



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Validación de una selección de parámetros estáticos y dinámicos
para la predicción de tipo de buque utilizando técnicas de
inteligencia artificial*

Grado en Ingeniería Mecánica

ALUMNO: Raúl Fernández Martín

DIRECTORES: Belén Barragáns Martínez
Pablo Sendín Raña

CURSO ACADÉMICO: 2022-2023

UniversidadeVigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Validación de una selección de parámetros estáticos y dinámicos
para la predicción de tipo de buque utilizando técnicas de
inteligencia artificial*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

Este TFG se encuadra dentro de una línea de investigación del CUD-ENM centrada en la aplicación de técnicas de inteligencia artificial para la mejora del conocimiento del entorno marítimo. En particular, este trabajo supone una continuación de un TFG desarrollado en el curso 2021-2022, que exploró la posibilidad de predecir el campo “tipo de buque” (muchas veces no cubierto en los mensajes AIS) a partir de otros campos de esos mensajes.

En este TFG se pretende revisar en profundidad la propuesta previa, enfatizando en el preprocesado y tratamiento inicial de los datos de entrada, mejorando el código para optimizar tiempos de ejecución y comprobando de manera ordenada, exhaustiva y más completa la validez de la propuesta de parámetros que optimicen la calidad de la predicción del tipo de barco.

Como principales conclusiones, se confirma la bondad de la selección previa, habida cuenta de que se han replicado sus experimentos con éxito, y los resultados se mantienen aun variando las condiciones de los mismos (diferentes momentos temporales, diferentes tamaños del *dataset*, etc.). Asimismo, tras haber profundizado en la etapa de tratamiento previo de los datos, se ha mejorado el modelo de predicción consiguiendo aumentar la confianza y fiabilidad de sus resultados.

PALABRAS CLAVE

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

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer al personal destinado en la Escuela Naval Militar y en el Centro Universitario de la Defensa su esfuerzo y dedicación durante estos cinco años para contribuir en nuestra formación.

A continuación, expresar mi gratitud hacia mis tutores, Belén y Pablo, por su excelente labor dirigiendo este proyecto. Solo puedo sentir admiración hacia ellos por su entrega y compromiso, desde la asignación de este TFG hasta la realización del depósito, y por ayudarme a resolver cada obstáculo encontrado.

A mis compañeros de la promoción 423-153, quienes juntos hemos superado un sinfín de dificultades y con quienes he vivido en estos últimos años momentos inolvidables. Este camino no hubiese sido igual sin vosotros.

Y, por último, a mi familia. Siempre he contado con su apoyo incondicional, y sin ellos no sería la persona en la que me he convertido a día de hoy.

Gracias de corazón.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	5
1 Introducción y objetivos	7
1.1 Motivación y contexto	7
1.2 Objetivos del TFG.....	8
1.3 Organización de la memoria	8
2 Estado del arte	9
2.1 Inteligencia artificial	9
2.1.1 Definición	9
2.1.2 Contexto histórico.....	10
2.2 Técnicas de aprendizaje automático (<i>Machine Learning</i>)	11
2.2.1 Aprendizaje supervisado.....	12
2.2.2 Aprendizaje no supervisado.....	13
2.2.3 Aprendizaje semisupervisado	13
2.2.4 Aprendizaje por refuerzo	13
2.2.5 Deep learning.....	14
2.3 Algoritmo empleado de aprendizaje automático supervisado	14
2.4 COVAM y CEM	15
3 Desarrollo del TFG.....	17
3.1 Herramientas utilizadas en el desarrollo del proyecto	17
3.1.1 Jupyter Lab	17
3.1.2 Python	18
3.1.3 Bibliotecas empleadas	18
3.1.4 SQLite 3	18
3.1.5 Algoritmo de clasificación empleado	18
3.2 Descripción del conjunto de datos de entrada.....	19
3.2.1 Datos estáticos	19
3.2.2 Datos dinámicos.....	20
3.2.3 Combinación de datos estáticos y dinámicos.	20
3.3 Fase de preprocesado y depuración de datos de entrada.....	21
3.3.1 Limpieza de datos	21
3.3.2 Valores anómalos.....	21
3.3.3 Filtrado de tipos de buque.....	22

3.3.4 Eliminación de outliers	22
3.4 Fase de generación de atributos adicionales	29
3.4.1 Atributos estáticos	29
3.4.2 Atributos dinámicos	30
3.5 Fase de tratamiento de datos (previo a experimentos)	31
3.5.1 OverSampling	32
3.5.2 UnderSampling	32
3.5.3 MinMaxScaler	32
3.6 Fase de experimentación	32
3.6.1 Experimentos con datos estáticos	32
3.6.2 Experimentos datos dinámicos	37
4 Resultados obtenidos	39
4.1 Calidad del modelo	39
4.2 Resultados estáticos	40
4.3 Resultados dinámicos	44
5 Conclusiones y líneas futuras	47
5.1 Conclusiones	47
5.2 Líneas futuras	47
6 Bibliografía	49
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales	52
Anexo II: Reflexiones Éticas y Sociales	54
Anexo III: Resultados de otros experimentos realizados	56
Anexo IV: Código Static	72
Anexo IV: Código Dinamic	76

