceder es la misma:

```
def aggregate(df):
    df['avg_sog'] = df['sog'].mean()
    df['sdv_sog'] = df['sog'].std()
    df['iod_sog'] = df['sog'].var()/df['sog'].mean()
    df['q50_sog'] = df['sog'].quantile(0.50)
    df['skw_sog'] = df['sog'].skew()
    df['krt_sog'] = df['sog'].kurt()
    return df
```

3.6 Optimización final de los datos

Una vez realizados todos los pasos anteriores, ya se estaría en disposición de comenzar con las técnicas de *Machine Learning* para el desarrollo del modelo. Sin embargo, antes de exponer el entrenamiento y puesta en marcha del algoritmo, se ha considerado importante reseñar las diferentes herramientas de ingeniería de datos que se han utilizado para realizar los diferentes experimentos.

La necesidad de usar estas herramientas para desarrollar el modelo no está impuesta, pues tal y como están los datos ya se puede entrenar el algoritmo. Sin embargo, como veremos en el siguiente capítulo, estas herramientas pueden hacer variar el rendimiento de nuestro modelo. De hecho, estas herramientas se pueden usar de forma simultánea en algunos casos.

3.6.1 Oversampling

En algunas ocasiones nos podemos encontrar con que los datos disponibles para alimentar los algoritmos de aprendizaje automático se encuentran distribuidos de forma desequilibrada, es decir, dentro de las clases que se quieren predecir existe una proporción mucho mayor de unas clases sobre otras.

Esta clase de sucesos puede resultar perjudicial para el rendimiento del algoritmo. Esto se debe a que debido a la escasez de datos de la clase minoritaria es posible que el algoritmo las considere como “ruido” dentro del conjunto de datos y la ignore. El problema que se genera es que en muchas ocasiones la clase que nos interesa estudiar es precisamente la clase minoritaria [91].

Para combatir este tipo de problemas de desequilibrio de los datos, existen diferentes técnicas entre las cuales podemos destacar la que se trata en este apartado en cuestión: el *oversampling*.

Esta técnica consiste en la creación de datos de las clases minoritarias con el objetivo de que el número de datos de las clases menos densas en cuanto a cantidad de datos antes de llevar a cabo *oversampling* pase a ser más relevante a la hora de ser procesada por el algoritmo. Para la generación de datos lo que suele realizar el algoritmo es generar datos sintéticos que posean características similares a los datos de la clase a la que se quieran que pertenezcan.

En la Figura 3-9 podemos ver un ejemplo aplicado precisamente a los datos de nuestro estudio. A la izquierda observamos cómo existen muchos más datos de la clase *Cargo* que de la clase *PleasureCraft*. Al ejecutar *oversampling* sobre nuestro conjunto de datos obtendremos un nuevo conjunto de datos en el que las clases minoritarias han aumentado su volumen, como podemos ver a la derecha.

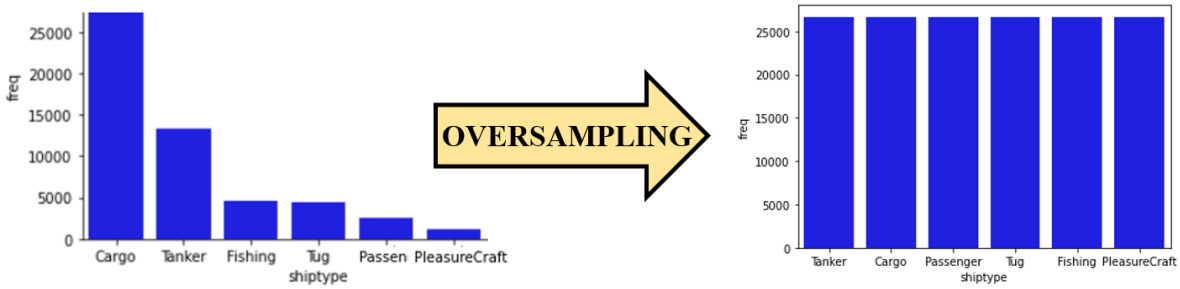

Figura 3-9 Ejemplo de *oversampling* (fuente: propia)

En el ejemplo de la Figura 3-9 los datos de las clases minoritarias aumentan hasta igualar la mayoritaria. Sin embargo, la cantidad de datos que se adicionan al conjunto es seleccionable por el usuario, no necesariamente hay que igualar la clase mayoritaria.

Este tipo de técnicas proporciona la ventaja de que se igualan las clases sin que se produzca una pérdida de información. Sin embargo, se corre el riesgo de que el algoritmo se sobreajuste a la hora de detectar datos de la clase minoritaria, pues estos nuevos datos no dejan de ser réplicas bastante similares de los datos ya existentes.

Para la ejecución en *Python* de esta práctica se utilizará la función SMOTE. Bastará con ejecutar la siguiente sentencia en la que los datos de entrenamiento originales están señalados con la palabra *train* y los nuevos después del *oversampling* están seguidos del término *res*:

```
oversample=SMOTE(random_state = 42)
X_res, Y_res=oversample.fit_resample(X_train, y_train)
```

3.6.2 Undersampling

El *undersampling*, al igual que el *oversampling*, es un tipo de técnica que se emplea para lidiar con los conjuntos de datos desequilibrados que se han explicado anteriormente. Aunque el objetivo es el mismo, que es igualar la proporción de los datos de cada clase para que el modelo las distinga como tales, la forma de proceder es completamente opuesta [91].

Cuando se utilizan técnicas de *undersampling* sobre los datos lo que se hace es eliminar datos de las clases mayoritarias hasta que estas clases dejan de ser muy dispares con las clases minoritarias, tal y como se puede ver en la Figura 3-10. Este tipo de técnicas soluciona el problema de la falta de datos de las clases minoritarias e incluso puede llegar a mejorar el tiempo de entrenamiento del modelo debido a que se procesa una menor cantidad de datos. Sin embargo, presenta la desventaja de que se elimina información útil y que la selección de los datos que se mantienen intactos es aleatoria, por lo que los datos de una clase pueden llegar a ser un conjunto no representativo de la misma.

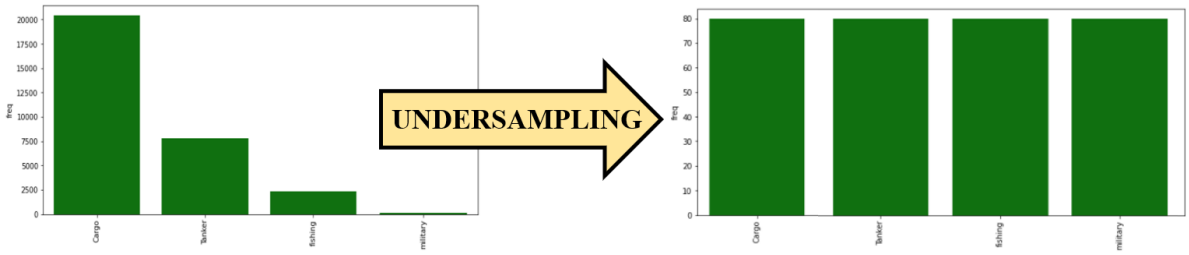

Figura 3-10 Ejemplo de *undersampling* (fuente: propia)

Para hacer uso de técnicas de *undersampling* sobre los datos de entrenamiento simplemente se ha hecho uso de la siguiente sentencia:

```
under = RandomUnderSampler()
X_res, Y_res=under.fit_resample(X_train, y_train)
```

3.6.3 MinMaxScaler

MinMaxScaler es una función que proporciona la librería *Sklearn* para la normalización/estandarización de variables. La importancia de tomar estas acciones radica en que muchos algoritmos funcionan mejor o convergen más rápido cuando las características con las que trabajan poseen una escala similar. Además, el empleo de este tipo de técnicas evita que, ante la entrada de características que están a mayor escala del resto de datos, el modelo se sobreajuste de forma errónea. Existen otras metodologías que realizan la misma función, sin embargo, se ha optado por esta por ser de las que nos ofrece *Sklearn* la que menos distorsiona los datos [92].

MinMaxScaler ajusta todas las características en función del valor máximo y mínimo, de tal forma que se conserva la forma de la distribución original de los datos, evitando que se modifique excesivamente la información presente en los datos originales. En la Figura 3-11 se puede ver un ejemplo en el que los datos de la izquierda son los originales y a la derecha después de hacer uso de la función *MinMaxScaler.* Su ejecución es tan simple como llamar a la función con el mismo nombre cuando se diseña el proceso de aprendizaje automático del algoritmo.

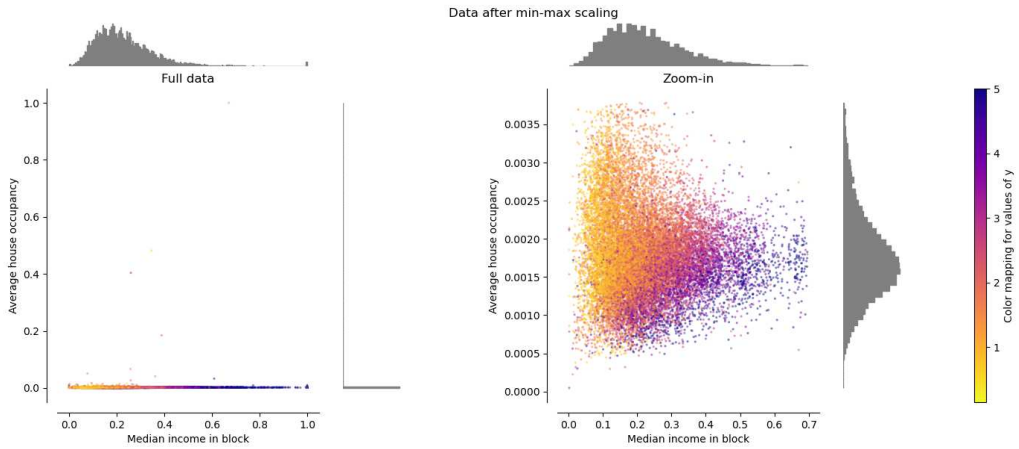

Figura 3-11 Ejemplo de uso de *MinMaxScaler* [93]

3.7 Machine Learning

Una vez generado el conjunto de los datos deseado, todo está listo para comenzar a entrenar el modelo. Lo primero que se debe realizar es determinar qué variable dentro del conjunto de datos es la dependiente, es decir, la variable de la que se quiere predecir su valor en función de las variables independientes. También debemos definir estas variables independientes con respecto a las cuales se trate de encontrar patrones o características en los datos para ser capaz de determinar el valor deseado.

En nuestro caso, la variable dependiente es el tipo de buque y las variables independientes son los atributos que se han creado a partir de las funciones *aggregate* y *statics.* También se han usado algunos campos de los mensajes AIS originales como variables independientes.

Una vez definidas ambas variables, el siguiente paso es dividir los datos en datos de entrenamiento y datos de test. Los datos de entrenamiento están constituidos por el conjunto de datos del *dataset* completo que se utilizará para que el algoritmo aprenda y sea capaz de clasificar los nuevos datos de entrada. Por otro lado, los datos de test son aquellos datos que nos reservamos del *dataset* original para evaluar el modelo una vez se haya finalizado el aprendizaje. El tamaño de los subconjuntos formados por los datos de entrenamiento y los datos de test los establece el usuario. Además, la selección de los datos para cada subconjunto es completamente aleatoria. Esta división de los datos se realizará ejecutando la siguiente línea de código:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state = 42)
```

El parámetro *test_size* es el que determina qué porcentaje de los datos será el utilizado como datos de test, expresado en tanto por uno. El resto de los datos constituirán los datos de entrenamiento. Como se puede observar, en este caso, el 75% de los datos se usarán para entrenar el modelo y el 25% restante para validarlo.

Una vez definidos los datos de entrenamiento y de test, se procede a la creación del modelo. Para ello se hace uso de otra de las funciones que nos ofrece *Sklearn* denominada *Pipeline*. Esta función lo que hace es simplemente permitirnos agrupar todos los parámetros que definan nuestro algoritmo facilitándonos así su elaboración y modificación a la hora de realizar diferentes experimentos. De este modo la configuración de nuestro modelo quedaría definida en una línea de código:

```
pipe = Pipeline([('scaler', MinMaxScaler()),('classifier',
RandomForestClassifier(n_estimators=100))])
```

Básicamente lo que se está haciendo es indicarle al sistema que el algoritmo a utilizar será *Random Forest*, se le indica el número de árboles (el cual se podrá variar para realizar diferentes experimentos) y que utilice la función *MinMaxScaler* sobre el conjunto de datos. Con esta sentencia el modelo ya estaría creado, por lo que solo falta entrenarlo y probarlo.

Para el entrenamiento y evaluación del modelo, la librería *Sklearn* nos proporciona diferentes funciones. En primer lugar, la función *fit*, la cual usaremos para entrenar el modelo. A esta función le pasaremos los datos que queremos que use para entrenar al modelo, tanto la variable dependiente como las independientes. Para evaluar el modelo usaremos la función *predict.* Esta función le pasa los datos sin etiquetar al modelo y este se encarga de devolver la predicción de la etiqueta perteneciente a cada uno de los datos en base a lo aprendido durante la fase de entrenamiento del algoritmo. Gracias a que parte de los datos han sido separados creando un conjunto de datos de test, podemos comparar la etiqueta que ha proporcionado el modelo a estos datos y compararla con la etiqueta real de los mismos. Para ello *Sklearn* nos brinda la función *classification_report*, la cual nos proporcionará información sobre la calidad de nuestro modelo en base a una serie de coeficientes que serán explicados en mayor profundidad en el siguiente capítulo. En la Figura 3-12 se muestra un ejemplo de cuál sería la salida proporcionada por esta funcion a partir de la cual se estará en condiciones de evaluar el modelo.

	precision	recall	f1-score	support
Cargo	0.62	0.87	0.72	3533
Passenger	0.79	0.46	0.58	452
Tanker	0.36	0.10	0.16	1592
fishing	0.61	0.48	0.54	621
tug	0.61	0.64	0.63	735
accuracy			0.61	6933
macro avg	0.60	0.51	0.53	6933
weighted avg	0.57	0.61	0.56	6933

Figura 3-12 Ejemplo de salida de la función *classification_report* (fuente: propia)

La ejecución en *Python* de todo el proceso descrito en el apartado anterior se puede resumir en las siguientes líneas de código:

```
model = pipe.fit(X_train, y_train) #entrenamiento del modelo
y_pred = model.predict(X_test) #predicción de la etiqueta de los datos de test
print(classification_report(y_test, y_pred, zero_division=0)) #informe
```

Otra función que nos proporciona *Sklearn* muy interesante y que ha sido empleada a la hora de desarrollar el código es la función *feature_importances_*. Esta función nos proporciona la importancia de cada uno de los atributos, es decir, cuán útiles son los atributos para discernir un tipo de buque de otro. Esta función nos devolverá la importancia de cada atributo, establecida en tanto por uno para todos los atributos utilizados.

3.8 Experimentos realizados

Todo lo redactado anteriormente en el presente capítulo compone los pasos a seguir desde la obtención del conjunto de datos inicial hasta el desarrollo de los clasificadores utilizando *Random Forest*. Como resultado final de este desarrollo se han generado tres scripts: *PrettyStatic.ipynb*, *PrettyDynamic.ipynb* y *PrettyMix.ipynb*.

El script *PrettyStatic.ipynb* es el que trabaja con los datos estáticos cuyo código completo se puede consultar en el Anexo IV: Código del archivo *PrettyStatic.ipynb*. El script *PrettyDynamic* es el que trabaja con los datos dinámicos y cuyo código completo se puede ver en el Anexo V: Código del archivo *PrettyDinamic.ipynb*. Finalmente, el script *PrettyMix.ipynb* es el que combina ambos conjuntos de datos cuyo código completo se puede ver en el Anexo VI: Código del archivo *PrettyMix.ipynb*. En ellos se detalla desde el procesado de los datos hasta el entrenamiento y prueba del algoritmo.

A continuación se muestran tres tablas (Tabla 7, Tabla 8, Tabla 9) en las que se definen los experimentos realizados a partir de los scripts mencionados, remarcando las diferencias entre ellos y cada una de las peculiaridades utilizadas en cada uno de los mismos. Para ello cada columna hace referencia a un parámetro que se tiene capacidad de variar para cada uno de los experimentos. Los campos de estas tablas son los siguientes:

1. ***Nº***: identificador que se le da a cada uno de los experimentos para posteriormente hacer referencia a los mismos.
2. ***Nº arb***: número de árboles empleados para la ejecución del algoritmo *Random Forest*.
3. ***MMS***: empleo de la función *MinMaxScaler* explicada previamente.
4. ***OverS***: empleo de técnicas de *Oversampling.*
5. ***UnderS***: empleo de técnicas de *UnderSampling*

6. ***De/Dt***: relación de los datos del *dataset* completo utilizados como datos que serán utilizados para entrenar al algoritmo y los utilizados para evaluar el modelo obtenido (Datos de entrenamiento/Datos de test). Está expresado en tanto por uno.
7. ***Atributos***: se hace referencia a los atributos que se le pasan al algoritmo para entrenarse y realizar la clasificación de los datos.

Más adelante, en el capítulo cuatro, se mostrarán y evaluarán los resultados obtenidos de los experimentos de los que se ha podido extraer información de interés.

3.8.1 Experimentos con conjunto de datos estáticos

En la Tabla 7 se muestran los experimentos realizados utilizando el conjunto de datos estáticos extraídos de los mensajes AIS de tipo 5.

Nº	Nº arb	MMS	OverS	UnderS	De/Dt	Atributos
1	100	SI	NO	NO	0,75/0,25	Todos los estáticos.
2	50	SI	NO	NO	0,75/0,25	Todos los estáticos.
3	200	SI	NO	NO	0,75/0,25	Todos los estáticos.
4	100	SI	NO	NO	0,75/0,25	Estáticos de los datos sin calcular nuevos atributos (*A, B, C, D* y *draught).*
5	50	SI	NO	NO	0,75/0,25	Estáticos de los datos sin calcular nuevos atributos (*A, B, C, D* y *draught).*
6	200	SI	NO	NO	0,75/0,25	Estáticos de los datos sin calcular nuevos atributos (*A, B, C, D* y *draught).*
7	100	SI	NO	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
8	100	SI	NO	NO	0,75/0,25	Tres atributos más importantes, según experimentos anteriores (*aol, ldivw, ldivd*).
9	200	SI	NO	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
10	50	SI	NO	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
11	100	SI	NO	NO	0,9/0,1	Todos los estáticos.
12	100	SI	NO	NO	0,8/0,2	Todos los estáticos.
13	100	SI	NO	NO	0,65/0,35	Todos los estáticos.
14	100	NO	NO	NO	0,75/0,25	Todos los estáticos.
15	100	NO	NO	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).

16	100	SI	SI	NO	0,75/0,25	Todos los estáticos.
17	100	SI	SI	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
18	100	SI	NO	SI	0,75/0,25	Todos los estáticos.
19	100	SI	NO	SI	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
En los siguientes experimentos se eliminan los buques militares del conjunto de datos						
20	100	SI	NO	NO	0,75/0,25	Todos los estáticos.
21	100	SI	NO	NO	0,75/0,25	Estáticos de los datos sin calcular nuevos atributos (*A, B, C, D* y *draught).*
22	100	SI	NO	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
23	100	SI	NO	NO	0,75/0,25	Tres atributos más importantes, según experimentos anteriores (*aol, ldivw, ldivd*).
24	100	SI	SI	NO	0,75/0,25	Todos los estáticos.
25	100	SI	SI	NO	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
26	100	SI	SI	NO	0,75/0,25	Cinco atributos más importantes, según experimentos anteriores (*aol, ldivw, wdivd, A, ldivd).*
27	100	SI	NO	SI	0,75/0,25	Todos los estáticos.
28	100	SI	NO	SI	0,75/0,25	Ocho atributos más importantes, según experimentos anteriores (*aol, ldivw, len, grith, ldivd, A, area, B*).
En los siguientes experimentos se eliminan los buques de pasajeros, militares y recreativos del conjunto de datos						
29	100	SI	SI	NO	0,75/0,25	Cinco atributos más importantes, según experimentos anteriores (*aol, ldivw, wdivd, a ldivd*).

Tabla 7 Experimentos con datos estáticos

3.8.2 Experimentos con conjunto de datos dinámicos

En la Tabla 8 se muestran los experimentos realizados utilizando el conjunto de datos dinámicos extraídos de los mensajes AIS de tipos 1, 2 y 3.

Nº	Nº arb	MMS	OverS	UnderS	De/Dt	Atributos
30	100	SI	NO	NO	0,75/0,25	Todos los dinámicos.
31	200	SI	NO	NO	0,75/0,25	Todos los dinámicos.
32	50	SI	NO	NO	0,75/0,25	Todos los dinámicos.
33	100	SI	NO	NO	0,75/0,25	Siete atributos más importantes, según experimentos anteriores (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*).
34	100	SI	NO	NO	0,75/0,25	Atributo más importante, según experimentos anteriores (*avg_sog*).
35	100	SI	NO	NO	0,9/0,1	Todos los dinámicos.
36	100	SI	NO	NO	0,8/0,2	Todos los dinámicos.
37	100	SI	NO	NO	0,65/0,35	Todos los dinámicos.
38	100	NO	NO	NO	0,75/0,25	Todos los dinámicos.
39	100	SI	SI	NO	0,75/0,25	Todos los dinámicos.
40	100	SI	SI	NO	0,75/0,25	Siete atributos más importantes, según experimentos anteriores (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*).
41	100	SI	NO	SI	0,75/0,25	Todos los dinámicos.
42	100	SI	NO	SI	0,75/0,25	Siete atributos más importantes, según experimentos anteriores (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*).
43	100	SI	NO	NO	0,75/0,25	Todos los dinámicos y la eslora.
44	100	SI	NO	NO	0,75/0,25	Siete atributos más importantes, según experimentos anteriores (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) y la eslora.
45	100	SI	SI	NO	0,75/0,25	Todos los dinámicos y la eslora.
46	100	SI	SI	NO	0,75/0,25	Siete atributos más importantes, según experimentos anteriores (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) y la eslora.

47	100	SI	NO	SI	0,75/0,25	Todos los dinámicos y la eslora.
48	100	SI	NO	SI	0,75/0,25	Siete atributos más importantes, según experimentos anteriores (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) y la eslora.
En los siguientes experimentos se eliminan los buques militares y recreativos del conjunto de datos						
49	100	SI	NO	NO	0,75/0,25	Todos los dinámicos.
50	100	SI	SI	NO	0,75/0,25	Todos los dinámicos.
51	100	SI	NO	SI	0,75/0,25	Todos los dinámicos.

Tabla 8 Experimentos con datos dinámicos

3.8.3 Experimentos con conjunto de datos combinando datos estáticos y dinámicos

En la Tabla 9 se muestran los experimentos realizados utilizando el conjunto de datos resultado de combinar las dos bases de datos anteriores y extrayendo los barcos comunes de los que disponemos de todos los campos necesarios.

Nº	Nº arb	MMS	OverS	UnderS	De/Dt	Atributos
52	100	SI	NO	NO	0,75/0,25	Todos los dinámicos y estáticos.
53	100	SI	SI	NO	0,75/0,25	Todos los dinámicos y estáticos.
54	100	SI	NO	SI	0,75/0,25	Todos los dinámicos y estáticos.
55	100	SI	NO	NO	0,75/0,25	Ocho atributos estáticos (*aol*, *ldivw*, *len*, *grith*, *ldivd*, *A*, *area*, *B*) y siete atributos dinámicos (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) más importantes en sus respectivos experimentos.
56	100	SI	SI	NO	0,75/0,25	Ocho atributos estáticos (*aol*, *ldivw*, *len*, *grith*, *ldivd*, *A*, *area*, *B*) y siete atributos dinámicos (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) más importantes en sus respectivos experimentos.
57	100	SI	NO	SI	0,75/0,25	Ocho atributos estáticos (*aol*, *ldivw*, *len*, *grith*, *ldivd*, *A*, *area*, *B*) y siete atributos dinámicos (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) más importantes en sus respectivos experimentos.
58	100	SI	NO	NO	0,75/0,25	Todos los estáticos y siete atributos dinámicos (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) más importantes en sus respectivos experimentos.

59	100	SI	NO	NO	0,75/0,25	Todos los dinámicos y ocho atributos estáticos (*aol*, *ldivw*, *len*, *grith*, *ldivd*, *A*, *area*, *B*) más importantes en sus respectivos experimentos.
60	100	SI	NO	NO	0,75/0,25	Siete atributos dinámicos más importantes (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) y ocho atributos estáticos menos importantes (*C*, *D*, *vs*, *aml*, *wid*, *wdivd*, *amt*, *draught*) en sus respectivos experimentos.

Tabla 9 Experimentos con combinación de datos dinámicos y estáticos

4 RESULTADOS OBTENIDOS Y VALIDACIÓN

En el siguiente capítulo se exponen los resultados obtenidos a partir de los experimentos realizados, mencionando únicamente aquellos experimentos de cuyo resultado se han podido extraer conclusiones. Previamente al análisis de los resultados obtenidos se explicarán las métricas utilizadas para evaluar las diferentes variaciones del algoritmo, para a continuación analizar los experimentos distinguiéndolos según el conjunto de datos empleado. Finalmente, se valorarán los experimentos de forma global para determinar el modelo que mejor se ajusta al problema planteado.

Para agilizar la redacción de los resultados de los experimentos se hará referencia a los mismos siguiendo la numeración que se recoge en las tablas presentes en el capítulo anterior (Tabla 7, Tabla 8 y Tabla 9).

4.1 Métricas empleadas

Como ya se ha mencionado anteriormente, la librería de *Sklearn* nos proporciona la función *classification_report*, la cual nos aporta una serie de métricas, como los que se pueden ver en la Figura 3-12, que serán las que se utilicen para evaluar el resultado de cada uno de los experimentos. A continuación, se procede a citar y explicar las métricas que aparecen en los informes resultantes de cada experimento [94]:

- **Exactitud (*Accuracy*):** es la métrica que mide el porcentaje de casos que el modelo ha predicho correctamente. Se calcula como la división del número de predicciones realizadas correctamente entre un número total de predicciones realizadas. Es uno de los coeficientes más utilizados por su facilidad de comprensión, sin embargo, hay que tener especial atención pues puede resultar engañoso. Por ejemplo, si quisiéramos clasificar manzanas y plátanos y nuestro conjunto de datos estuviese compuesto por noventa manzanas y diez plátanos, un modelo que dijera que todos los datos son manzanas tendría una exactitud del 90%, lo cual podría parecer como algo bueno, pero no lo es pues nuestro modelo es incapaz de detectar lo que es un plátano. Es por ello por lo que no es óptimo tener en cuenta este parámetro para conjuntos de datos con clases desbalanceadas.
- **Precisión (*Precision*)**: es la métrica que mide el porcentaje de predicciones correctas realizadas para cada una de las clases. Para una clase determinada se calcula como el cociente entre las predicciones correctas realizadas de esa clase entre las predicciones totales realizadas de esa clase. Por ejemplo, si el modelo del ejemplo anterior clasificara diez elementos como manzanas, pero de esas diez únicamente ocho lo son realmente, la precisión para esa clase seria del 80%.
- **Exhausividad (*Recall*)**: es la métrica que mide el porcentaje de elementos de una clase determinada que ha sido capaz de clasificar correctamente el modelo. Se calcula como el

cociente entre los datos clasificados correctamente de una clase y los datos totales que existen de esa clase. En el ejemplo anterior si en los datos existen diez manzanas en total, pero de esas diez solo se clasifican siete cómo tales estamos diciendo que el *recall* del modelo para esa clase es 70%.

- **Valor-F1 (*F1-Score*)**: es la medida que combina el *recall* y la precisión en un solo valor. Se utiliza para los casos en los que nos interesa tener en cuenta ambos porque considerar solo uno de los dos valores puede llevar a confusión a la hora de interpretar resultados. Por ejemplo, si nuestro modelo trabaja con un conjunto de cien datos que tiene manzanas y plátanos a partes iguales, pero solo clasifica tres como manzanas y estas lo son realmente, la precisión es del 100% aunque no sea capaz de detectar prácticamente las manzanas. Por el contrario, si el modelo dice que existen 100 manzanas, el modelo tendrá un *recall* del 100%, pero se habrá equivocado a la hora de clasificar todos los plátanos. Es por este motivo por el que el *F1-Score* es tan útil, porque tiene ambos en consideración al mismo tiempo evitando casos como el del ejemplo expuesto. Se calcula como la media armónica del *recall* y la precisión.

Además, el informe generado proporciona una media aritmética de estas métricas para todas las clases disponibles en el algoritmo de clasificación y una media ponderada en función de la cantidad de datos que exista de cada clase.

En este estudio, al disponer de un conjunto de datos en los que las clases están muy desbalanceadas, las métricas que tendremos más en consideración serán la precisión, el *recall* y el *F1-Score*. El parámetro *accuracy* y la media ponderada de los parámetros pasarán a un segundo plano debido, precisamente, a esta disparidad de tamaño de clases.

4.2 Experimentos con conjunto de datos estáticos

A continuación, se muestran los experimentos realizados con el fin de hallar cuál es el modelo que se ajusta mejor a la predicción del tipo de buque nutriéndose a partir de datos estáticos procedentes de los mensajes AIS del tipo 5.

En primer lugar, se ha realizado un estudio para determinar la cantidad de árboles a utilizar que resultará más apropiada para el modelo. Para ello se han realizado los experimentos 1, 2 y 3 obteniéndose los resultados que se aprecian en la Figura 4-1.

Se puede observar que cuándo usamos 50 árboles (Experimento-2) se pierde algo de precisión y *recall* que, si observamos detenidamente, afecta únicamente a las clases minoritarias. Por otro lado, si comparamos el empleo de 100 árboles (Experimento-1) con el de 200 árboles (Experimento-3) los resultados son muy similares en cuanto al *F1-Score* y las discrepancias producidas son a causa de la aleatoriedad del algoritmo. Por lo tanto, el número de árboles empleado será 100, puesto que para los 200 árboles el tiempo de procesado es mayor y no se consiguen ventajas sustanciales. Los experimentos 4, 5 y 6 realizan una comparación de árboles utilizando diferentes atributos, al igual que los experimentos 7, 9 y 10 y se llega a las mismas conclusiones.