ÍNDICE DE FIGURAS

Figura 2-1 Talos: el autómatas gigante de la mitología [9]	10
Figura 2-2 Reconstrucción del león mecánico de Leonardo da Vinci [10].....	11
Figura 2-3 Tipos de aprendizaje automático [12].....	12
Figura 2-4 Procesos de clasificación y regresión [13].....	12
Figura 2-5. Aprendizaje por refuerzo [16]	14
Figura 2-6. Comparativa 1-100 árboles de decisión [17]	15
Figura 3-1 Ejemplos de datos estáticos de buques con el campo <i>draught</i> = 0 (elaboración propia)	21
Figura 3-2 Comparativa del tamaño del <i>dataset</i> de datos estáticos tras eliminar <i>draught</i> =0 (elaboración propia)	22
Figura 3-3 Influencia de <i>outliers</i> en una recta de regresión [22]	23
Figura 3-4 Agrupación de datos [22].....	23
Figura 3-5 <i>Outlier</i> de una dimensión [22].....	23
Figura 3-6 <i>Outliers</i> de dos dimensiones [22]	24
Figura 3-7 <i>Outliers</i> de tres dimensiones [22].....	24
Figura 3-8 Representación <i>boxplot</i> [24].....	25
Figura 3-9 Representación gráfica <i>plotbox</i> [26]	25
Figura 3-10 Distribución barcos de tipo ESTADO	25
Figura 3-11 Distribución barcos de tipo ESTADO eliminando <i>outliers</i>	25
Figura 3-12 Distribución barcos de tipo MERCANTE	26
Figura 3-13 Distribución barcos de tipo MERCANTE eliminando <i>outliers</i>	26
Figura 3-14 Distribución barcos de tipo MILITAR	26
Figura 3-15 Distribución barcos de tipo MILITAR eliminando <i>outliers</i>	26
Figura 3-16 Distribución barcos de tipo OCIO eliminando <i>outliers</i>	26
Figura 3-17 Distribución barcos de tipo OCIO	26
Figura 3-18 Distribución barcos de tipo PASAJEROS eliminando <i>outliers</i>	27
Figura 3-19 Distribución barcos de tipo PASAJEROS	27
Figura 3-20 Distribución barcos de tipo PESQUERO	27
Figura 3-21 Distribución barcos de tipo PESQUERO eliminando <i>outliers</i>	27
Figura 3-22 Distribución barcos de tipo PETROLEROS	27
Figura 3-23 Distribución barcos de tipo PETROLEROS eliminando <i>outliers</i>	27
Figura 3-24 Distribución barcos de tipo REMOLCADOR.....	28
Figura 3-25 Distribución barcos de tipo REMOLCADOR eliminando <i>outliers</i>	28
Figura 3-26 Distribución barcos de tipo SUBACUATICO eliminando <i>outliers</i>	28

Figura 3-27 Distribución barcos de tipo SUBACUATICO	28
Figura 3-28 Función para eliminar <i>outliers</i> (elaboración propia)	28
Figura 3-29 Función ‘statics’ para crear los atributos mencionados.....	29
Figura 3-30 Relación de los atributos estáticos calculados a partir de <i>draught</i> que alimentan al modelo (elaboración propia)	29
Figura 3-31 Función <i>cinematics</i> para la creación de atributos	30
Figura 3-32 Estadísticos que conformarán los atributos dinámicos	30
Figura 3-33 Ejemplo de resultados de predicción del tipo de buque con clases desbalanceadas (elaboración propia)	31
Figura 3-34 Comparativa <i>undersampling</i> vs <i>oversampling</i> [29].....	31
Figura 3-35 Ejemplos variaciones en el calado con mismo <i>MMSI</i> (elaboración propia).....	36
Figura 4-1 Matriz de confusión en un algoritmo de clasificación con datos desbalanceados [28]..	39
Figura 4-2 Importancia de los atributos empleados con 9 tipos de barcos con ‘LOA’	44
Figura 4-3 Importancia de los atributos empleados con 9 tipos de barcos sin ‘LOA’	45

ÍNDICE DE TABLAS

Tabla 3-1 Experimentos base	33
Tabla 3-2 Experimentos realizando ajustes en el número de árboles y la relación De/Dt	34
Tabla 3-3 Experimentos con los tipos de buque más representativos	34
Tabla 3-4 Experimentos con datos estáticos de 14 días de igual tamaño.....	35
Tabla 3-5 Experimentos con otro conjunto de datos estáticos de 14 días	35
Tabla 3-6 Experimentos con datos estáticos de los meses de enero y junio	36
Tabla 3-7 Experimentos con separación correcta con datos de entrenamiento y test, y $draught=0$	37
Tabla 3-8 Experimentos de datos estáticos del mes de junio 2022 eliminando <i>outliers</i>	37
Tabla 3-9 Experimentos con datos dinámicos.....	38

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Motivación y contexto

La inteligencia artificial (IA) se ha convertido en una de las tecnologías más importantes de la actualidad. Ha transformado la manera en la que interactuamos con nuestro entorno, favoreciendo la aparición de avances en una gran cantidad de campos como la medicina, la informática o la educación. Muchas industrias se han adaptado a esta nueva tecnología, lo que les ha permitido mejorar su eficiencia en muchos aspectos, y resolver situaciones complejas con soluciones innovadoras gracias a la IA.

La Armada [1], institución centenaria, tiene el constante propósito de actualizar los equipos, técnicas y tecnologías de las que dispone, como describe el Almirante Jefe del Estado Mayor de la Armada (AJEMA) en las líneas generales de la Armada. En ellas se reitera la necesidad de aplicar esta nueva tecnología para la resolución de problemas y el cumplimiento de la misión. En base a estas instrucciones, nacen distintos proyectos cuyo objetivo es la implementación de estas nuevas técnicas en el desarrollo de las actividades de las unidades. En el Centro de Operaciones y Vigilancia de Acción Marítima (COVAM) [2][3], estas medidas ya se han visto implementadas [4][5][6], redundando en una mejora de los sistemas que emplea este centro. En paralelo, la Armada encarga al CUD-ENM un proyecto de investigación y desarrollo para realizar un demostrador tecnológico que aplique técnicas de inteligencia artificial para la mejora del Conocimiento del Entorno Marítimo (CEM), en particular, para ayudar en la detección de determinadas anomalías en el comportamiento del tráfico marítimo.

Estas anomalías las define el COVAM y en dicha definición es un aspecto importante el tipo de buque que realiza dicho comportamiento. En dicho proyecto, el CUD-ENM recibe como datos de entrada el flujo AIS (*Automatic Identification System*) en tiempo real, y, tras analizarlo, se detecta que del orden de un tercio de los mensajes recibidos no rellenan el campo correspondiente al tipo de barco que emite dicho mensaje.

Para abordar este problema, se propone un TFG en el curso 2021-2022 [7] que plantea emplear técnicas de inteligencia artificial para poder predecir el valor de dicho campo a partir del resto de la información que contienen los mensajes AIS (datos estáticos, como las dimensiones del buque o dinámicos, como su velocidad o rumbo). Dicho TFG concluye presentando una selección de parámetros que maximizan la calidad de la predicción del tipo de buque, para tiempos de ejecución razonables. Sin embargo, dicho TFG adolece de una fase de validación completa para dicha propuesta.

En este contexto, se plantea el TFG actual, como una continuación de dicho trabajo, siendo su premisa revisar con detalle todo el desarrollo previo y sugerir mejoras.

1.2 Objetivos del TFG

Tras haber contextualizado el ámbito del trabajo en el apartado anterior, podemos decir que este TFG está enfocado en analizar con detalle la propuesta previa para poder validar de forma exhaustiva los resultados que obtiene. Para ello, se realizará una revisión en profundidad de la propuesta anterior, poniendo especial atención en el preprocesado y tratamiento inicial de los datos de entrada. Este proceso permitirá mejorar la calidad de los datos, lo que se traducirá, previsiblemente, en una mejora en la calidad de las predicciones.