Experimento-1

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.95	8930
Fishing	0.79	0.80	0.79	1186
MilitaryOps	0.82	0.36	0.50	90
Passenger	0.71	0.71	0.71	587
PleasureCraft	0.63	0.55	0.59	282
Tanker	0.92	0.86	0.89	3268
Tug	0.82	0.82	0.82	1107
accuracy			0.90	15450
macro avg	0.80	0.72	0.75	15450
weighted avg	0.90	0.90	0.90	15450

Experimento-2

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.95	8930
Fishing	0.78	0.80	0.79	1186
MilitaryOps	0.79	0.33	0.47	90
Passenger	0.70	0.71	0.70	587
PleasureCraft	0.61	0.53	0.57	282
Tanker	0.92	0.86	0.89	3268
Tug	0.82	0.81	0.82	1107
accuracy			0.90	15450
macro avg	0.79	0.72	0.74	15450
weighted avg	0.90	0.90	0.89	15450

Experimento-3

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.95	8930
Fishing	0.79	0.80	0.80	1186
MilitaryOps	0.81	0.33	0.47	90
Passenger	0.70	0.72	0.71	587
PleasureCraft	0.63	0.55	0.59	282
Tanker	0.92	0.87	0.89	3268
Tug	0.82	0.82	0.82	1107
accuracy			0.90	15450
macro avg	0.80	0.72	0.75	15450
weighted avg	0.90	0.90	0.90	15450

Figura 4-1 Comparativa del número de árboles (datos estáticos)

A continuación, se analiza el cambio de los resultados al variar el reparto de los datos en datos de entrenamiento y los de test para observar si hay alguna distribución que favorezca al modelo. Los experimentos realizados para comprobar esta hipótesis son los experimentos 11, 12 y 13, obteniéndose los resultados de la Figura 4-2.

Experimento-11

	precision	recall	f1-score	support
Cargo	0.94	0.96	0.95	3654
Fishing	0.79	0.79	0.79	459
MilitaryOps	0.78	0.33	0.46	43
Passenger	0.70	0.75	0.73	230
PleasureCraft	0.52	0.50	0.51	104
Tanker	0.91	0.87	0.89	1245
Tug	0.83	0.86	0.85	445
accuracy			0.90	6180
macro avg	0.78	0.72	0.74	6180
weighted avg	0.90	0.90	0.90	6180

Experimento-12

	precision	recall	f1-score	support
Cargo	0.94	0.96	0.95	7197
Fishing	0.79	0.79	0.79	949
MilitaryOps	0.70	0.31	0.43	75
Passenger	0.71	0.72	0.72	457
PleasureCraft	0.60	0.58	0.59	225
Tanker	0.92	0.87	0.90	2569
Tug	0.83	0.83	0.83	888
accuracy			0.90	12360
macro avg	0.78	0.72	0.74	12360
weighted avg	0.90	0.90	0.90	12360

Experimento-13

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.94	12467
Fishing	0.77	0.80	0.78	1664
MilitaryOps	0.79	0.30	0.44	126
Passenger	0.69	0.69	0.69	841
PleasureCraft	0.63	0.54	0.58	436
Tanker	0.91	0.87	0.89	4551
Tug	0.82	0.80	0.81	1545
accuracy			0.89	21630
macro avg	0.79	0.71	0.73	21630
weighted avg	0.89	0.89	0.89	21630

Figura 4-2 Comparativa datos de entrenamiento/datos de test

Ante los resultados obtenidos en los experimentos, se puede observar que el tamaño de los datos de entrenamiento y de test no influye excesivamente en el modelo, siempre y cuando los datos de entrenamiento sean suficientes para entrenar el modelo. Se establece para el resto de los experimentos una proporción de datos entrenamiento/test igual a 0.75/0.25.

Una vez ya hemos establecido todos los parámetros que se pueden variar de los que proporciona *Random Forest*, procedemos a estudiar cómo se comportaría el modelo ante las variaciones en el conjunto de datos.

Hemos observado en todos los experimentos anteriores que la falta de datos de las clases minoritarias provoca que el valor de *F1-Score* sea menor que para el resto de clases. Para lidiar con esta situación se recurre a comprobar si la creación de datos ficticios hasta igualar la cantidad de datos de cada clase (*oversampling*) o la reducción de datos hasta igualar las clases (*undersampling*) resulta de utilidad a la hora de mejorar la calidad del modelo. En la Figura 4-3 comparamos experimentos de igual número de árboles y atributos, pero uno de ellos usando técnicas de *oversampling* (Experimento 16) y otro usando técnicas de *undersampling* (Experimento 18).

Experimento-1

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.95	8930
Fishing	0.79	0.80	0.79	1186
MilitaryOps	0.82	0.36	0.50	90
Passenger	0.71	0.71	0.71	587
PleasureCraft	0.63	0.55	0.59	282
Tanker	0.92	0.86	0.89	3268
Tug	0.82	0.82	0.82	1107
accuracy			0.90	15450
macro avg	0.80	0.72	0.75	15450
weighted avg	0.90	0.90	0.90	15450

Experimento-16

	precision	recall	f1-score	support
Cargo	0.95	0.93	0.94	8930
Fishing	0.78	0.78	0.78	1186
MilitaryOps	0.48	0.49	0.48	90
Passenger	0.62	0.75	0.68	587
PleasureCraft	0.55	0.56	0.56	282
Tanker	0.88	0.89	0.89	3268
Tug	0.81	0.81	0.81	1107
accuracy			0.89	15450
macro avg	0.73	0.75	0.74	15450
weighted avg	0.89	0.89	0.89	15450

Experimento-18

	precision	recall	f1-score	support
Cargo	0.91	0.76	0.83	8930
Fishing	0.76	0.68	0.72	1186
MilitaryOps	0.09	0.63	0.16	90
Passenger	0.42	0.69	0.52	587
PleasureCraft	0.35	0.70	0.46	282
Tanker	0.65	0.75	0.70	3268
Tug	0.78	0.75	0.77	1107
accuracy			0.75	15450
macro avg	0.57	0.71	0.59	15450
weighted avg	0.80	0.75	0.77	15450

Figura 4-3 Comparativa de uso de *undersampling* y *oversampling*

De esta comparativa se pueden sacar varias conclusiones. En primer lugar, vemos que el uso del *oversampling* (Experimento 16) empeora ligeramente el valor de *F1-Score* con respecto a no usarlo (Experimento 1). Sin embargo, si desglosamos *F1-Score* en precisión y *recall* podemos observar diferencias. El empleo del *oversampling* no afecta de forma significativa en las clases mayoritarias, pero las clases minoritarias se ven claramente afectadas a pesar de tener valores similares de *F1-Score* para ambos experimentos. Y es que se puede ver que en las clases minoritarias el valor del *recall* aumenta considerablemente y, por el contrario, el valor de la precisión disminuye también de forma sustancial. De esta forma se explica que el *F1-Score* tenga valores similares en ambos experimentos. Los resultados de esta prueba demuestran que el uso de *oversampling* ayuda a la identificación de buques de la clase minoritaria puesto que aumenta el *recall*, sin embargo, al disminuir la precisión quiere decir que realizará un mayor número de predicciones falsas de esta clase. Por lo tanto, el empleo de *oversampling* puede resultar interesante si priorizamos la detección de buques de las clases minoritarias frente al resto. No

hay que olvidar que los datos añadidos al conjunto de datos son datos ficticios pudiendo ser el resultado obtenido fruto de un sobreajuste a datos ficticios muy similares.

En el caso del uso de *undersampling* (Experimento 16) podemos observar que todos los valores empeoran excepto el *recall* de las clases minoritarias, lo cual se debe a la falta de datos para generalizar los tipos de buques. Por la reducida cantidad de datos resultante el uso de *undersampling* queda descartado.

Por último, se ha variado el número de atributos que se le pasan al algoritmo para observar si los atributos de menor importancia son más bien "productores de ruido" que atributos que aporten valor al algoritmo. Se ha representado la importancia de los atributos del Experimento-1, que coincide con la del resto de experimentos iniciales, obteniéndose la gráfica de la Figura 4-4.

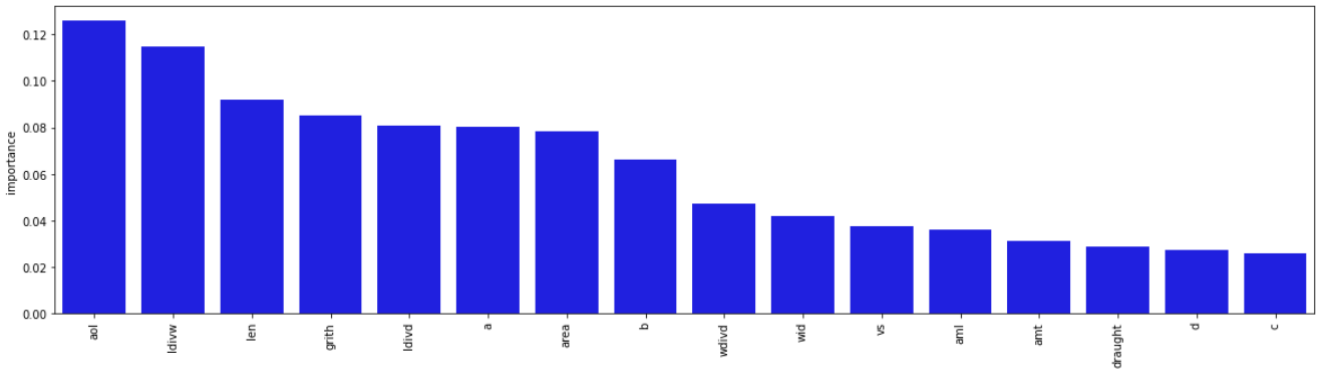

Figura 4-4 Representación de importancia de los atributos estáticos para el Experimento-1

De esta forma se ha construido un modelo con los ocho atributos más importantes (Experimento 7) y otro con los tres atributos más importantes (Experimento 8) cuyos resultados podemos ver en la Figura 4-5.

Experimento-1

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.95	8930
Fishing	0.79	0.80	0.79	1186
MilitaryOps	0.82	0.36	0.50	90
Passenger	0.71	0.71	0.71	587
PleasureCraft	0.63	0.55	0.59	282
Tanker	0.92	0.86	0.89	3268
Tug	0.82	0.82	0.82	1107
accuracy			0.90	15450
macro avg	0.80	0.72	0.75	15450
weighted avg	0.90	0.90	0.90	15450

Experimento-7

	precision	recall	f1-score	support
Cargo	0.94	0.95	0.94	8880
Fishing	0.76	0.79	0.78	1145
MilitaryOps	0.76	0.30	0.43	88
Passenger	0.72	0.69	0.70	641
PleasureCraft	0.61	0.51	0.56	341
Tanker	0.90	0.89	0.90	3295
Tug	0.79	0.81	0.80	1060
accuracy			0.89	15450
macro avg	0.78	0.70	0.73	15450
weighted avg	0.89	0.89	0.89	15450

Experimento-8

	precision	recall	f1-score	support
Cargo	0.89	0.94	0.91	8930
Fishing	0.75	0.76	0.75	1186
MilitaryOps	0.64	0.28	0.39	90
Passenger	0.64	0.58	0.61	587
PleasureCraft	0.49	0.39	0.44	282
Tanker	0.86	0.78	0.82	3268
Tug	0.73	0.76	0.75	1107
accuracy			0.85	15450
macro avg	0.72	0.64	0.67	15450
weighted avg	0.85	0.85	0.85	15450

Figura 4-5 Comparativa del número de atributos usados

Se puede apreciar a partir de la comparativa de los valores de *F1-Score* que la reducción del número de atributos a los ocho más importantes (Experimento 7) no degrada la calidad del modelo a excepción de las clases más minoritarias. Además, la reducción del número de atributos agiliza la fase de entrenamiento y clasificación de nuevos datos. Sin embargo, el empleo de únicamente los tres atributos más importantes (Experimento 8) degrada demasiado la efectividad del algoritmo como se puede ver en los valores de *F1-Score.*

Esta comparativa invita a considerar que, si se dispusiera de una base de datos menos desequilibrada, el empleo de la reducción de atributos sería muy útil. Para comprobar esta hipótesis, se ha hecho u*ndersampling* de los datos para tener un conjunto de datos reales de igual tamaño y comparar la reducción de atributos (Experimento 19). Los resultados son los de la Figura 4-6.

Experimento-18

	precision	recall	f1-score	support
Cargo	0.91	0.76	0.83	8930
Fishing	0.76	0.68	0.72	1186
MilitaryOps	0.09	0.63	0.16	90
Passenger	0.42	0.69	0.52	587
PleasureCraft	0.35	0.70	0.46	282
Tanker	0.65	0.75	0.70	3268
Tug	0.78	0.75	0.77	1107
accuracy			0.75	15450
macro avg	0.57	0.71	0.59	15450
weighted avg	0.80	0.75	0.77	15450

Experimento-19

	precision	recall	f1-score	support
Cargo	0.92	0.75	0.83	8930
Fishing	0.73	0.70	0.71	1186
MilitaryOps	0.10	0.53	0.17	90
Passenger	0.37	0.63	0.47	587
PleasureCraft	0.35	0.63	0.45	282
Tanker	0.65	0.79	0.71	3268
Tug	0.78	0.74	0.76	1107
accuracy			0.75	15450
macro avg	0.56	0.68	0.58	15450
weighted avg	0.80	0.75	0.77	15450

Figura 4-6 Comparativa del número de atributos usados después de hacer *undersampling* de los datos

Podemos observar en los valores de precisión y *recall* que, pese a la reducción del número de atributos, los resultados obtenidos son muy similares ante el equilibrio de las clases, apoyando así la hipótesis planteada.

Para finalizar con los datos estáticos, se realizaron experimentos eliminando las clases minoritarias del conjunto de los datos obteniéndose, como cabría esperar, una mejora de la media aritmética de los valores de *precisión*, *recall* y *F1-Score* puesto que se eliminan valores que bajaban la media. Sin embargo, estos valores para cada uno de los tipos de buque se mantienen igual que si estuvieran las clases minoritarias.

A modo de resumen, se estima que el mejor modelo es aquel que emplea 100 árboles y, ante la falta de datos de las clases minoritarias, el que emplea todos los atributos. Además, las técnicas de *oversampling* han demostrado ayudar a la detección de clases minoritarias. De esta forma se estima que, haciendo uso de datos dinámicos, el mejor modelo para tráfico general sería el del Experimento 1 y para la detección de los buques de las clases minoritarias el modelo del Experimento 16. Estos modelos tendrían un valor medio de *F1-Score* de 0.75 y 0.74 respectivamente.

4.3 Experimentos con conjuntos de datos dinámicos

A continuación, se muestran los experimentos realizados con el fin de hallar cuál es el modelo que se ajusta mejor a la predicción del tipo de buque nutriéndose de datos dinámicos procedentes de los mensajes AIS de tipos 1, 2 y 3.

La forma de proceder en estos experimentos es la misma que la que se ha seguido en los experimentos que utilizan el conjunto de datos estáticos. En primer lugar, determinaremos el número de árboles que se ajusta mejor al modelo haciendo experimentos con 50 (Experimento 32), 100 (Experimento 30) y 200 (Experimento 31) arboles, obteniéndose los resultados que podemos ver en la Figura 4-7.

Experimento-30

	precision	recall	f1-score	support
Cargo	0.63	0.92	0.75	3446
Passenger	0.85	0.49	0.62	443
Tanker	0.40	0.06	0.10	1536
fishing	0.70	0.57	0.63	579
military	0.00	0.00	0.00	31
pleasure craft	0.00	0.00	0.00	49
tug	0.64	0.68	0.66	738
accuracy			0.63	6822
macro avg	0.46	0.39	0.39	6822
weighted avg	0.59	0.63	0.57	6822

Experimento-31

	precision	recall	f1-score	support
Cargo	0.63	0.93	0.75	3446
Passenger	0.85	0.49	0.62	443
Tanker	0.40	0.05	0.09	1536
fishing	0.69	0.57	0.62	579
military	0.00	0.00	0.00	31
pleasure craft	0.00	0.00	0.00	49
tug	0.65	0.67	0.66	738
accuracy			0.63	6822
macro avg	0.46	0.39	0.39	6822
weighted avg	0.59	0.63	0.56	6822

Experimento-32

	precision	recall	f1-score	support
Cargo	0.62	0.91	0.74	3446
Passenger	0.82	0.49	0.61	443
Tanker	0.39	0.08	0.13	1536
fishing	0.71	0.56	0.63	579
military	0.00	0.00	0.00	31
pleasure craft	0.00	0.00	0.00	49
tug	0.64	0.65	0.65	738
accuracy			0.63	6822
macro avg	0.46	0.38	0.39	6822
weighted avg	0.59	0.63	0.57	6822

Figura 4-7 Comparativa del número de árboles utilizados (datos dinámicos)

Los resultados obtenidos son muy similares, aunque se aprecia una pequeña pérdida en los valores promedios de las características para 50 árboles (Experimento 32) por lo que se establecen para el resto de los experimentos 100 árboles (Experimento 30) que es el número de árboles mínimo a partir del cual el modelo mantiene el valor de las métricas.

Proseguimos, al igual que en los experimentos con datos estáticos, con la observación del modelo ante el cambio de la cantidad de los datos de entrenamiento con los experimentos que se muestran en la Figura 4-8.

Experimento-35

	precision	recall	f1-score	support
Cargo	0.64	0.94	0.76	1418
Passenger	0.88	0.51	0.65	180
Tanker	0.40	0.06	0.10	603
fishing	0.74	0.57	0.65	220
military	0.00	0.00	0.00	12
pleasure craft	0.00	0.00	0.00	18
tug	0.65	0.68	0.66	278
accuracy			0.65	2729
macro avg	0.47	0.39	0.40	2729
weighted avg	0.60	0.65	0.58	2729

Experimento-36

	precision	recall	f1-score	support
Cargo	0.64	0.91	0.75	2789
Passenger	0.84	0.49	0.62	367
Tanker	0.40	0.07	0.12	1185
fishing	0.70	0.60	0.65	467
military	0.00	0.00	0.00	25
pleasure craft	0.00	0.00	0.00	37
tug	0.63	0.68	0.65	588
accuracy			0.64	5458
macro avg	0.46	0.39	0.40	5458
weighted avg	0.60	0.64	0.58	5458

Experimento-37

	precision	recall	f1-score	support
Cargo	0.63	0.86	0.72	4978
Passenger	0.72	0.44	0.55	603
Tanker	0.37	0.12	0.18	2190
fishing	0.63	0.55	0.59	846
military	0.00	0.00	0.00	42
pleasure craft	0.00	0.00	0.00	62
tug	0.64	0.66	0.65	1096
accuracy			0.61	9817
macro avg	0.43	0.38	0.38	9817
weighted avg	0.57	0.61	0.56	9817

Figura 4-8 Comparativa datos de entrenamiento/datos de test

A diferencia de los experimentos con el conjunto de datos estáticos, con los datos dinámicos sí se aprecia diferencia entre las diferentes proporciones de los datos de entrenamiento/test. Se observa que conforme se disminuyen los datos de entrenamiento y aumentan los de test, los valores de precisión, *recall* y *F1-Score* van disminuyendo. Esto se debe a que los modelos con datos dinámicos, como podemos observar, tienden a fallar más las clasificaciones por lo que cuanto más grande sea el conjunto de test más fallará. Además, cuantos menos datos de entrenamiento, más le costará al algoritmo encontrar patrones en los datos. Hay que encontrar un equilibrio entre entrenar lo suficiente el algoritmo y que los datos de test sean una muestra representativa del conjunto de datos.

Una vez seleccionados los parámetros que se consideran óptimos para el algoritmo *Random Forest*, pasamos a estudiar la mejora del modelo a partir de la variación del conjunto de datos que se le pasa al mismo.

En la primera de las aproximaciones modificando el conjunto de los datos, se valora el uso de las técnicas de *oversampling* y *undersampling*. Para ello se compara el empleo de estas técnicas para modelos iguales en los que solo varía el empleo de técnicas de *oversampling* (Experimento 39) y técnicas de *undersampling* (Experimento 41), obteniéndose los resultados que se aprecian en la Figura 4-9.

Experimento-30

	precision	recall	f1-score	support
Cargo	0.63	0.92	0.75	3446
Passenger	0.85	0.49	0.62	443
Tanker	0.40	0.06	0.10	1536
fishing	0.70	0.57	0.63	579
military	0.00	0.00	0.00	31
pleasure craft	0.00	0.00	0.00	49
tug	0.64	0.68	0.66	738
accuracy			0.63	6822
macro avg	0.46	0.39	0.39	6822
weighted avg	0.59	0.63	0.57	6822

Experimento-39

	precision	recall	f1-score	support
Cargo	0.69	0.65	0.67	3446
Passenger	0.58	0.63	0.61	443
Tanker	0.36	0.32	0.34	1536
fishing	0.55	0.67	0.61	579
military	0.14	0.03	0.05	31
pleasure craft	0.12	0.12	0.12	49
tug	0.56	0.74	0.64	738
accuracy			0.58	6822
macro avg	0.43	0.45	0.43	6822
weighted avg	0.57	0.58	0.57	6822

Experimento-41

	precision	recall	f1-score	support
Cargo	0.67	0.40	0.50	3446
Passenger	0.41	0.60	0.49	443
Tanker	0.30	0.32	0.31	1536
fishing	0.47	0.62	0.54	579
military	0.02	0.32	0.05	31
pleasure craft	0.03	0.31	0.06	49
tug	0.51	0.58	0.54	738
accuracy			0.43	6822
macro avg	0.34	0.45	0.35	6822
weighted avg	0.53	0.43	0.46	6822

Figura 4-9 Comparación del uso de técnicas de *undersampling* y *oversampling* sobre los datos dinámicos

En primer lugar, se puede observar que el empleo de técnicas de *undersampling* (Experimento 41) no resulta rentable. Aunque el *recall* de las clases minoritarias aumente debido a un mayor equilibrio entre la cantidad de datos de cada clase, la falta de datos debido a la eliminación de estos hace el modelo más deficiente como se puede apreciar en los valores de *F1-Score* medios cuando usamos *undersampling* (0.39) y cuando no lo hacemos (0.35).

En cuanto el empleo del *oversampling* (Experimento 39), podemos ver como en general disminuye los valores de la precisión, pero aumenta los valores del *recall*. Para saber si compensa esta pérdida de precisión con respecto a la ganancia de *recall*, se usará el valor de *F1-Score* que tiene ambas en consideración por igual, pues nos interesa que nuestro modelo tenga el mayor valor posible de ambas. Podemos ver que el valor promedio sin *oversampling* (Experimento 30) es del 0.39 y que con *oversampling* (Experimento 39) es del 0.43. El *oversampling* hace que las clases minoritarias pasen de ser indetectables a que tengan presencia en las predicciones del modelo. Por lo tanto, en lo referente al uso de datos dinámicos, se estima beneficioso el empleo de técnicas de *oversampling*.

La última de las mejores propuestas consiste en reducir el número de atributos que el modelo utiliza para clasificar los datos. Para ello se ha representado la importancia de los atributos en los experimentos en los que no se modifican los datos (Experimento 30). Si se seleccionara en base a la importancia de los datos en el experimento que se ha utilizado *oversampling* (Experimento 39) podría obtenerse una visión deformada de la realidad, pues gran parte de los datos son datos ficticios, muy similares a los que realmente existen. La importancia de los atributos obtenida viene representada en la Figura 4-10.

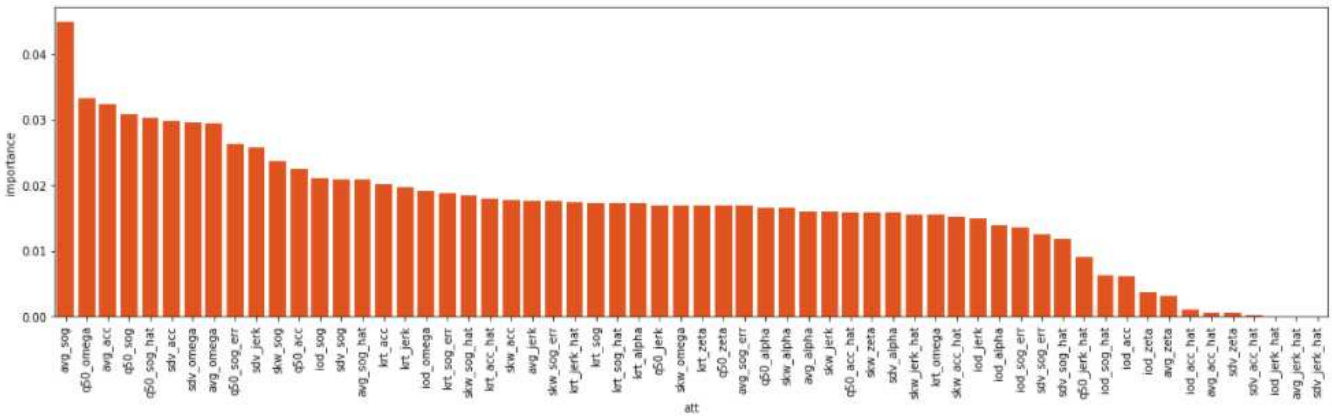

Figura 4-10 Representación de importancia de los atributos estáticos para el Experimento-30

Para poder llevar a cabo la prueba se han tomado los siete atributos más importantes (*avg_sog, avg_omega, avg_acc, sdv_acc, q50_omega, q50_sog_hat, q50_sog*) y se ha recreado un experimento similar al que por el momento nos aporta mejores resultados, con la diferencia de que se utilizan únicamente los siete atributos más importantes según la Figura 4-10. En la Figura 4-11 se pueden observar los resultados obtenidos del experimento recortando el número de atributos (Experimento 40) y manteniendo todos los atributos (Experimento 39).

Experimento-39

	precision	recall	f1-score	support
Cargo	0.69	0.65	0.67	3446
Passenger	0.58	0.63	0.61	443
Tanker	0.36	0.32	0.34	1536
fishing	0.55	0.67	0.61	579
military	0.14	0.03	0.05	31
pleasure craft	0.12	0.12	0.12	49
tug	0.56	0.74	0.64	738
accuracy			0.58	6822
macro avg	0.43	0.45	0.43	6822
weighted avg	0.57	0.58	0.57	6822

Experimento-40

	precision	recall	f1-score	support
Cargo	0.67	0.62	0.64	3446
Passenger	0.56	0.57	0.56	443
Tanker	0.35	0.31	0.33	1536
fishing	0.46	0.59	0.52	579
military	0.02	0.03	0.02	31
pleasure craft	0.04	0.08	0.06	49
tug	0.53	0.66	0.59	738
accuracy			0.54	6822
macro avg	0.38	0.41	0.39	6822
weighted avg	0.55	0.54	0.54	6822

Figura 4-11 Comparación de resultados para diferente número de atributos

Podemos ver que todas las métricas empeoran ante la reducción del número de atributos, por lo que mantendremos todos los atributos en el modelo.

Además de los experimentos expuestos, se han desarrollado otros experimentos combinando las variables que se han ido describiendo a lo largo del apartado sin mejorar las propiedades del modelo.

También se ha probado a eliminar las clases minoritarias para observar si así el algoritmo se ajusta mejor al resto de clases. Como es obvio el valor promedio de la precisión, *recall* y *F1-Score* mejora debido a que se eliminan los valores que bajan el valor de la media. Pero si nos fijamos en cada una de las clases, la precisión y el *recall* se mantienen con respecto a los experimentos anteriores, pese a que se eliminen las clases minoritarias.

Para finalizar con este conjunto de datos, se ha desarrollado la idea de observar qué sucedería al incorporar un atributo estático al modelo. Esta acción será muy simple pues, como ya se explicó en el capítulo anterior, el conjunto de los datos de los que partimos para la creación de atributos dinámicos posee también la eslora del buque. Los resultados obtenidos de incluir la eslora como un atributo más junto con los atributos dinámicos, teniendo en consideración todos los atributos y sin modificar el *dataset* original, son los que se pueden ver en la Figura 4-12.

Experimento-43

	precision	recall	f1-score	support
Cargo	0.67	0.95	0.78	3422
Passenger	0.80	0.52	0.63	392
Tanker	0.70	0.16	0.27	1579
fishing	0.77	0.72	0.74	603
military	0.00	0.00	0.00	30
pleasure craft	0.00	0.00	0.00	56
tug	0.76	0.81	0.78	765
accuracy			0.69	6847
macro avg	0.53	0.45	0.46	6847
weighted avg	0.69	0.69	0.64	6847

Figura 4-12 Resultados del experimento 43

Se puede observar que los resultados obtenidos mejoran comparados con cualquiera de los experimentos que usan únicamente datos dinámicos, llegando a conseguir mejorar este algoritmo mediante técnicas de *oversampling* y reducción del número de atributos un valor de *F1_Score* promedio igual a 0.49 en los experimentos 44 y 45. Además, si representamos la importancia de los atributos en la Figura 4-13 podemos ver como un simple atributo estático cobra mayor importancia que cualquiera de los atributos dinámicos.

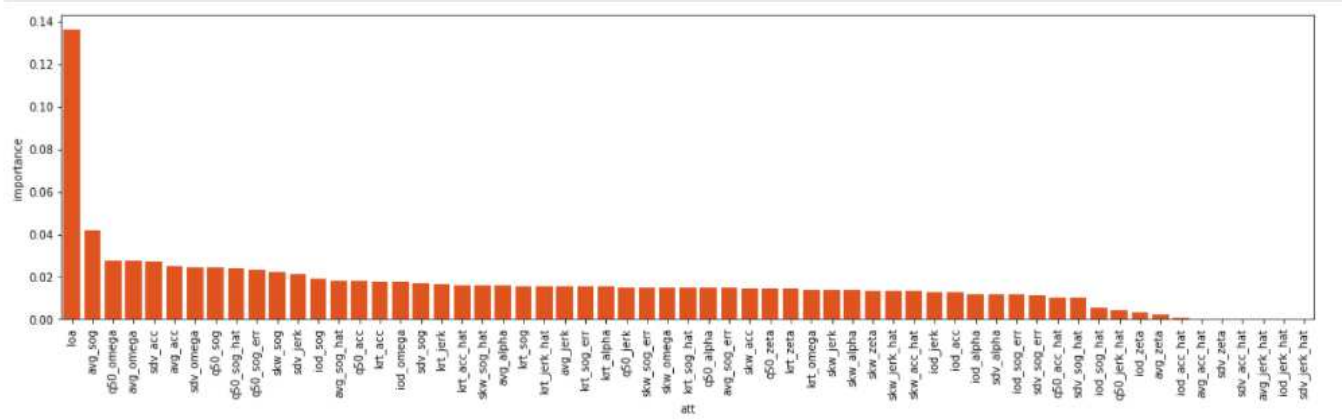

Figura 4-13 Importancia de los atributos empleados en el experimento 43

Los resultados de los experimentos de este apartado nos muestran que el mejor modelo para los datos dinámicos sería el que resulta de emplear 100 árboles, *oversampling* y todos los atributos disponibles (Experimento 39), consiguiéndose un valor de F1-Score igual a 0.43.

Por otro lado, los experimentos realizados al final del apartado nos invitan a analizar qué sucedería en caso de combinar ambos conjuntos de datos. En el siguiente apartado se procede a estudiar esta situación.

4.4 Experimentos con conjunto de datos combinando datos estáticos y dinámicos

Una vez estudiados los mejores modelos para cada uno de los conjuntos de datos se pasa a estudiar qué sucedería si tratásemos de mejorar el modelo incorporando datos de diferentes tipos de mensajes AIS en un mismo modelo.

Puesto que los experimentos anteriores realizados con ambas bases de datos llegan a las mismas conclusiones con respecto al número de árboles y la proporción de datos de entrenamiento/test, se procederá directamente a estudiar el modelo ante la variación de los datos y los atributos con los que trabaja el algoritmo.

Sin entrar en detalle, y como era de esperar por los resultados de los experimentos anteriores, el *undersampling* nos ofrece una solución que empeora francamente el modelo, mientras que el *oversampling* nos propone situaciones más interesantes que trataremos a continuación. En la Figura 4-14 se muestran tres experimentos con todos los atributos dinámicos y estáticos usando el *dataset* original (Experimento 52) usando *oversampling* (Experimento 53) y usando *undersampling* (Experimento 54).

Experimento-52

	precision	recall	f1-score	support
Cargo	0.96	0.98	0.97	4984
Passenger	0.90	0.85	0.88	230
Tanker	0.96	0.92	0.94	1916
fishing	0.86	0.90	0.88	376
military	0.55	0.38	0.44	16
pleasure craft	0.80	0.38	0.52	21
tug	0.91	0.91	0.91	476
accuracy			0.95	8019
macro avg	0.85	0.76	0.79	8019
weighted avg	0.95	0.95	0.95	8019

Experimento-53

	precision	recall	f1-score	support
Cargo	0.97	0.96	0.97	4984
Passenger	0.81	0.92	0.86	230
Tanker	0.93	0.93	0.93	1916
fishing	0.83	0.93	0.87	376
military	0.46	0.38	0.41	16
pleasure craft	0.45	0.43	0.44	21
tug	0.91	0.92	0.92	476
accuracy			0.95	8019
macro avg	0.77	0.78	0.77	8019
weighted avg	0.95	0.95	0.95	8019

Experimento-54

	precision	recall	f1-score	support
Cargo	0.88	0.63	0.73	4984
Passenger	0.27	0.71	0.39	230
Tanker	0.54	0.72	0.62	1916
fishing	0.72	0.82	0.77	376
military	0.05	0.56	0.10	16
pleasure craft	0.06	0.86	0.12	21
tug	0.85	0.83	0.84	476
accuracy			0.67	8019
macro avg	0.48	0.73	0.51	8019
weighted avg	0.77	0.67	0.70	8019

Figura 4-14 Comparación del uso de técnicas de *undersampling* y *oversampling* para datos combinados

El problema de los experimentos de la Figura 4-14 es que el tiempo de entrenamiento del algoritmo es considerablemente mayor debido a la gran cantidad de atributos que se manejan. Es por ello por lo que se ha decidido repetir estos experimentos (a excepción del que hace uso de *undersampling*) recortando el número de atributos a los más importantes de cada conjunto de datos (estáticos y dinámicos) según los apartados anteriores, obteniéndose el resultado de la Figura 4-15.

Experimento-55

	precision	recall	f1-score	support
Cargo	0.96	0.98	0.97	4986
Passenger	0.91	0.81	0.86	275
Tanker	0.95	0.94	0.95	1856
fishing	0.83	0.86	0.84	336
military	0.57	0.27	0.36	15
pleasure craft	0.86	0.40	0.55	30
tug	0.88	0.88	0.88	419
accuracy			0.95	7917
macro avg	0.85	0.73	0.77	7917
weighted avg	0.95	0.95	0.95	7917

Experimento-56

	precision	recall	f1-score	support
Cargo	0.98	0.96	0.97	4986
Passenger	0.83	0.88	0.86	275
Tanker	0.93	0.95	0.94	1856
fishing	0.81	0.88	0.85	336
military	0.50	0.33	0.40	15
pleasure craft	0.58	0.50	0.54	30
tug	0.89	0.90	0.89	419
accuracy			0.95	7917
macro avg	0.79	0.77	0.78	7917
weighted avg	0.95	0.95	0.95	7917

Figura 4-15 Resultados obtenidos de reducir los atributos en el *dataset* combinado

Como se puede observar en los valores de *F1-Score*, el rendimiento de los modelos ante la reducción de atributos prácticamente se mantiene, por lo que la reducción de atributos es una solución válida para el problema del tiempo de entrenamiento.

Por otro lado, aunque se obtengan valores de *F1-Score* muy similares en todos los experimentos que usan *oversampling* y los que no, podemos observar que los valores de precisión y *recall* son muy

dispares, siendo mejores los modelos que utilizan *oversampling* en cuanto al *recall* pero los que no utilizan esta técnica son mejores en cuanto a la precisión.

Esto quiere decir que el algoritmo que emplee *oversampling* será capaz de detectar en mayor medida los diferentes tipos de buque, pero clasificará erróneamente una mayor cantidad de datos. En caso de no usar *oversampling* tendríamos la situación opuesta. Debido a que en nuestro problema no existen clases más importantes que otras porque las clases que no resultaban de interés ya se han eliminado, frente a iguales valores de *F1-Score* nos interesa primar la precisión frente al *recall*. Nos interesa saber con seguridad que un buque se trata realmente, por ejemplo, de un pesquero, más que detectar todos los pesqueros a cambio de decir que todos los barcos son pesqueros y así detectarlos a todos. Además, la mejora del *recall* usando *oversampling* analizada para cada tipo de buque no es tan significativa.

Por lo tanto, en concordancia con lo expuesto en el apartado, se considera que el modelo más acertado para el empleo de datos combinados sería el resultante del Experimento 55, usando únicamente los mejores atributos de cada tipo y sin usar técnicas de *oversampling.*

Para finalizar el apartado se quiere hacer referencia a la importancia de los atributos estáticos frente a los dinámicos. En la Figura 4-16 se puede observar la importancia de los atributos en la ejecución del Experimento 52. En ella se puede ver que los atributos dinámicos pasan prácticamente a un segundo plano frente a los atributos estáticos, siendo todos los atributos estáticos, excepto *C* y *D*, más importantes que todos los atributos dinámicos.

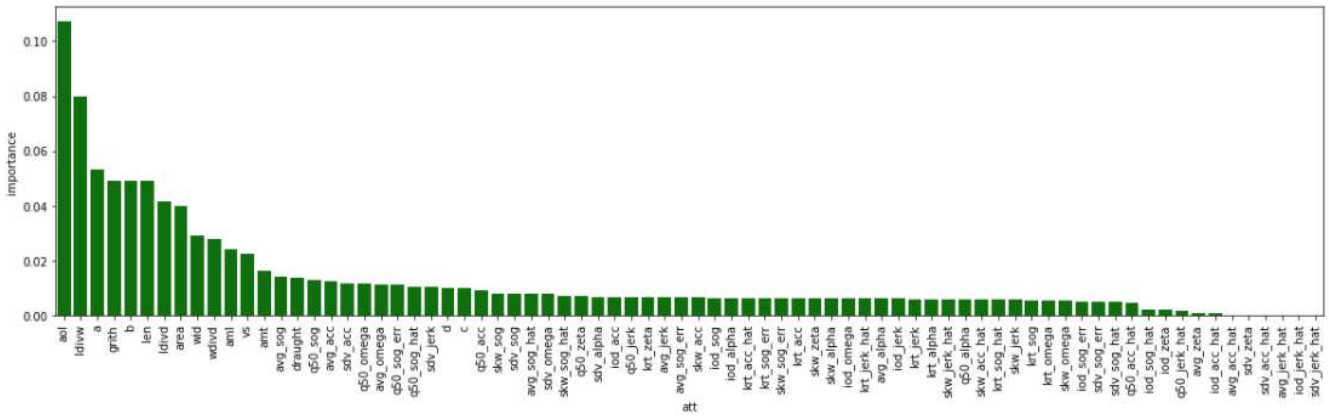

Figura 4-16 Importancia de los atributos en el Experimento 52

Este comportamiento se mantiene, aunque seleccionemos solo los atributos dinámicos y estáticos más importantes, como se puede ver en la Figura 4-17, donde todos los atributos estáticos tienen mayor importancia que todos los atributos dinámicos.

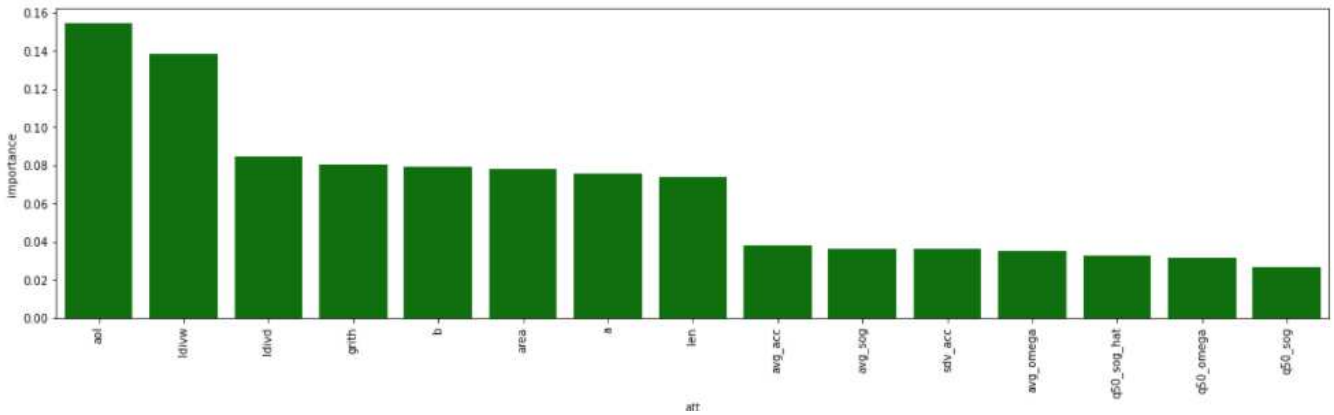

Figura 4-17 Importancia de los atributos en el Experimento 55

De hecho, para demostrar de forma más exagerada la importancia que da el algoritmo a los atributos estáticos frente a los dinámicos, se ha desarrollado un experimento en el que usamos todos los atributos dinámicos y los atributos estáticos de menor importancia, obteniéndose los resultados de la Figura 4-18. En estos se puede ver que incluso los atributos estáticos más irrelevantes adquieren mayor importancia que prácticamente cualquiera de los datos dinámicos.

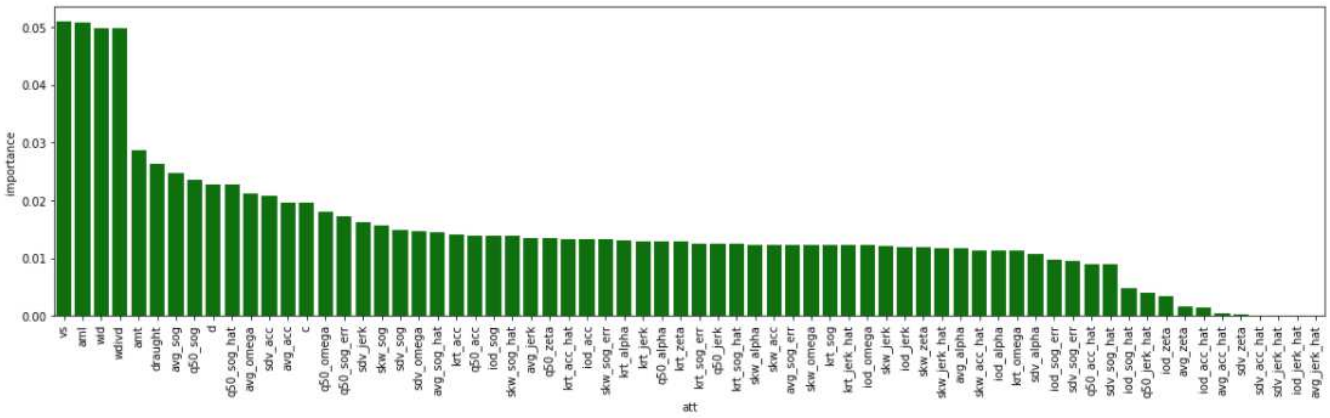

Figura 4-18 Importancia de los atributos empleados en el Experimento 60

Sin embargo, ya se ha podido ver que el empleo de atributos dinámicos mejora la precisión, *recall* y el *F1-Score* del modelo, por lo que su uso se estima apropiado.

4.5 Evaluación global de los resultados

Una vez que se han comentado los experimentos que nos han servido para obtener conclusiones de cuál es el modelo que mejor se ajusta a la solución del problema planteado, pasamos a la selección del modelo y su justificación.

Se ha podido observar que el modelo donde las métricas utilizadas son más favorables es en la combinación de datos estáticos y dinámicos. Los atributos dinámicos resultan muy pobres para la clasificación del tipo de buque en comparación con los estáticos. Los resultados obtenidos usando atributos dinámicos por si solos no son capaces de obtener los resultados esperados. Por el contrario, los atributos estáticos son capaces de generar modelos que podrían cumplir esta tarea relativamente con éxito, y se ven mejorados con la adición de atributos dinámicos, por lo que los atributos dinámicos no resultan inútiles.

En cuanto al uso de técnicas para equilibrar la disparidad de clases de los datos se ha podido apreciar que el uso del *undersampling* elimina demasiados datos por lo que no es una técnica útil. Sin embargo, el *oversampling* es una práctica que, por lo general, mejora la detección de las clases minoritarias a cambio de que el modelo tenga una menor precisión.