Además, se buscará minimizar los tiempos de ejecución del código con el objetivo de aumentar la eficiencia del modelo de predicción. Para ello, se estudiará el código existente proponiendo alternativas más eficaces en el tratamiento de los datos. Si se consiguen reducir los tiempos de ejecución, se podrá plantear la realización de un abanico mayor de experimentos y que estos, a su vez, puedan emplear conjuntos de datos de entrada de mayor tamaño.

Asimismo, se comprobará la validez de la propuesta de parámetros realizada en el TFG previo y su consistencia bajo diferentes condiciones. Para ello se realizarán una serie de ensayos y experimentos que identifiquen cuáles son los parámetros con mayor repercusión para la predicción del tipo de barco en diferentes circunstancias (momento temporal, tamaño del *dataset*, etc.)

1.3 Organización de la memoria

En este apartado, se describirá la estructura de esta memoria para la organización del contenido de la misma. Para ello, la memoria se ha dividido en seis capítulos:

- Capítulo 1: En esta sección, como acabamos de presentar, se introduce el contexto del TFG y se expone la principal motivación del trabajo. Se presentan los objetivos buscados y se detalla la estructura de la memoria.
- Capítulo 2: Estado del arte. En este apartado se pretende proporcionar una base de conocimientos necesarios para la comprensión del trabajo. Se definen los conceptos de inteligencia artificial, así como su evolución hasta la actualidad, y las principales técnicas de aprendizaje automático, donde se incluyen los diferentes tipos, dependiendo de su funcionamiento. Por último, se explican términos empleados por la Armada o, relacionados con la navegación que resultan fundamentales para este trabajo.
- Capítulo 3: Desarrollo del TFG. En este capítulo se describen todas aquellas herramientas utilizadas en el proceso de experimentación, además de una descripción de los datos de entrada generados por el CUD-ENM. A continuación, se detalla todo el trabajo realizado en la fase de preprocesado y depuración de los datos de entrada, en la que se incluyen técnicas variadas para mejorar este conjunto, seguida de la generación de atributos, que nos permitirán realizar la predicción del modelo. Por último, se presentará el trabajo realizado en una última fase de tratamiento de datos previa al entrenamiento del modelo, seguido de la descripción de los experimentos realizados.
- Capítulo 4: Resultados obtenidos. Este capítulo incluirá una explicación de las métricas que evalúan la calidad del modelo y los resultados obtenidos en los experimentos realizados en el capítulo anterior, empleando dichas métricas para medir la bondad de la predicción.
- Capítulo 5: Conclusiones y líneas futuras. Este apartado resumirá las principales conclusiones obtenidas, analizando el cumplimiento de los objetivos de este TFG, además de indicar las posibles líneas futuras del proyecto.
- Capítulo 6: Bibliografía. En ella se indican las fuentes consultadas durante la realización de este trabajo. Por último, se adjuntan una serie de anexos que complementan el desarrollo del trabajo.

2 ESTADO DEL ARTE

En este capítulo se explicarán aquellos conceptos necesarios para la comprensión del trabajo y posterior análisis de los resultados obtenidos. Algunos de estos conceptos, como inteligencia artificial (IA) o *Machine Learning*, han surgido gracias al desarrollo tecnológico vivido en los últimos años. Como consecuencia, uno de los principales objetivos del trabajo es la aplicación de estas modernas técnicas en el entorno de la Armada, buscando resolver situaciones desfavorables o de posible peligro, permitiendo el correcto desarrollo de las obligaciones de ciertos organismos como el COVAM, en particular, la clasificación de buques.

2.1 Inteligencia artificial

2.1.1 Definición

La inteligencia artificial [8] se puede definir de manera muy simple como la capacidad que tienen las máquinas de imitar comportamientos y capacidades humanas, es decir, realizar tareas propias de una inteligencia humana. No obstante, en la actualidad se pueden encontrar muchas definiciones de este concepto, al ser su primer término (inteligencia) otra palabra cuya definición puede tener diversas variantes o interpretaciones. La RAE [9] define la IA como “disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.” Cabe destacar que “inteligencia artificial” ha sido elegida como la expresión del año 2022 debido a su importancia y presencia en la sociedad a día de hoy, y en el futuro.

Los ordenadores han sufrido una constante evolución desde su creación hasta el día de hoy. En sus inicios, estos equipos contaban con capacidad de cálculo, y con una no muy elevada capacidad de almacenar o memorizar datos. Sin embargo, con los avances en los diferentes ámbitos que afectan a estas tecnologías estas capacidades mencionadas se han visto enormemente ampliadas.

De igual manera, en los últimos años, la evolución de las matemáticas y de las tecnologías ha provocado un avance considerable en estas máquinas, pudiéndose observar la aparición de otras capacidades, como autoaprendizaje y creatividad. Estas cualidades acercarían el pensamiento humano a las máquinas.

Sin embargo, cuando se habla de IA, otra cualidad propia del pensamiento humano es la autoconciencia, la cual apenas se ha visto reflejada en estas máquinas, y es considerada, con los avances actuales, como ciencia ficción, por lo que no se considera posible a corto plazo.

La creatividad es la habilidad de inventar, crear, encontrar soluciones o generar ideas ante situaciones o problemas que se plantean. Un aspecto de gran interés son los límites que puede establecer la IA en esta materia. Este concepto es conocido como creatividad computacional, un subcampo de la

inteligencia artificial que estudia la creación de un software que actúe de manera ingeniosa. A día de hoy, algunos programas pueden generar contenidos creativos como dibujos, música, e incluso chistes.

Sin embargo, aunque el crecimiento tecnológico haya originado un aumento de procesos creativos en diversas materias muy variadas, aproximadamente el 90% de las veces en las que se utiliza la inteligencia artificial, esta se basa en su capacidad de autoaprendizaje. Esta habilidad, conocida también como aprendizaje automático, o en inglés *Machine Learning*, se puede considerar a día de hoy un sinónimo del término inteligencia artificial.

2.1.2 Contexto histórico

Aunque los primeros ordenadores digitales surgieron a mediados del siglo XX, la idea de resolver problemas por parte de las máquinas de manera automática imitando el pensamiento humano ha estado presente desde hace más de 2500 años. La primera aparición del concepto de vida artificial o inteligencia de las máquinas fue en el año 650 a.C, a través de las historias o mitos de la época. El personaje de esta historia era Talos [10], robot creado a petición de Zeus para proteger la isla de Creta de posibles ataques enemigos. Este robot mostrado en la Figura 2-1 estaba programado para realizar rondas de vigilancia y combatir en caso necesario.



Figura 2-1 Talos: el autómeta gigante de la mitología [10]

Hacia el año 322 a.C Aristóteles mostraba su inquietud por reproducir las capacidades humanas en las máquinas. Herón de Alejandría, ingeniero, físico y matemático explicó los conocimientos adquiridos sobre la idea de la existencia de robots, en su trabajo “Autómata” en el siglo I d.C. En él describía objetos y artefactos de autoría propia y ajena, cuyos componentes eran principalmente engranajes, piezas mecánicas y mecanismos de conducción de agua para diferentes fines. A Herón también se le atribuye la creación del dispositivo Eolípila, capaz de transformar energía térmica a mecánica, predecesor de la máquina de vapor, principal causante de la Revolución Industrial.

Egipto también fue un referente debido a la construcción de numerosos artefactos mecánicos facilitados por los avances de la época. Otro personaje histórico a destacar es Leonardo da Vinci, quien con sus aportaciones demostró su deseo de crear alguna forma de vida artificial. Esto pudo verse en su creación del León mecánico, encargo de Francisco I en el siglo XVI, que se puede ver en la Figura 2-2.



Figura 2-2 Reconstrucción del león mecánico de Leonardo da Vinci [11]

A pesar de ser una idea asentada entre las mentes más brillantes desde hace más de 2000 años, cuando nos referimos al término inteligencia artificial hablamos de una tecnología casi centenaria, relativamente moderna. Si tenemos que señalar un momento como el origen de la inteligencia artificial sería en 1943, año en el que los científicos Warren McCulloch y Walter Pitts publicaron un artículo llamado “*A logical calculus of ideas immanent in nervous activity*”, en el cual propusieron un modelo matemático para la creación de una red neuronal.

Sin embargo, Alan Turing es considerado como el padre de la inteligencia artificial, además del precursor de la informática moderna, y explicar su contribución a la informática y a las matemáticas es fundamental para conocer la historia de la inteligencia artificial. El matemático intentó darle respuesta a la pregunta “¿pueden las máquinas pensar?”. Para solucionar esa cuestión creó una prueba conocida como “test de Turing”, en el que pone a prueba la capacidad de la máquina examinada en imitar el pensamiento humano. La prueba requiere tres participantes: un interrogador y dos interrogados (una persona y una máquina). El objetivo del interrogador es averiguar cuál de los interrogados es la máquina, mientras que la máquina intenta convencer al interrogador de que es una persona. Fue el año 1950 cuando Turing publicó su artículo “*Computing machinery and intelligence*” donde incluía este método de evaluación y asentaba las bases de lo que sería la inteligencia artificial.

Para superar el test, el interrogador debe ser incapaz de determinar si el interrogado es humano o máquina. En caso de superar esta evaluación se consideraría a la máquina inteligente. Esto ocurrió por primera vez en el año 2014, cuando el bot “Eugene Goostman” [12] superó de manera exitosa la prueba al engañar a más del 30% del jurado (límite necesario para superar el test de Turing), específicamente a un 33%. No obstante, existe el debate de si el test ya había sido superado anteriormente, pero en esas competiciones similares existían límites de tiempo o temas preestablecidos, por lo que no se dieron como válidos.

Aunque Turing fue el primer y gran impulsor de la inteligencia artificial, este término no fue introducido hasta el año 1956 por parte del matemático John McCarthy, quien acuñó formalmente este término durante la conferencia “*Dartmouth Summer Research Project on Artificial Intelligence*”. De esta manera conseguía diferenciar esta materia de otros campos como la cibernética.

2.2 Técnicas de aprendizaje automático (*Machine Learning*)

El *Machine Learning* (ML) o aprendizaje automático es un conjunto de técnicas que permiten mejorar el funcionamiento de un algoritmo a partir de la experiencia, modificando su comportamiento basándose en acciones pasadas o resultados preestablecidos. Su funcionamiento se basa en la detección de patrones y ajuste de las acciones del programa en consecuencia. Proporciona a las máquinas la capacidad de aprender sin haber sido programadas explícitamente para ello.

El aprendizaje automático es utilizado como solución a diferentes problemas reales. Sin embargo, no todas estas cuestiones se resuelven de la misma manera, sino que tienen diferentes enfoques y deben

afrontarse de una manera óptima para cada dificultad. Debido a este motivo, dentro del aprendizaje automático se pueden distinguir diferentes tipos según el modo de aprendizaje que utilice el algoritmo y los problemas que pueda solucionar o errores que pueda corregir. Estos son aprendizaje supervisado, no supervisado, semi-supervisado y aprendizaje por refuerzo, como se puede observar en la Figura 2-3.

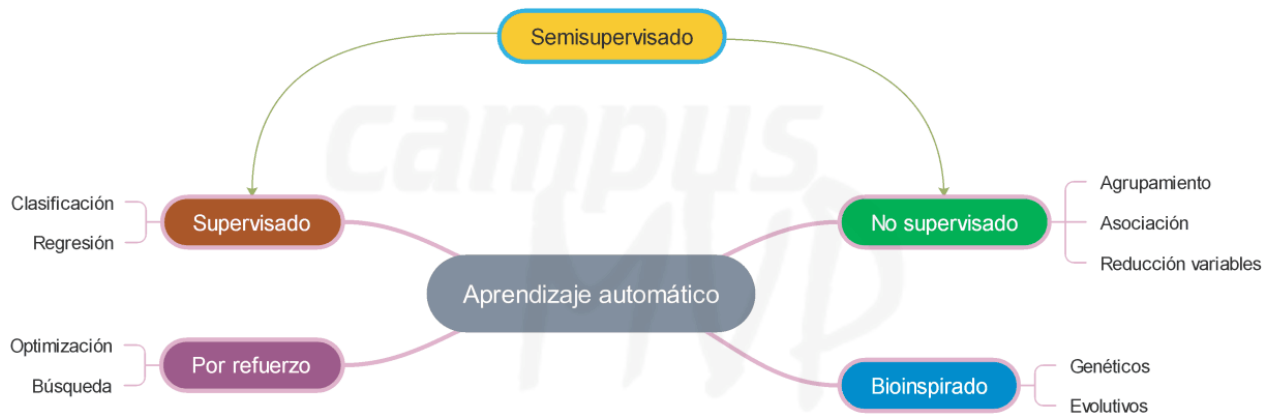


Figura 2-3 Tipos de aprendizaje automático [13]