Finalmente se ha llegado a la conclusión que el modelo que mejor se ajusta a la clasificación del tipo de buque es el modelo resultante del Experimento 55, del cual recordamos sus características en la Tabla 10.

Nº	Nº arb	MMS	OverS	UnderS	De/Dt	Atributos
55	100	SI	NO	NO	0,75/0,25	Ocho atributos estáticos (*aol*, *ldivw*, *len*, *grith*, *ldivd*, *A*, *area*, *B*) y siete atributos dinámicos (*avg_sog*, *avg_omega*, *avg_acc*, *sdv_acc*, *q50_omega*, *q50_sog_hat*, *q50_sog*) más importantes en sus respectivos experimentos.

Tabla 10 Características del modelo seleccionado

Además de ser el modelo que mejor se ajusta a todas las clases, se trata de un modelo que gana mucho en cuanto a los tiempos de ejecución gracias a la reducción de atributos a los más importantes de cada tipo. Las métricas obtenidas de este modelo son las que se pueden ver en la Figura 4-19, habiendo obtenido una precisión promedio de 0.85, un *recall* promedio de 0.73, y un *F1-Score* promedio de 0.77.

Experimento-55

	precision	recall	f1-score	support
Cargo	0.96	0.98	0.97	4986
Passenger	0.91	0.81	0.86	275
Tanker	0.95	0.94	0.95	1856
fishing	0.83	0.86	0.84	336
military	0.57	0.27	0.36	15
pleasure craft	0.86	0.40	0.55	30
tug	0.88	0.88	0.88	419
accuracy			0.95	7917
macro avg	0.85	0.73	0.77	7917
weighted avg	0.95	0.95	0.95	7917

Figura 4-19 Resultados del Experimento 55

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Los avances tecnológicos en la sociedad actual están forzando a los diferentes países y organizaciones a la actualización de las infraestructuras y sistemas de los que disponen. Es tal la importancia de los avances tecnológicos emergentes, que están marcando una nueva revolución industrial, cambiando la forma de operar de las empresas y de las instituciones de todo el mundo. La inteligencia artificial se postula como una de estas nuevas tecnologías, siendo un campo de estudio en el que se han logrado avances que hasta hace pocos años serían inimaginables.

La principal conclusión que se extrae de este trabajo es que el uso de inteligencia artificial en tareas que para una persona podrían llegar a ser tediosas o incluso inalcanzables empieza a ser más sencillo. La tendencia actual está desembocando en una acumulación de datos masiva (*Big Data*) siendo imposible su procesamiento por personas. Las técnicas de *Machine Learning*, no solo nos permiten que una máquina haga este trabajo por nosotros, sino que son capaces de encontrar patrones que un humano difícilmente podría encontrar.

En cuanto a los objetivos establecidos en el capítulo 1 de la memoria, el principal objetivo de la creación de un modelo que sea capaz de predecir el tipo de buque de un contacto valiéndose únicamente de la información que este transmite por AIS ha sido cumplido satisfactoriamente. Se ha conseguido encontrar un algoritmo que se ajuste a las necesidades y adecuar el conjunto de los datos para que el modelo pueda ser entrenado satisfactoriamente.

El tratamiento previo de los datos ha sido fundamental. La adecuación del conjunto de datos utilizados seleccionando solo aquellos que sean fiables para el entrenamiento del algoritmo ha jugado un papel fundamental a la hora de alcanzar los resultados propuestos. Además, la creación de atributos a partir de los parámetros disponibles en los mensajes AIS ha jugado un papel crucial para que el algoritmo lograra discernir entre los diferentes tipos de buque.

Como segundo objetivo, se proponía un estudio de la validez de los datos estáticos y dinámicos para discernir entre diferentes tipos de buques. A la vista de los resultados, los datos estáticos son claramente representativos del tipo de buque, mientras que la cinemática no resulta tan significativa. Sin embargo, la adición de cinemática a las características estáticas de los buques logra mejorar los resultados obteniendo que el modelo distinga mejor las diferentes clases, por lo que queda demostrada la utilidad de ambas clases de datos.

También se ha apreciado cómo un conjunto de datos desbalanceado hace que el modelo se ajuste más a las clases mayoritarias en detrimento de la calidad de la predicción generada por las clases

minoritarias. Sin embargo, se ha demostrado la eficacia de las técnicas propuestas para solucionar este problema.

Los principales problemas a la hora de desarrollar el trabajo han sido el gran volumen de datos con el que se trabaja, haciendo necesario un filtrado y procesado de los datos para adaptar los datos para que puedan ser usados por el algoritmo y obtener los resultados óptimos. Además, ha sido necesario lidiar con el problema de los datos desbalanceados.

Finalmente, a título personal me gustaría concluir indicando que el desarrollo de este trabajo me ha ayudado a introducirme dentro de lo que sería este nuevo campo de la inteligencia artificial que hoy en día sigue creciendo y mejorándose. Me ha ayudado a comprender el funcionamiento de muchos de los sistemas que nos rodean y de los que hacemos uso diariamente que utilizan este tipo de técnicas de aprendizaje automático. Por último y seguramente lo más importante, me ha hecho darme cuenta de la importancia de la introducción de la inteligencia artificial en la Armada, al ser una tecnología tan relativamente joven y la cantidad de posibilidades que se nos presentan haciendo uso de ella.

5.2 Lineas futuras

Una vez se ha conseguido crear varios modelos capaces de clasificar los diferentes tipos de buque en función de los datos AIS, se proponen las siguientes líneas futuras que permiten a continuar con el desarrollo de este trabajo o aplicar mejoras sobre el mismo:

- Estudio de nuevos posibles atributos que sirvan para alimentar el algoritmo y que no hayan sido empleados en este trabajo.
- Creación de una muestra de datos de clases equilibradas adquiriendo datos de franjas temporales más amplias o diferentes zonas para solventar el problema del desequilibrio de clases sin la necesidad de la creación de tramas sintéticas o la eliminación de información de las clases mayoritarias.
- Estudiar la posibilidad de emplear los datos procedentes de los mensajes AIS no utilizados, como por ejemplo la zona en la que se encuentra un buque, para llevar a cabo la predicción del tipo de buque.
- Valorar el empleo de otro tipo de algoritmos de aprendizaje supervisado para alcanzar los objetivos propuestos en este trabajo. Por ejemplo, puede ser interesante el uso del algoritmo *Gradient Boosting* por su similitud con *Random Forest* y así poder realizar una comparación con los resultados obtenidos en este trabajo. También se plantea la posibilidad de aplicar técnicas de aprendizaje semisupervisado e incluso aprendizaje no supervisado.

Al margen de la temática abierta en este TFG, se podría proponer enfocar la clasificación de contactos mediante uso de técnicas de inteligencia artificial a otros campos de aplicación de origen militar como, por ejemplo, la clasificación de trazas como hostiles, neutrales, amigas… en las consolas tácticas del sistema de combate de los buques en función de los criterios establecidos para llevar a cabo dicha clasificación.

6 BIBLIOGRAFÍA

[1] Almirante Jefe del Estado Mayor de la Armada (AJEMA), «Lineas generales de la Armada 2022,» Ministerio de Defensa, 2022.

[2] Defensa.com, «Indra y la Armada Española utilizan la Inteligencia Artificial para mejorar el mantenimiento de los buques de nueva generación,» 2020. [En línea]. Available: https://www.defensa.com/espana/indra-armada-espanola-utilizan-inteligencia-artificial-para. [Último acceso: 08/02/2022].

[3] Centro Universitario de la Defensa en la Escuela Naval Militar (CUD-ENM) «Proyectos/contratos,» [En línea]. Available: https://cud.uvigo.es/proyectos-contratos/. [Último acceso: 12/03/2022].

[4] OMI, «Convenio internacional para la seguridad de la vida humana en el mar, 1974 (Convenio SOLAS),» [En línea]. Available: https://www.imo.org/es/About/Conventions/Paginas/International-Convention-for-the-Safety-of-Life-at-Sea-(SOLAS),-1974.aspx. [Último acceso: 07/02/2022].

[5] B. G. Torres, «El verdadero padre de la inteligencia artificial. Ventana al conocimiento. Periodismo científico.,» 2016. [En línea]. Available: https://www.bbvaopenmind.com/tecnologia/inteligencia-artificial/el-verdadero-padre-de-la-inteligencia-artificial/. [Último acceso: 21/01/2022].

[6] «¿Qué es la inteligencia artificial y cómo se usa?,» Parlamento Europeo Noticias, 8/9/2020. [En línea]. Available: https://www.europarl.europa.eu/news/es/headlines/society/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa#:~:text=Definici%C3%B3n%20de%20inteligencia%20artificial,y%20la%20capacidad%20de%20planear.. [Último acceso: 20/01/2022].

[7] S. Russell y P. Norvig, Artificial Intelligence: A Modern Approach, Emglewood Cliffs, New Jersey: Prentice Hall, Inc., 1995.

[8] T. Rodríguez, «*Machine Learning* y *Deep Learning*: cómo entender las claves del presente y futuro de la inteligencia artificial,» Xataka, 2017. [En línea]. Available:

https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial. [Último acceso: 23/01/2022].

[9] G. V. Guillén, «Aristóteles y la automatización de la lógica. Una lectura desde la inteligencia artificial,» *Estudios de Filosofía*, nº 23, pp. 25-42, 2001-01-05.

[10] R. Gestión, «¿Qué es la inteligencia artificial y para qué sirve?,» 2018. [En línea]. Available: https://gestion.pe/tecnologia/inteligencia-artificial-historia-origen-funciona-aplicaciones-categorias-tipos-riesgos-nnda-nnlt-249002-noticia/. [Último acceso: 20/01/2022].

[11] M. Delgado, «La Inteligencia Artificial. Realidad de un mito moderno.,» de *Discurso de apertura Universidad de Granada Curso Académico 1996-1997*, Granada, 1996.

[12] L. E. Munuera, «Inteligencia Artificial y sistemas expertos,» *ICESI*, 1991.

[13] A. M. Turing, «*Computing Machinery and Intelligenc*e,» *Mind*, 1950.

[14] G. G. Martínez, «El experimento de la habitación china: ¿ordenadores con mente?,» Psicología y Mente, 2018. [En línea]. Available: https://psicologiaymente.com/psicologia/experimento-de-la-habitacion-china. [Último acceso: 20/01/2022].

[15] N. G. España, «Breve historia visual de la inteligencia artificial,» NG España, 2019. [En línea]. Available: https://www.nationalgeographic.com.es/ciencia/breve-historia-visual-inteligencia-artificial_14419. [Último acceso: 20/01/2022].

[16] D. G. Bobrow, «*Natural Language Input for a Computer Problem Solving System*,» MIT libraries, 1964. [En línea]. Available: https://dspace.mit.edu/handle/1721.1/6903. [Último acceso: 01/20/2022].

[17] B. N. Mundo, «La sorprendente y poco conocida historia de Eliza, el primer bot conversacional de la historia,» BBC, 2018. [En línea]. Available: https://www.bbc.com/mundo/noticias-44290222. [Último acceso: 21/01/2022].

[18] E. Pérez, «Qué es el efecto ELIZA, o por qué nos sorprende tanto leer un artículo "escrito" por una inteligencia artificial como GPT-3,» 2020. [En línea]. Available: https://www.xataka.com/otros/que-efecto-eliza-que-nos-sorprende-leer-articulo-escrito-inteligencia-artificial-como-gpt-3. [Último acceso: 21/01/2022].

[19] M. J. Díaz, «Inteligencia Artificial y datos para la sociedad,» de *Lección Inaugural Curso 2021-2022*, Universidad de Jaen, 2021.

[20] A. F. Candial, «Deep Blue-Kaspárov: cuando la máquina venció al hombre,» La vanguardia, 2021. [En línea]. Available: https://www.lavanguardia.com/vida/junior-report/20210210/6234712/kasparov-deep-blue-maquina-vencio-hombre.html. [Último acceso: 22/01/2022].

[21] J. M. Nieves, «Un ordenador pasa por primera vez el test de Turing y convence a los jueces de que es humano,» ABC, 2014. [En línea]. Available: bc.es/ciencia/20140609/abci-superordenador-supera-primera-test-201406091139.html?ref=https%3A%2F%2Fwww.abc.es%2Fciencia%2F20140609%2Fabci-superordenador-supera-primera-test-201406091139.html. [Último acceso: 21/01/2022].

[22] B. Edwards, «*Vintage Computing and Gaming*,» 2006. [En línea]. Available: https://www.vintagecomputing.com/index.php/archives/127/retro-scan-of-the-week-10-megabyte-hard-disk-3495. [Último acceso: 22/01/2022].

[23] J. M. Norman, «HistoryofInformation.com,» 2020. [En línea]. Available: https://www.historyofinformation.com/detail.php?id=740. [Último acceso: 22/01/2022].

[24] S. V. Bourdié, «Aplicaciones de la inteligencia artificial en la empresa» 2019. [En línea]. Available: https://repositorio.unican.es/xmlui/bitstream/handle/10902/17521/Valverdebourdiesandra.pdf?sequence=1&isAllowed=y. [Último acceso: 22/01/2022].

[25] R. M. Peña, «La inteligencia artificial en la actualidad,» Escuela especializada en ingeniería ITCA-FEPADE, 2011.

[26] M. Saltzman, «AARP,» 2019. [En línea]. Available: https://www.aarp.org/espanol/hogar-familia/tecnologia/info-2019/consejos-para-usar-siri-de-apple.html. [Último acceso: 22/01/2022].

[27] Iberdrola, «Descubre los principales beneficios del '*Machine Learning*',» 2022. [En línea]. Available: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico. [Último acceso: 23/01/2022].

[28] IBM Cloud Education, «*Machine Learning*,» IBM, 2020. [En línea]. Available: https://www.ibm.com/es-es/cloud/learn/machine-learning#toc-cmo-funcio-2KNEOt4z. [Último acceso: 28/01/2022].

[29] TIBCO, «*What is Supervised Learning*?,» TIBCO, 2022. [En línea]. Available: https://www.tibco.com/es/reference-center/what-is-supervised-learning. [Último acceso: 26/01/2022].

[30] P. J. González Negro, «Sistema de localización de personal a bordo basado en técnicas, de aprendizaje automático y Bluetooth Low Energy» 2018. [En línea]. Available: http://calderon.cud.uvigo.es/handle/123456789/229. [Último acceso: 26/01/2022].

[31] G. Romero Jiménez, «El CAPTCHA y tu contribución (anónima) en el entrenamiento sistemas de *machine learning*» LegalToday, 2021. [En línea]. Available: https://www.legaltoday.com/legaltech/novedades-legaltech/el-captcha-y-tu-contribucion-anonima-en-el-entrenamiento-sistemas-de-machine-learning-2021-02-11/. [Último acceso: 25/01/2022].

[32] «Protege tu sitio del spam con reCAPTCHA,» ELSATE.com, 2019. [En línea]. Available: https://www.elsate.com/viewtopic.php?t=2462. [Último acceso: 26/01/2022].

[33] IONOS , «*Unsupervised learning*: aprendizaje automático sin restricciones,» IONOS Digital Guide, 2020. [En línea]. Available: https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/unsupervised-learning/. [Último acceso: 26/01/2022].

[34] J. Luna González, «Tipos de aprendizaje automático,» Soldai, 2018. [En línea]. Available: https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2. [Último acceso: 27/01/2022].

[35] D. Calvo, «Aprendizaje no supervisado,» 2019. [En línea]. Available: https://www.diegocalvo.es/aprendizaje-no-supervisado/. [Último acceso: 27/01/2022].

[36] Centum, «Aprendizaje no supervisado para la detección de anomalías,» CENTUM Solutions, S.L, [En línea]. Available: https://centum.com/aprendizaje-no-supervisado-para-la-deteccion-de-anomalias/. [Último acceso: 27/01/2022].

[37] M. F. Abdel Hady y F. Schwenker, «*Semi-supervised Learning*,» de *Handbook on Neural Information Processing*, Springer, 2013, pp. 215-239.

[38] Predictiva, «Aprendizaje Semi-Supervisado,» [En línea]. Available: https://www.predictiva.com.co/blog-predictiva/aprendizaje-semi-supervisado/. [Último acceso: 28/01/2022].

[39] R. S. Sutton y A. G. Barto, *Reinforcement Learning*, second edition: *An Introduction*, *Cambridge*: The MIT Press, 2018.

[40] B. Osiński y K. Budek, «*What is reinforcement learning? The complete guide*,» Deepsense.ai, 2018. [En línea]. Available: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/. [Último acceso: 30/01/2022].

[41] D. Silver y D. Hassabis, «*AlphaGo Zero: Starting from scratch*,» Deeomind, 2017. [En línea]. Available: https://deepmind.com/blog/article/alphago-zero-starting-scratch. [Último acceso: 30/01/2022].