2.2.1 Aprendizaje supervisado

Es el más utilizado. En este tipo de aprendizaje se utiliza un modelo de datos que actúa como guía y enseña al algoritmo las conclusiones que debe obtener. El algoritmo se entrena con un conjunto de datos ya etiquetados con un resultado definido previamente. Un ejemplo simple puede ser el algoritmo capaz de reconocer a una persona en diferentes fotografías. Para ello, es necesario decirle al algoritmo en cuales de esas instantáneas está la persona en concreto. El algoritmo busca el patrón y es capaz de reconocer a esa persona en nuevas imágenes. Este proceso recibe el nombre de entrenamiento o aprendizaje, y para su mejor funcionamiento es necesario recopilar una gran cantidad de datos.

Una vez finalizado el entrenamiento del algoritmo, este es capaz de procesar nuevos datos y proporcionar un resultado adecuado sin ayuda, por lo que lo podrá ser utilizado para solucionar problemas de clasificación y regresión [14].

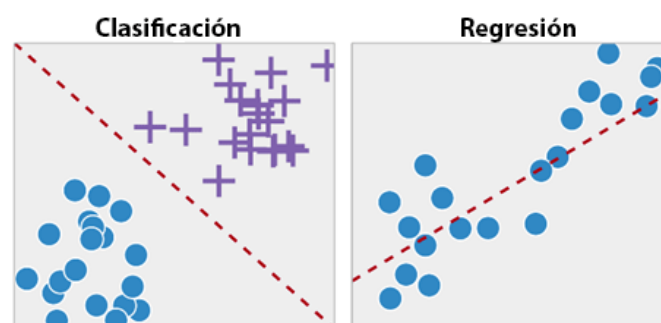


Figura 2-4 Procesos de clasificación y regresión [14]

En los primeros, se busca que el algoritmo sea capaz de asignar un dato nuevo o entrada a una de las categorías preestablecidas, denominadas clases o etiquetas, de manera correcta. Un ejemplo puede ser el utilizado en los servidores del correo electrónico para identificar ciertos correos como *spam*.

Respecto a los problemas de regresión, el entrenamiento al que se somete el algoritmo es similar al caso anterior. En este caso lo que se desea predecir no es una categoría, sino un número dentro de un conjunto de infinitos posibles valores. Un ejemplo de este tipo de algoritmos son aquellos modelos que

permiten determinar la altura de las personas de manera aproximada basándose en su sexo, edad y nacionalidad.

Por lo tanto, la principal diferencia entre estos dos tipos de aprendizaje supervisado es la finalidad. Al utilizar un algoritmo de regresión, se busca predecir valores continuos, mientras que la clasificación asigna el dato a una clase o categoría establecida, es decir, valores discretos.

Los algoritmos de aprendizaje supervisado tienen aplicaciones variadas, como por ejemplo reconocimiento facial, de voz, de huellas digitales o coches de conducción automática.

2.2.2 Aprendizaje no supervisado

En el aprendizaje no supervisado el algoritmo aprende a identificar los patrones sin la guía humana, y utiliza datos no etiquetados para el entrenamiento, siendo esta la principal diferencia entre el aprendizaje supervisado y el no supervisado. Es por ello que a estos algoritmos se les denomina descriptivos, a diferencia de los de aprendizaje supervisado, conocidos como predictivos. El aprendizaje no supervisado tiene como objetivo comprender las relaciones dentro de los conjuntos de datos. Los problemas en los que se suele aplicar este método son tareas de agrupamiento, de asociación y reducción de variables.

Al realizar tareas de agrupamiento, se analiza la muestra y se calcula la distancia entre los datos, es decir, analiza la similitud de los mismos. Esta técnica recibe el nombre de *clustering* y facilita la visualización de datos en grupos, sirviendo como un método de clasificación básico cuando no hay etiquetas en las que asignar los datos.

Otra aplicación es la búsqueda de valores comunes en las variables de la muestra. Establece unas reglas de asociación buscando la concurrencia entre los datos.

También se utiliza para la reducción de variables, en la que se busca conocer el peso o importancia que tiene un tipo de dato, es decir, que tipo aporta más información. De esta manera es posible eliminar aquellos datos redundantes y eliminarlos de la muestra, simplificando así el análisis.

2.2.3 Aprendizaje semisupervisado

Este tipo de aprendizaje combina los dos métodos mencionados anteriormente, ya que utiliza tanto datos etiquetados como datos no etiquetados. Esta técnica nace con el propósito de solucionar las posibles ineficiencias que pueden originar ambos métodos. El aprendizaje supervisado requiere tiempo para etiquetar todos los datos a utilizar y exige formación del personal encargado, generando ciertos costes en el proceso, que en algunos casos puede llegar a ser inviable. Sin embargo, la precisión obtenida es mayor que en los otros métodos. En cuanto al aprendizaje no supervisado, los costes se ven reducidos en gran medida debido a la utilización de datos no etiquetados. No obstante, los resultados obtenidos no son tan precisos. El aprendizaje semisupervisado nace con el propósito de obtener el equilibrio entre los resultados obtenidos respecto al coste del proceso.

El aprendizaje supervisado utiliza un conjunto limitado de datos etiquetados y un conjunto mucho más grande de datos no etiquetados. El sistema aprovecha los datos no etiquetados para generar modelos predictivos mejorando las prestaciones que daría usar únicamente datos etiquetados.

2.2.4 Aprendizaje por refuerzo

Este tipo de aprendizaje (*Reinforcement Learning*) [15] es una variedad del *Machine Learning* que permite a una inteligencia artificial planear estrategias efectivas en base a la experimentación con los datos. Consiste en realizar un experimento o actividad de manera continuada, estudiando las decisiones

tomadas y los resultados a los que le han llevado a la máquina. De esta manera debe ser capaz de aprender cuales de esas decisiones debe tomar para conseguir el objetivo y cuales debe descartar. Es denominado comúnmente como proceso de prueba y error. No requiere datos previamente etiquetados para el aprendizaje del algoritmo y se diferencia de los modelos de aprendizaje supervisado y no supervisado en que busca maximizar la recompensa [16], mientras que los últimos dos persiguen minimizar los costes.