[42] F. Marín Bellón, «Cómo funcionan Alpha Go, AlphaZero y DeepMind, dioses de la inteligencia artificial,» ABC, 2018. [En línea]. Available: https://abcblogs.abc.es/jugar-con-cabeza/otros-temas/alpha-go-alphazero-deepmind-inteligencia-artificial.html. [Último acceso: 30/01/2022].

[43] Iberdrola, «'*Deep learning*': un concepto clave para llevar la inteligencia artificial al siguiente nivel,» [En línea]. Available: https://www.iberdrola.com/innovacion/deep-learning. [Último acceso: 30/01/2022].

[44] N. Developer, «*Deep learning*» NVIDIA, [En línea]. Available: https://developer.nvidia.com/deep-learning. [Último acceso: 31/01/2022].

[45] I. C. Education, «*What is deep learning?*,» IBM, 2020. [En línea]. Available: https://www.ibm.com/cloud/learn/deep-learning. [Último acceso: 30/01/2022].

[46] P. Larrañaga, I. Inza y A. Moujahid, «Redes Neuronales,» Departamento de ciencias de la computación e Inteligencia Artificial, Universidad del Pais Vasco, 2007.

[47] Atria innovation, «Qué son las redes neuronales y sus funciones,» 2019. [En línea]. Available: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/. [Último acceso: 30/01/2022].

[48] J. . M. Cárdenas y D. Hinestroza Ramírez, «El *machine learning* a traves de los tiempos, y los aportes a la humanidad ,» 2018. [En línea]. Available: https://repository.unilibre.edu.co/bitstream/handle/10901/17289/EL%20MACHINE%20LEARNING.pdf?sequence=1&isAllowed=y. [Último acceso: 01/02/2022].

[49] TIBCO, «*What is Machine Learning?*,» [En línea]. Available: https://www.tibco.com/reference-center/what-is-machine-learning. [Último acceso: 01/02/2022].

[50] S. Asenjo, «Google Lens ya está disponible en forma de app independiente para Android,» 2018. [En línea]. Available: https://wwwhatsnew.com/2018/06/06/google-lens-ya-esta-disponible-en-forma-de-app-independiente-para-android/. [Último acceso: 01/02/2022].

[51] K. Schwab, T*he Fourth Industrial Revolution*, Penguin Books, 2017.

[52] G. Gastelu, «*Get in line Tesla Bot, other automakers have done the robot thing*,» FOX Business, 2020. [En línea]. Available: https://www.foxbusiness.com/lifestyle/teslabot-automakers-robot-thing. [Último acceso: 01/02/2022].

[53] A. Navada, A. N. Ansari, S. Patil y B. A. Sonkamble, «*Overview of use of decision tree algorithms in machine learning*,» de *2011 IEEE Control and System Graduate Research Colloquium*, IEEE, 2011.

[54] O. Mbaabu, «*Introduction to Random Forest in Machine Learning*,» Section, 2020. [En línea]. Available: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/. [Último acceso: 03/02/2022].

[55] J. Ali, R. Khan, N. Ahmad y I. Maqsood, «*Random Forests and Decision Trees*,» de *IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3*, IJCSI, 2012, pp. 272-278.

[56] T. Yiu, «*Understanding Random Forest*,» Towards Data Science, 2019. [En línea]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Último acceso: 03/02/2022].

[57] A. Vidhya, «*Understanding Random Forest*,» 2021. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/. [Último acceso: 03/02/2022].

[58] A. Natekin y A. Knoll, «*Gradient boosting machines, a tutorial*,» Frontiers, 2013. [En línea]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full. [Último acceso: 06/02/2022].

[59] S. Gawali, «*Linear Regression in machine learning*,» Analytics Vidhya, 2021. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/06/linear-regression-in-machine-learning/. [Último acceso: 03/02/2022].

[60] Microsoft, «*Two-Class Logistic Regression component*,» 2021. [En línea]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/two-class-logistic-regression. [Último acceso: 03/02/2022].

[61] D. Rodríguez, «La regresión logística,» Analytics Lane, 2018. [En línea]. Available: https://www.analyticslane.com/2018/07/23/la-regresion-logistica/. [Último acceso: 03/02/2022].

[62] S. Ray, «*6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R*,» Analytics Vidhya, 2017. [En línea]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Último acceso: 04/02/2022].

[63] S. Ray, «*Understanding Support Vector Machine* (SVM) *algorithm from examples (along with code)*,» Analytics Vidhya, 2017. [En línea]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. [Último acceso: 04/02/2022].

[64] J. Martinez Heras, «Máquinas de Vectores de Soporte (SVM),» IArtificial.net, 2019. [En línea]. Available: https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/. [Último acceso: 23/02/2022].

[65] Merkle, «El algoritmo K-NN y su importancia en el modelado de datos,» 2020. [En línea]. Available: https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos. [Último acceso: 05/02/2022].

[66] S. Patwardhan, «*Simple understanding and implementation of KNN algorithm!*,» Analytics Vidhya, 2021. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/. [Último acceso: 05/02/2022].

[67] D. Pascual, F. Pla y S. Sánchez, «Algoritmos de agrupamiento,» Universidad de Oriente, Santiago de Cuba, 2007.

[68] Universidad de Granada, «Métodos Jerárquicos de Análisis,» 2012. [En línea]. Available: https://www.ugr.es/~gallardo/pdf/cluster-3.pdf. [Último acceso: 09/02/2022].

[69] Xlstat, «DBSCAN (Agrupamiento espacial basado en densidad de aplicaciones con ruido),» Addinsoft, [En línea]. Available: https://www.xlstat.com/es/soluciones/funciones/dbscan-density-based-spatial-clustering-of-applications-with-noise. [Último acceso: 10/02/2022].

[70] Z. Jaadi, «*A Step-by-Step Explanation of Principal Component Analysis (PCA),*» Built in, 2021. [En línea]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Último acceso: 11/02/2022].

[71] S. Rawat, «*Introduction to Independent Component Analysis in Machine Learning,*» Analytics step, 2021. [En línea]. Available: https://www.analyticssteps.com/blogs/introduction-independent-component-analysis-machine-learning. [Último acceso: 17/01/2022].

[72] A. Española, «ENCOMAR,» Ministerio de Defensa, [En línea]. Available: https://encomar.covam.es/. [Último acceso: 06/02/2022].

[73] A.-L. S. Lapinski, «*A Strategy for Uncertainty Visualization Design*.,» 2009. [En línea]. Available: http://www.sci.utah.edu/~kpotter/Library/Papers/lapinski:2009:SUVD/index.html. [Último acceso: 08/02/2022].

[74] M. de la Gándara García, «El COVAM de la Armada al servicio de la comunidad,» *Instituto Español de Estudios Estadistico*, nº 55, 2011.

[75] C. Systems, «*What is AIS?,*» [En línea]. Available: https://comarsystems.com/what-is-ais/. [Último acceso: 07/02/2022].

[76] «Sistema AIS en barcos: Tipos, ventajas y funcionamiento,» Boletin Patron, [En línea]. Available: https://www.boletinpatron.com/el-sistema-de-identificacion-automatica-de-buques-ais-y-la-pirateria-un-arma-de-doble-filo/#%C2%BFQu%C3%A9%20es%20el%20sistema%20AIS?. [Último acceso: 07/02/2022].

[77] S. Bhattacharjee, «*Automatic Identification System (AIS): Integrating and Identifying Marine Communication Channels,*» Marine Insight, 2021. [En línea]. Available: https://www.marineinsight.com/marine-navigation/automatic-identification-system-ais-integrating-and-identifying-marine-communication-channels/. [Último acceso: 07/02/2022].

[78] B. Carrasco, «Defensa modernizará el centro de operaciones y vigilancia de la Armada,» InfoDefensa, 2019. [En línea]. Available: https://www.infodefensa.com/texto-diario/mostrar/3129114/defensa-modernizara-centro-operaciones-vigilancia-armada. [Último acceso: 08/02/2022].

[79] L. Álamo, «La Armada usará inteligencia artificial para vigilar el entorno marítimo desde Cartagena,» La Opinión de Murcia, 2021. [En línea]. Available: https://www.laopiniondemurcia.es/cartagena/2021/11/10/armada-usara-inteligencia-artificial-vigilar-59387366.html. [Último acceso: 08/02/2022].

[80] Y. Zhou, W. Daamen, T. Vellinga y . S. P. Hoogendoorn, «*Ship classification based on ship behavior clustering from AIS data*,» Delft University of Technology, 2019.

[81] Y. Wang, L. Yang, X. Song y X. Li, «*Ship classification based on random forest using static information from AIS data*,» College of Aerospace Science and Engineering, National University of Defense, 2021.

[82] S. Matteucci, «*Maritime traffic characterization with the Automated Identification System*,» 2009. [En línea]. Available: https://www.cmre.nato.int/research/publications/technical-reports/formal-reports/340-maritime-traffic-characterization-with-the-automated-identification-system. [Último acceso: 17/02/2022].

[83] Jupyter, [En línea]. Available: https://jupyter.org/. [Último acceso: 14/03/2022].

[84] Python, [En línea]. Available: https://www.python.org/. [Último acceso: 14/03/2022].

[85] «Jupyter.org,» Jupyter, [En línea]. Available: https://jupyter.org/. [Último acceso: 27/02/2022].

[86] IONOS, «Jupyter Notebook: documentos web para análisis de datos, código en vivo y mucho más,» Digital Guide IONOS, 2019. [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/#:~:text=Jupyter%20Notebook%20proporciona%20un%20entorno,c%C3%A1lculos%20y%20ver%20los%20resultados.. [Último acceso: 27/02/2022].

[87] S. Universidades, «Python: qué es y por qué deberías aprender a utilizarlo,» Santander, 2021. [En línea]. Available: https://www.becas-santander.com/es/blog/python-que-es.html. [Último acceso: 27/02/2022].

[88] SQLite, [En línea]. Available: https://www.sqlite.org/index.html. [Último acceso: 14/03/2022].

[89] Microsoft, «Access SQL: conceptos básicos, vocabulario y sintaxis,» [En línea]. Available: https://support.microsoft.com/es-es/office/access-sql-conceptos-b%C3%A1sicos-vocabulario-y-sintaxis-444d0303-cde1-424e-9a74-e8dc3e460671. [Último acceso: 14/03/2022].

[90] *United States Coast Guards Navigation Center*, «*AIS class a ship static and voyage related data (message 5),*» U.S Department of Homeland Security, 2017. [En línea]. Available: https://www.navcen.uscg.gov/?pageName=AISMessagesAStatic. [Último acceso: 04/03/2022].

[91] K. Pykes, «*Oversampling and Undersampling*,» Towards Data Science, 2020. [En línea]. Available: https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf. [Último acceso: 06/03/2022].

[92] J. Hale, «*Scale, Standardize, or Normalize with Scikit-Learn*,» Towards Data Science, 2019. [En línea]. Available: https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02. [Último acceso: 07/03/2022].

[93] S.-l. developers, «*Compare the effect of different scalers on data with outliers*,» Scikit-learn, 2021. [En línea]. Available: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html. [Último acceso: 07/03/2022].

[94] J. M. Heras, «*Precision, Recall, F1, Accuracy* en clasificación,» IArtificial.net, 2020. [En línea]. Available: https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/. [Último acceso: 10/03/2022].

[95] V. Alonso Aller, «Análisis de los sistemas de indexado geoespacial para el Conocimiento del Entorno Marítimo,» 2021. [En línea]. Available: http://calderon.cud.uvigo.es/xmlui/bitstream/handle/123456789/402/Alonso%20Aller%2C%20

V%C3%ADctor%20-%20Memoria.pdf?sequence=1&isAllowed=y. [Último acceso: 08/02/2022].

ANEXO I: DICCIONARIO DE SIGLAS, ACRÓNIMOS Y ABREVIATURAS

-AIS: *Automatic Identification System* (Sistema de Identificación Automática)

-AJEMA: Almirante Jefe del Estado Mayor de la Armada

-ALMART: Almirante de Acción Marítima

-*Bagging: Bootstrap aggregating*

-CAPTCHA: *Completely Automated Public Turing test to tell Computers and Humans Apart*

-CEM: Conocimiento del Entorno Marítimo

-CMOM: Comandante del Mando Operativo Marítimo

-CMRE: *Center of Maritime Research & Experimentation*

-COG: *Course Over Ground* (rumbo sobre el fondo)

-COVAM: Centro de Operaciones y Vigilancia de Acción Marítima

-CSV: *Comma Separated Values*

-DGAM: Dirección General de Armamento y Material

-ECM: Error Cuadrático Medio

-FAM: Fuerza de Acción Marítima

-GPS: *Global Positioning System*, Sistema de Geoposicionamiento Global

-IA: Inteligencia artificial

-ICA: *Independent Component Analysis*, o análisis de componentes independientes

-IMO: International Maritime Organization (Organización Marítima Internacional (OMI))

-Iot: *Internet of Things*

-KNN: *K-Nearest Neighbor*

-MIT: *Massachusetts Institute of Technology* (Instituto de Tecnología de Massachusetts)

-MMSI: *Marine Mobile Service Identity*

-MSA: *Maritime Situational Awareness*

-RMP: *Recognized Maritime Picture*

-SOG: *Speed Over Ground* (velocidad sobre el fondo)

-SQL: *Structured Query Language* (Lenguaje de consulta estructurado)

-SVM: *Support Vector Machine* (Máquinas de Vector Soporte)

-ZEE: Zona Económica Exclusiva

ANEXO II: CÓDIGOS NUMÉRICOS IDENTIFICADORES DEL TIPO DE BUQUE EN LOS DATOS AIS

Código	Tipo de buque	Código	Tipo de buque
20-29	*Wing in ground (WIG)*	52	*Tug*
30	*Fishing*	53	*Port Tender*
31	*Towing*	54	*Anti-pollution equipment*
32	*Towing: length exceeds 200m or breadth exceeds 25m*	55	*Law Enforcement*
33	*Dredging or underwater ops*	56	*Spare - Local Vessel*
34	*Diving ops*	57	*Spare - Local Vessel*
35	*Military ops*	58	*Medical Transport*
36	*Sailing*	59	*Noncombatant ship according to RR Resolution No. 18*
37	*Pleasure Craft*	60-69	*Passenger*
40-49	*High speed craft (HSC)*	70-79	*Cargo*
50	*Pilot Vessel*	80-89	*Tanker*
51	*Search and Rescue vessel*	90-99	*Other Type*

Tabla 11 Código asociado a cada tipo de buque en los mensajes AIS [95]

ANEXO III: CAMPOS DEL ARCHIVO CSV DE DATOS DINÁMICOS ORIGINAL

Campo	Descripción	Ejemplo
MMSI	El número de identificación del servicio móvil marítimo es una serie de 9 dígitos que identifica inequívocamente a cada estación transmisora.	368189880
IMO	Número de 7 dígitos asignados para cada buque de forma única, con el propósito de identificarlos y auxiliar en la consecución de la seguridad marítima.	2346531
MSG Type	Tipo de mensaje AIS	18
Timestamp	Es la hora de recepción del mensaje AIS por parte del sistema receptor redactada en formato AIS.	1621335430000
Latitude	Latitud geográfica.	27.643230000000003
Longitude	Longitud geográfica.	-80.37971999999999
H3	Celda hexagonal en formato H3 en la que se encuentra el contacto.	8944a856917ffff
Macrozones	Designación de la macrozona en la que se encuentra el contacto.	OTHER
Microzones	Designación de la microzona en la que se encuentra el contacto.	OTHER
COG	Rumbo del contacto.	350
SOG	Velocidad del contacto.	5.2
LOA	Eslora del contacto.	16.0
Name	Nombre del buque que transmite el mensaje AIS.	STAR
Shiptype AIS Raw	Tipo de buque según el mensaje AIS.	37
Shiptype AIS	Tipo de buque según el mensaje AIS.	37
Shiptype shipdata	Tipo de buque según la base de datos comercial *shipdata*.	37
Fishing	Responde si el contacto se trata de un pesquero.	NO
Cargo	Responde si el contacto se trata de un carguero.	NO
Leisure	Responde si el contacto se trata de una embarcación de recreo.	SI

Military	Responde si el contacto se trata de un buque militar.	NO
Stopped	Responde si el contacto está parado.	NO
Shipstatcode	Código que indica que acción (estado) está llevando a cabo el buque.	0 (Navegando con propulsión mecánica)
Destination	Puerto de destino.	CARTAGENA
Operator	Operador del buque.	MAERSK
Flag	Bandera del buque.	SPAIN
Flag Change	Responde a si se le ha visto con otra bandera	NaN
Year	Año del buque.	2005
Overcable/ Duct	Responde si el contacto se encuentra sobre cables o conductos submarinos.	NO
Within territory	Responde si el contacto se encuentra dentro del territorio nacional.	NO
Is spanish	Responde si el contacto es de nacionalidad española.	YES
Reapeated MMSI	Responde si el MMSI del contacto está duplicado.	NO

Tabla 12 Estructura de la base de datos inicial

ANEXO IV: CÓDIGO DEL ARCHIVO *PRETTYSTATIC.IPYNB*

```
#importar librerías

import pandas as pd
import sqlite3
import numpy as np
from tqdm import tqdm
import seaborn as sns
import matplotlib.pyplot as plt

#librerias de Machine Learning

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler

#Función para el cálculo de atributos estáticos

def statics(df):
    df['len']=df.a+df.b# Eslora
    df['wid']=df.c+df.d# Manga
    df['ldivw']= df.len/df.wid# Cociente entre eslora y manga
    df['ldivd']= df.len/df.draught# Cociente entre eslora y calado
    df['wdivd']= df.wid/df.draught# Cociente entre manga y calado
    df['area']= df.len*df.wid# Area de la cubierta aproximada= eslora x manga
    df['grith']= df.len+df.wid# Suma de eslora y manga
    df['aml']=df.len*df.draught# area longitudinal sumergida
    df['amt']=df.wid*df.draught# area transversal sumergida
    df['vs']=df.len*df.draught*df.wid # volumen sumergido
    df['aol']=df.a/df.len# proporción que supone a sobre la eslora total
    return df

#procesado del archivo CSV y creación de atributos

cols=['MMSI','to_bow','to_stern','to_port','to_starboard','draught','shiptype']
ncol=['mmsi','a','b','c','d','draught','shiptype']

# Troceado y procesado del CSV

data=pd.read_csv('../Gonzalo/static_info.csv', sep=',',chunksize=200000)
for df in data:
    chunk=df[cols]
    chunk.columns=ncol
    atributos=statics(chunk) #Cálculo de los atributos

#eliminación de atributos no válidos

atributos=atributos[atributos.replace([np.inf, -np.inf], np.nan).notnull().all
(axis=1)]
atributos=atributos.dropna()
```

```
df = atributos[atributos.shiptype.isin(['Cargo','Tanker','Fishing','Tug',
'Passenger','PleasureCraft','MilitaryOps'])] #seleccion de tipos de buque de
interés

#visualización de datos disponibles

gr = df.shiptype.value_counts().reset_index()
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='b')


#sql

#creación de la conexión

path_data = '../Gonzalo/DataStatic.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()

#almacenamiento de los datos en sql

df.to_sql('aistatic', con=conn, if_exists='append', index=False)

#selección de datos únicos para evitar duplicados

df = pd.read_sql("""SELECT DISTINCT mmsi FROM static""", con=conn)
for m in tqdm(df.mmsi):
    dft = pd.read_sql(f"""SELECT mmsi,a,b,c,d,draught,shiptype,len,wid,ldivw,
ldivd,wdivd,area,grith,aml,amt,vs,aol FROM static WHERE mmsi = {m} """, con=conn)
    dft.to_sql('best_static', con=conn, if_exists='append', index=False)

#Asignación de variables dependientes e independientes

df = pd.read_sql("""SELECT * FROM best_static""", con=conn)
df=df.drop('mmsi',axis=1) #eliminación de datos que no se usan como atributo
X=df.drop('shiptype', axis=1) #Variables independientes
y=df.shiptype #Variable dependiente


#Machine Learning

#División de los datos de partida en datos de entrenamiento y datos de test

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42)

#Undersampling

under = RandomUnderSampler()
X_res, Y_res=under.fit_resample(X_train, y_train)

#Oversampling

oversample=SMOTE(random_state = 42)
X_res, Y_res=oversample.fit_resample(X_train, y_train)

#Representación de los datos de entrenamiento
gr = Y_train.value_counts().reset_index()
gr = Y_res.value_counts().reset_index()
```

```
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='b')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

#Random Forest

pipe = Pipeline([('scaler', MinMaxScaler()),('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_train, y_train) #entrenamiento del modelo
y_pred = model.predict(X_test) #evaluación del modelo
print(classification_report(y_test, y_pred, zero_division=0))