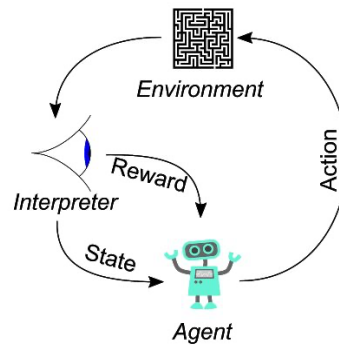


Figura 2-5. Aprendizaje por refuerzo [17]

Su funcionamiento se basa en un constante bucle de estado, acción y recompensa, como se puede observar en la Figura 2-5. El agente situado en un medio ambiente realiza distintas acciones hasta que obtiene una respuesta. Esta puede ser positiva (recompensa) o negativa. Un ejemplo de este ciclo es el famoso videojuego Pacman, basado en este funcionamiento. Este tipo de aprendizaje se utiliza en un amplio abanico de aplicaciones, desde partidas de ajedrez hasta tareas de gran dificultad.

2.2.5 Deep learning

Es una de las ramas del *Machine Learning*. Se denomina como aprendizaje profundo y es un tipo de algoritmo basado en redes neuronales profundas. Se encuentra en nuestro día a día y está integrado en prácticamente la totalidad de aparatos que utilizamos.

2.3 Algoritmo empleado de aprendizaje automático supervisado

En este apartado describiremos el algoritmo empleado en este TFG: *Random Forest*. Se trata de un algoritmo enmarcado dentro de las técnicas de aprendizaje automático supervisado. *Random Forest* es un algoritmo basado en un conjunto de árboles de decisión combinados. Cada árbol se entrena con distintas muestras de datos para un mismo problema. De esta forma, al combinar sus resultados, unos errores se compensan con otros y tenemos una predicción mejor o de mayor ajuste.

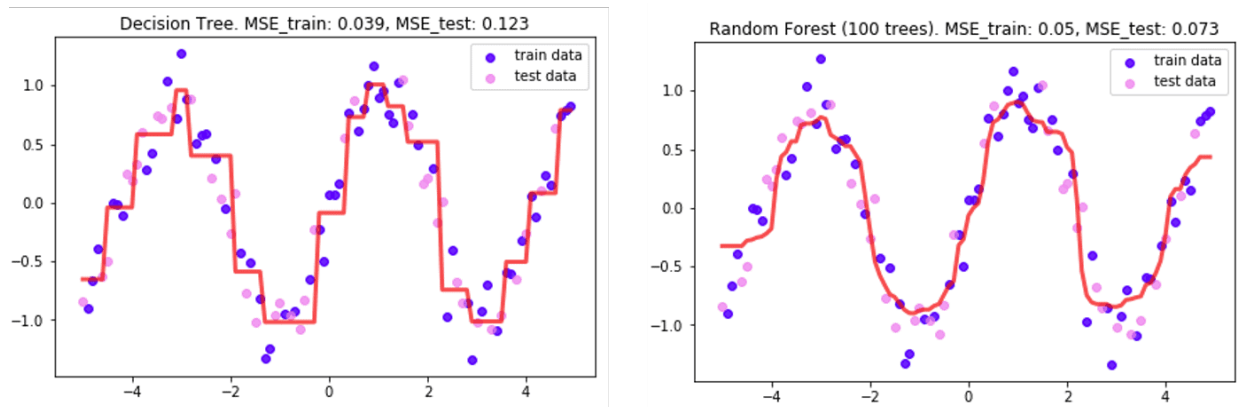


Figura 2-6. Comparativa 1-100 árboles de decisión [18]

Para poder entender qué es *Random Forest*, primero es necesario conocer el concepto “*bagging*”. Los métodos *bagging* son aquellos en los que se trabaja con algoritmos simples de manera paralela. Este tipo de aprendizaje es utilizado comúnmente para reducir la varianza en un conjunto de datos. Por lo tanto, podríamos decir que los árboles de decisión son considerados como candidatos ideales para este tipo de trabajos.

La principal ventaja de este algoritmo es que, utilizado con un conjunto de datos lo suficientemente grande, los resultados son muy certeros. Sin embargo, si los datos contienen grupos de atributos correlacionados con similar relevancia para el rendimiento, entonces los grupos más pequeños están favorecidos sobre los grupos más grandes.

2.4 COVAM y CEM

El COVAM es el Centro de Operaciones y Vigilancia de Acción Marítima y su función es proveer a la Fuerza de Acción Marítima (FAM) de toda la información que ayude a conocer que sucede en los espacios marítimos de interés nacional, a lo que se conoce como “Conocimiento del Entorno Marítimo” (CEM) [19].

El objetivo de una unidad como el COVAM es facilitar una representación real y precisa de la actividad marítima en las aguas de interés nacional, concepto que se conoce como “*Recognized Maritime Picture*” (RMP). El COVAM proporciona esta información a todos los buques de la Armada desplegado en distintas misiones, y a un determinado número de Agencias Civiles que lo solicitan.

Para cumplir con este objetivo, la Armada ha introducido muchos equipos relacionados con la inteligencia artificial en el COVAM, ya que su labor es a día de hoy fundamental para el desarrollo de nuestras actividades. Una de las maneras de mantener nuevas propuestas es a partir de este TFG, que tiene como objetivo seguir desarrollando este modelo de clasificación para implementarlo en las unidades en un posible futuro.

3 DESARROLLO DEL TFG

Tras conocer en el capítulo anterior algunos conceptos considerados fundamentales para el desarrollo de este proyecto, como inteligencia artificial o *Machine Learning*, y definir las principales características de los posibles algoritmos a utilizar basándonos en los objetivos de este trabajo, se procederá a explicar los programas y las herramientas a utilizar en el desarrollo del presente TFG.

Para ello también será necesario desarrollar tanto el algoritmo final a utilizar, como la optimización y tratamiento de datos, además de las técnicas o experimentos realizados a partir de los cuales obtendremos unos resultados, que serán analizados en el próximo capítulo.

3.1 Herramientas utilizadas en el desarrollo del proyecto

Manteniendo la línea del Trabajo Fin de Grado precursor de este proyecto, se conserva el empleo del mismo entorno de programación, *Jupyter Lab*, y el mismo lenguaje de programación, *Python*.