#Representación de los atributos en función de su importancia en el modelo

forest = model['classifier']
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=0)
fi = pd.Series(importances, index=X.columns).reset_index()
fi.columns = ['att', 'importance']
fi = fi.sort_values(by='importance', ascending=False).reset_index(drop=True)
g = sns.barplot(data=fi, x='att', y='importance', color='b')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

cur.close()
conn.close()
```

ANEXO V: CÓDIGO DEL ARCHIVO *PRETTYDINAMIC.IPYNB*

```
#importar librerías

import pandas as pd
import sqlite3
import numpy as np
from tqdm import tqdm
import seaborn as sns
import matplotlib.pyplot as plt

#Librerias de Machine Learning

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler


#Diccionarios

#Tipo de buque

shiptype = dict.fromkeys(list(range(1,20)), 'reserved')
shiptype.update(dict.fromkeys(list(range(20,30)),'Wing In Ground'))
shiptype.update({30:'fishing',31:'tug',32:'tug',33:'dredger',34:'diving
ops',35:'military',36:'sailing',37:'pleasure craft'})
shiptype.update(dict.fromkeys(list(range(38,40)),'reserved'))
shiptype.update(dict.fromkeys(list(range(40,50)),'High speed craft'))
shiptype.update({50:'pilot',51:'SAR',52:'tug',53:'port tender',54:'Anti-pollution
equipment',55:'Law Enforcement'})
shiptype.update(dict.fromkeys(list(range(56,59)),'Spare-Local Vessel'))
shiptype.update({59:'noncombatant'})
shiptype.update(dict.fromkeys(list(range(60,70)),'Passenger'))
shiptype.update(dict.fromkeys(list(range(70,80)),'Cargo'))
shiptype.update(dict.fromkeys(list(range(80,90)),'Tanker'))
shiptype.update(dict.fromkeys(list(range(90,100)),'Other type'))
shiptype.update({0:'Not available', -1:'NaN'})

#Tipo de plataforma

platform = dict.fromkeys(list(range(0,100)), 'ship')
platform.update(dict.fromkeys(list(range(100,131)), 'beacon'))
platform.update(dict.fromkeys(list(range(131,256)), 'above'))
platform.update({0:'zero', -1:'nan'})

#Definición de funciones

#Función para la creación de variables cinemáticas

def cinematics(df):
```

```
    # time
    df = df.sort_values(by='tim')
    df.tim = pd.to_datetime(df.tim,unit='ms')
    df['dtime'] = df.tim.diff().dt.seconds
    hours = df.dtime/3600
    # linear
    dy = np.radians(df.lat.diff())*6373/1.852
    dx = np.radians(df.lon.diff())*6373/1.852
    df['nmi'] = np.sqrt(dx**2 + dy**2)
    df['sog_hat'] = df.nmi/hours
    df['sog_err'] = (df.sog_hat-df.sog)/(df.sog+0.1)*100
    df['acc_hat'] = df.sog_hat.diff()/hours
    df['jerk_hat'] = df.acc_hat.diff()/hours
    df['acc'] = df.sog.diff()/hours
    df['jerk'] = df.acc.diff()/hours
    # angular
    deg_norm = df.cog.diff()%360
    deg_diff = pd.concat([360-deg_norm, deg_norm], axis=1).min(axis=1)
    df['omega'] = np.radians(deg_diff)/hours
    df['alpha'] = df.omega.diff()/hours
    df['zeta'] = df.alpha.diff()/hours
    return df

#Función para la creación de estadísticos a partir de variables cinemáticas

def aggregate(df):
    #aceleración
    df['avg_acc'] = df['acc'].mean()
    df['sdv_acc'] = df['acc'].std()
    df['iod_acc'] = df['acc'].var()/df['acc'].mean()
    df['q50_acc'] = df['acc'].quantile(0.50)
    df['skw_acc'] = df['acc'].skew()
    df['krt_acc'] = df['acc'].kurt()
    #sog
    df['avg_sog'] = df['sog'].mean()
    df['sdv_sog'] = df['sog'].std()
    df['iod_sog'] = df['sog'].var()/df['sog'].mean()
    df['q50_sog'] = df['sog'].quantile(0.50)
    df['skw_sog'] = df['sog'].skew()
    df['krt_sog'] = df['sog'].kurt()
    #sog_hat
    df['avg_sog_hat'] = df['sog_hat'].mean()
    df['sdv_sog_hat'] = df['sog_hat'].std()
    df['iod_sog_hat'] = df['sog_hat'].var()/df['sog_hat'].mean()
    df['q50_sog_hat'] = df['sog_hat'].quantile(0.50)
    df['skw_sog_hat'] = df['sog_hat'].skew()
    df['krt_sog_hat'] = df['sog_hat'].kurt()
    #sog_err
    df['avg_sog_err'] = df['sog_err'].mean()
    df['sdv_sog_err'] = df['sog_err'].std()
    df['iod_sog_err'] = df['sog_err'].var()/df['sog_err'].mean()
    df['q50_sog_err'] = df['sog_err'].quantile(0.50)
    df['skw_sog_err'] = df['sog_err'].skew()
    df['krt_sog_err'] = df['sog_err'].kurt()
    #acc_hat
    df['avg_acc_hat'] = df['acc_hat'].mean()
    df['sdv_acc_hat'] = df['acc_hat'].std()
    df['iod_acc_hat'] = df['acc_hat'].var()/df['acc_hat'].mean()
```

```
    df['q50_acc_hat'] = df['acc_hat'].quantile(0.50)
    df['skw_acc_hat'] = df['acc_hat'].skew()
    df['krt_acc_hat'] = df['acc_hat'].kurt()
    #jerk_hat
    df['avg_jerk_hat'] = df['jerk_hat'].mean()
    df['sdv_jerk_hat'] = df['jerk_hat'].std()
    df['iod_jerk_hat'] = df['jerk_hat'].var()/df['jerk_hat'].mean()
    df['q50_jerk_hat'] = df['jerk_hat'].quantile(0.50)
    df['skw_jerk_hat'] = df['jerk_hat'].skew()
    df['krt_jerk_hat'] = df['jerk_hat'].kurt()
    #jerk
    df['avg_jerk'] = df['jerk'].mean()
    df['sdv_jerk'] = df['jerk'].std()
    df['iod_jerk'] = df['jerk'].var()/df['jerk'].mean()
    df['q50_jerk'] = df['jerk'].quantile(0.50)
    df['skw_jerk'] = df['jerk'].skew()
    df['krt_jerk'] = df['jerk'].kurt()
    #omega
    df['avg_omega'] = df['omega'].mean()
    df['sdv_omega'] = df['omega'].std()
    df['iod_omega'] = df['omega'].var()/df['omega'].mean()
    df['q50_omega'] = df['omega'].quantile(0.50)
    df['skw_omega'] = df['omega'].skew()
    df['krt_omega'] = df['omega'].kurt()
    #alpha
    df['avg_alpha'] = df['alpha'].mean()
    df['sdv_alpha'] = df['alpha'].std()
    df['iod_alpha'] = df['alpha'].var()/df['alpha'].mean()
    df['q50_alpha'] = df['alpha'].quantile(0.50)
    df['skw_alpha'] = df['alpha'].skew()
    df['krt_alpha'] = df['alpha'].kurt()
    #zeta
    df['avg_zeta'] = df['zeta'].mean()
    df['sdv_zeta'] = df['zeta'].std()
    df['iod_zeta'] = df['zeta'].var()/df['zeta'].mean()
    df['q50_zeta'] = df['zeta'].quantile(0.50)
    df['skw_zeta'] = df['zeta'].skew()
    df['krt_zeta'] = df['zeta'].kurt()
    return df

#Preprocesado de los datos y almacenamiento en sql

#creación de la conexión

path_data = r'../previo /Gonzalo/DinamicData.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()

#procesado del archivo CSV y creación de atributos

orig = ['TIMESTAMP','MMSI' 'IMO', 'COG', 'SOG', 'LATITUDE','LONGITUDE','LOA',
'SHIPTYPE AIS RAW']
subs = ['tim', 'mmsi','imo', 'cog', 'sog', 'lat', 'lon','loa','shiptype']

# Troceado y procesado del CSV

reader = pd.read_csv('../Gonzalo/DinamicData.csv', sep=';', chunksize=1000000)
```

```
for df in reader:
    chunk=df[orig]
    chunk.columns = subs
    chunk=chunk.replace({'shiptype': shiptype})#cambiar el valor numérico por su
valor en el diccionario
#eliminación de atributos no válidos
    chunk=chunk[chunk.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
    chunk=chunk.dropna()
    df = chunk[chunk.shiptype.isin(['fishing','tug','military','pleasure craft',
'Passenger','Cargo','Tanker'])]#seleccion de tipos de buque de interés
    df.to_sql('dinamic', con=conn, if_exists='append', index=False)

#creación de variables cinemáticas para cada IMO

df = pd.read_sql("""SELECT DISTINCT imo FROM dinamic WHERE imo<> 0 AND imo NOT
NULL AND sog BETWEEN 0 AND 130 AND sog>0  """, con=conn) #filtro
for m in tqdm(df.imo):
    dft = pd.read_sql(f"""SELECT mmsi, imo, tim, cog, sog, lat, lon, loa, shiptype
FROM dinamic WHERE imo = {m} """, con=conn)
    dft = cinematics(dft)
    dft.to_sql('pre_dinamic', con=conn, if_exists='append', index=False)

#creación de atributos a partir de estadísticos

for imo in tqdm(df.imo):
    dfe = pd.read_sql(f"""SELECT * FROM pre_dinamic WHERE imo = {imo} AND sog<>0
AND acc NOT NULL AND acc<>0 AND sog_hat<>0 AND sog_hat NOT NULL AND sog_err<>0 AND
sog_err NOT NULL AND acc_hat<>0 AND acc_hat NOT NULL AND jerk_hat<>0 AND jerk_hat
NOT NULL AND jerk<>0 AND jerk NOT NULL AND omega<>0 AND omega NOT NULL AND
alpha<>0 AND alpha NOT NULL AND zeta<>0 AND zeta NOT NULL """, con=conn) #filtro
de los datos no útiles
    dfe = aggregate(dfe)
    dfe.to_sql('complete_dinamic', con=conn, if_exists='append', index=False)

#eliminacion de datos repetidos

df=pd.read_sql("""SELECT DISTINCT * FROM complete_dinamic """, con=conn)

#eliminacion de elementos que no son atributos para ejecutar Random Forest
#mmsi se eliminará más tarde para poder cruzar datos dinámicos y estáticos

df=df[df.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
df=df.dropna()
df.drop('imo', axis=1, inplace=True)
df.drop('tim', axis=1, inplace=True)
df.drop('cog', axis=1, inplace=True)
df.drop('sog', axis=1, inplace=True)
df.drop('lat', axis=1, inplace=True)
df.drop('lon', axis=1, inplace=True)
df.drop('loa', axis=1, inplace=True)
df.drop('dtime', axis=1, inplace=True)
df.drop('nmi', axis=1, inplace=True)
df.drop('sog_hat', axis=1, inplace=True)
df.drop('sog_err', axis=1, inplace=True)
df.drop('jerk_hat', axis=1, inplace=True)
df.drop('acc_hat', axis=1, inplace=True)
df.drop('acc', axis=1, inplace=True)
df.drop('jerk', axis=1, inplace=True)
```

```
df.drop('omega', axis=1, inplace=True)
df.drop('alpha', axis=1, inplace=True)
df.drop('zeta', axis=1, inplace=True)

#almacenamiento de los atributos finales en SQL en una nueva tabla

 df.to_sql('best_dinamic', con=conn, if_exists='append', index=False)

 #selección de datos evitando barcos duplicados

 df=pd.read_sql("""SELECT DISTINCT * FROM best_dinamic """, con=conn)
 ddf=df.drop('mmsi',axis=1) #eliminación de datos que no se usan como atributo

 #Asignación de variable dependiente y variables independientes

 X=df.drop('shiptype', axis=1) #Variables independientes
 y=df.shiptype #Variable dependiente

#visualización de los datos disponibles

gr1 = y.value_counts().reset_index()
gr1.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr1, x='shiptype', y='freq', color='orangered')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

#Machine Learning

#División de los datos de partida en datos de entrenamiento y datos de test

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42)

#Undersampling

under = RandomUnderSampler()
X_res, Y_res=under.fit_resample(X_train, y_train)

#Oversampling

oversample=SMOTE(random_state = 42)
X_res, Y_res=oversample.fit_resample(X_train, y_train)

#Representación de los datos de entrenamiento

gr = Y_train.value_counts().reset_index()
gr = Y_res.value_counts().reset_index()
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='b')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

#Random Forest

pipe = Pipeline([('scaler', MinMaxScaler()),('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_train, y_train) #entrenamiento del modelo
y_pred = model.predict(X_test) #evaluación del modelo
```

```
print(classification_report(y_test, y_pred, zero_division=0))

#Representación de los atributos en función de su importancia en el modelo

forest = model['classifier']
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=0)
fi = pd.Series(importances, index=X.columns).reset_index()
fi.columns = ['att', 'importance']
fi = fi.sort_values(by='importance', ascending=False).reset_index(drop=True)
g = sns.barplot(data=fi, x='att', y='importance', color='b')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

cur.close()
conn.close()
```

ANEXO VI: CÓDIGO DEL ARCHIVO *PRETTYMIX.IPYNB*

```
#importar librerías

import pandas as pd
import sqlite3
import numpy as np
from tqdm import tqdm
import seaborn as sns
import matplotlib.pyplot as plt

#librerias de Machine Learning

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler

#establecer conexión a las tablas de datos dinámicos y estáticos

#Datos estáticos

path_datas = '../Gonzalo/DataStatic.sql'
conns = sqlite3.connect(path_datas)
cur = conns.cursor()

#Datos dinámicos

path_datad = r'../Gonzalo/DinamicData.sql'
connd = sqlite3.connect(path_datad)
curs = connd.cursor()

#conexión de salida de los datos combinados

path_out = '../Gonzalo/mix.sql'
conn = sqlite3.connect(path_out)
cr = conn.cursor()

#creación de la tabla combinada

static = pd.read_sql(""" SELECT mmsi, a, b, c, d,
draught,len,wid,ldivw,ldivd,wdivd,area,grith,aml,amt,vs,aol FROM best_static""",
con=conns)
static.to_sql('best_static', con=conn if_exists='append', index=False)
dinamic = pd.read_sql(""" SELECT * FROM best_dinamic""", con=connd)
dinamic.to_sql('best_dinamic', con=conn if_exists='append', index=False)
mix=pd.read_sql("""SELECT * FROM best_static INNER JOIN best_dinamic USING
(mmsi)""", con=conn)
mix.to_sql('best_mix', con=conn if_exists='append', index=False)

#selección de los atributos
```

```
df=pd.read_sql("""SELECT DISTINCT * FROM best_mix""", con=conn)
df=df[df.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
df=df.dropna()
df.drop('mmsi', axis=1, inplace=True)

#visualización de datos disponibles

gr = df.shiptype.value_counts().reset_index()
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='g')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);


#asignación de variables dependientes e independientes

X=df.drop('shiptype', axis=1)
y=df.shiptype


#Machine Learning

#División de los datos de partida en datos de entrenamiento y datos de test

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42)

#Undersampling

under = RandomUnderSampler()
X_res, Y_res=under.fit_resample(X_train, y_train)

#Oversampling

oversample=SMOTE(random_state = 42)
X_res, Y_res=oversample.fit_resample(X_train, y_train)

#Representación de los datos de entrenamiento

gr = Y_train.value_counts().reset_index()
gr = Y_res.value_counts().reset_index()
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='b')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

#Random Forest

pipe = Pipeline([('scaler', MinMaxScaler()),('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_train, y_train) #entrenamiento del modelo
y_pred = model.predict(X_test) #evaluación del modelo
print(classification_report(y_test, y_pred, zero_division=0))

#Representación de los atributos en función de su importancia en el modelo

forest = model['classifier']
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=0)
```

```
fi = pd.Series(importances, index=X.columns).reset_index()
fi.columns = ['att', 'importance']
fi = fi.sort_values(by='importance', ascending=False).reset_index(drop=True)
g = sns.barplot(data=fi, x='att', y='importance', color='b')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);

cur.close()
curs.close()
cr.close()
connd.close()
conns.close()
conn.close()
```