3.1.1 Jupyter Lab

Jupyter Lab [20] es un entorno *web* de desarrollo para programación en diferentes lenguajes que permite trabajar de manera muy sencilla con diferentes *notebooks* o cuadernos donde se escribe el código para su posterior ejecución. Esta interfaz fue creada con el fin de respaldar la computación científica y análisis de datos mediante el desarrollo de software de código abierto. Los *notebooks* se caracterizan por estar formados por celdas, donde se pueden definir fórmulas matemáticas, código, imágenes, elementos multimedia o textos.

Jupyter Lab es el sucesor oficial de *Jupyter Notebook*, que supuso el comienzo del proyecto *Jupyter* con la ciencia de datos como principal fin. Este proyecto nació con soporte para tres lenguajes de programación: *Julia*, *Python* y *R*. Sin embargo, actualmente *Jupyter Lab* soporta más de 50 lenguajes de programación.

Entre las principales características de la interfaz empleada se pueden destacar:

- Entorno web: en el empleo de la interfaz: permite la ejecución en un servidor remoto sin la necesidad de instalar nada en el entorno local. La ejecución se realiza en el servidor, por lo que las tareas complejas pueden quedar ejecutándose sin la necesidad de tener que estar conectados.
- Fragmentación de código: permite la ejecución del código por partes, favoreciendo así el estudio o modificación de alguna función en concreto sin necesidad de ejecutar todo el código de un archivo, empleando recursos como tiempo y energía de manera innecesaria. Esta propiedad también permite la detección y corrección rápida y sencilla de errores en los diferentes bloques del código.

3.1.2 Python

Python [21] es un lenguaje de programación de propósito general muy utilizado en la actualidad debido a su alta similitud con el lenguaje humano, haciéndolo muy intuitivo y fácil de emplear. Es un lenguaje muy adecuado para proyectos de aprendizaje automático e inteligencia artificial debido a las bibliotecas existentes que permiten el desarrollo de este tipo de proyectos.

Es un lenguaje muy versátil, ya que permite crear una gran variedad de programas diferentes sin especializarse únicamente en alguno de ellos. Además, al igual que la interfaz *Jupyter Lab*, es de código abierto, por lo que su uso y distribución son totalmente gratuitos.

Este lenguaje contiene las características y capacidades necesarias para el desarrollo de este trabajo. Asimismo, aunque la interfaz *Jupyter Lab* permite trabajar con una gran variedad de lenguajes de programación, fue originalmente diseñada para trabajar con este lenguaje. Por estas razones se continuará con el uso de *Python* en el desarrollo del proyecto.

3.1.3 Bibliotecas empleadas

Como se ha mencionado anteriormente, *Python* dispone de una gran variedad de bibliotecas adecuadas para abordar problemas de IA, lo que hace que sea un lenguaje adecuado a nuestros fines. Estas bibliotecas son conjuntos de funciones que permiten realizar tareas muy diversas de manera eficiente como cálculos numéricos, análisis de datos o visualización de los mismos.

Las librerías utilizadas en este TFG son:

- a. Pandas: para el manejo, análisis y procesamiento de datos.
- b. SQLite3: permite el empleo de bases de datos SQLite3.
- c. Numpy: para el cálculo numérico y análisis de datos.
- d. Seaborn: permite crear gráficos estadísticos.
- e. Matplotlib: creación de gráficos y visualización de datos en dos dimensiones.
- f. Sklearn: implementa los algoritmos de clasificación que utilizamos en este TFG.
- g. Imblearn: define las funciones de clasificación de datos con clases desbalanceadas.

3.1.4 SQLite 3

Para el almacenamiento persistente de los atributos (que definiremos en una sección posterior de este mismo capítulo) hemos seleccionado *SQLite3*, en primera instancia, por tratarse de un trabajo que continúa un TFG previo. Además, es una solución simple, sencilla y suficientemente eficaz para el propósito de este trabajo.

La librería *pandas*, que también permite almacenamiento de los datos, no nos sirve como única solución para desarrollar los objetivos de este TFG realizaremos múltiples experimentos sobre un mismo conjunto de atributos. Generar esos atributos en cada uno de los experimentos a lo que nos obligaría el uso de *pandas* sería una aproximación menos eficiente en términos de tiempo de ejecución.

3.1.5 Algoritmo de clasificación empleado

En el capítulo dos se han introducido los algoritmos de clasificación en general. Una vez establecido el empleo de las técnicas del *Machine Learning* para el desarrollo de este trabajo, se procederá a la elección del modelo de aprendizaje del algoritmo que se va a utilizar. Esta debe estar basada en el objetivo del trabajo, que es clasificar los nuevos datos que reciba el sistema en salidas previamente establecidas.

Por lo tanto, comparando entre aprendizaje supervisado y no supervisado, se puede observar que el primero cumple por definición los requisitos planteados anteriormente. Es por esto que se utilizarán algoritmos de aprendizaje supervisado para este TFG.

Con todo, el empleo de este tipo de algoritmos requiere un gran volumen de datos etiquetados para el entrenamiento de los mismos. Sin embargo, a diferencia del trabajo previo, se dispone de un volumen de datos más grande, debido a la recopilación de datos realizada a lo largo del curso actual, lo que permite validar los resultados pasados y realizar un mayor número de experimentos que serán explicados más adelante, además de las técnicas utilizadas para la optimización final de los datos. Dentro de las técnicas de aprendizaje supervisado, se pueden encontrar diferentes métodos, explicados en el capítulo anterior.

Finalmente, el algoritmo seleccionado ha sido *Random Forest* debido a dos motivos: el primero y más importante, es el elegido por el trabajo que queremos validar, por lo que al emplear el mismo algoritmo que en el TFG previo, podemos replicar los experimentos y comparar los resultados. El segundo, debido a que se adecúa perfectamente a nuestro propósito.

3.2 Descripción del conjunto de datos de entrada

Seleccionado el algoritmo a utilizar en nuestro proceso de clasificación, el siguiente paso es disponer de una fuente de datos que nos permita entrenar el algoritmo para que sea capaz de predecir la variable *shiptype* en aquellos mensajes AIS que no cubran esta información. Al utilizar un algoritmo de aprendizaje supervisado, utilizamos un conjunto de datos ya etiquetados con un resultado definido previamente para su entrenamiento. Este conjunto de datos ha sido generado por el CUD-ENM, que a partir del flujo de datos AIS proporcionado por el COVAM en tiempo real, ha recopilado una gran cantidad de mensajes AIS transmitidos por diferentes buques en distintas franjas temporales, pudiendo experimentar con datos de diferentes épocas del año, considerando posibles eventos puntuales que hayan podido tener cualquier tipo de incidencia en nuestros experimentos.

Existen dos tipos de transpondedores AIS: clase A y clase B. Los barcos que requieran este equipo, en base a criterios técnicos [22] establecidos por organismos internacionales, como ayuda para la navegación dispondrán del tipo A, mientras que el tipo B son usado de manera voluntaria para las necesidades de la navegación. Los datos AIS se clasifican en 26 tipos de mensajes distintos entre ellos. Los datos estáticos se transmiten en mensajes tipo 5, mientras que los dinámicos corresponden a mensajes de tipos 1, 2 y 3. El contenido de estos mensajes se detalla en los siguientes subapartados.

3.2.1 Datos estáticos

Este tipo de datos transmitidos en los mensajes AIS de tipo 5 son aquellos datos fijos de la plataforma, es decir, aquellos que no varían con el tiempo. Esta información se transmite cada seis minutos, ya que al mantenerse constante no existe la necesidad de aumentar la frecuencia de transmisión de esta información. Entre los datos incluidos en estos mensajes estáticos podemos destacar el nombre de la embarcación, el *MMSI*, el distintivo internacional, *IMO*, las dimensiones y el tipo de buque.

Una parte del trabajo supone la experimentación únicamente con datos estáticos, por lo cual utilizaremos estos datos recopilados de los mensajes mencionados anteriormente, creando *datasets* o archivos de formato ‘csv’ para facilitar su manipulación. Los datos estáticos que utilizaremos en estos experimentos son:

1. *MMSI*: en inglés *Maritime Mobile Service Identity*, es un número compuesto de nueve dígitos únicos para cada estación usuario, ya sean buques como estaciones costeras.
2. *to_bow*: representado en el *dataset* como variable A, hace referencia a la distancia en metros desde el transpondedor AIS hasta la parte más aproada del buque.

3. *to_stern*: representado en el *dataset* como variable B, mide la distancia en metros desde el transpondedor AIS hasta la popa del buque.
4. *to_port*: representado en el *dataset* como variable C, hace referencia a la distancia en metros desde el transpondedor AIS al costado de babor.
5. *to_starboard*: representado en el *dataset* como variable D, determina la distancia en metros desde el transpondedor AIS al costado de estribor.
6. *Draught*: calado del buque en metros.
7. *Shiptype*: tipo de buque.

3.2.2 Datos dinámicos

Este tipo de datos hace referencia a aquella información que varía con el paso del tiempo para cada barco. La frecuencia de transmisión de estos datos es mayor que la de los estáticos, ya que se transmite aproximadamente entre dos y diez segundos cuando se está navegando, y cada tres minutos en caso de estar fondeado. Esto permite realizar un seguimiento del buque en cuestión.

Se han generado diversos archivos que incluyen los datos dinámicos contenidos en los mensajes AIS transmitidos por diferentes buques, recopilando un elevado volumen de datos. En estos archivos están definidos muchos campos, de los que solo nos quedaremos con los utilizados en el TFG que vamos a validar, es decir, aquellos que su estudio nos permita obtener información de los contactos, de sus características o de posibles patrones de navegación que realicen. Los campos seleccionados son:

1. *MMSI*: definido en apartado 3.2.1.
2. *IMO*: número de siete dígitos únicos de cada buque para su identificación.
3. *Timestamp*: sello temporal a la que el receptor recibe el mensaje AIS.
4. Latitud.
5. Longitud.
6. *COG*: (*Course Over Ground*). Rumbo sobre fondo marino.
7. *SOG*: (*Speed Over Ground*). Velocidad sobre fondo marino.
8. *Shiptype (AIS)*: tipo de buque.

El campo *shiptype* está cubierto por un código numérico de dos dígitos asociado con un tipo de buque específico. Por ejemplo, los buques militares se representan con el código treinta y cinco, mientras que los buques de transporte de pasajeros tienen asignado desde el número sesenta hasta el sesenta y nueve.

Sin embargo, volviendo a la principal motivación para este trabajo, en muchas ocasiones los mensajes no contienen esta información, quedándose este campo sin rellenar. Gracias al estudio de los datos iniciales del anterior proyecto se concluyó que más del treinta por ciento de los buques no lo rellenaban, o lo hacían con información no válida, lo que imposibilita o dificulta la labor de organismos como, en este caso, el COVAM.

La otra conclusión obtenida del estudio inicial realizado es sobre la fiabilidad y veracidad de estos datos. El *MMSI* y el *IMO* son campos propios de cada buque, pero no siempre se completan. Sin embargo, se ha comprobado que aquellos mensajes en los que ambos campos están completos, la calidad del resto de campos es mayor comparada con aquellos en los que no lo están.

3.2.3 Combinación de datos estáticos y dinámicos.

Tras emplear dos tipos de fuentes para el entrenamiento del algoritmo, se procede a crear un tercer conjunto combinando los dos tipos anteriores. Para la creación de este conjunto de datos es necesario un elemento común que permita relacionar los datos estáticos y los dinámicos. Se utiliza el *MMSI* como elemento común en ambos conjuntos de datos, y al ser un dato único de cada buque, consideramos que los datos estáticos y dinámicos relacionados con ese *MMSI* corresponden a un mismo barco.

Dado que previamente hemos procesado los datos estáticos y dinámicos por separado, y los hemos almacenado en sus correspondientes bases de datos con *SQLite3*, para combinar los datos estáticos y dinámicos utilizamos la función *SQL INNER JOIN*.

3.3 Fase de preprocesado y depuración de datos de entrada

En el proceso de recopilación de los datos estáticos y dinámicos se ha detectado que muchos de ellos contienen datos inválidos o erróneos, están incompletos o poseen valores ilógicos, por lo que es necesario realizar un filtrado exhaustivo de estos datos con el fin de entrenar correctamente el sistema y conseguir un conjunto de datos fiable, evitando que el resultado se vea perturbado por datos de mala calidad. Como hemos mencionado, este filtrado se realiza tanto en datos estáticos como en dinámicos, y aunque las condiciones de filtrado son distintas en cada uno de estos grupos, una gran parte del proceso de depuración de datos es muy similar.

3.3.1 Limpieza de datos

El primer paso en el proceso de depuración de los datos es reemplazar los valores infinitos positivos y negativos por el término *NaN (Not a Number)*, para posteriormente eliminar estos datos. También se procede a eliminar aquellas filas de datos incompletas, y se mantienen las filas que tienen valores numéricos válidos.

A continuación, se procede a eliminar aquellas filas con valores duplicados. Esto es necesario para evitar pervertir el modelo, ya que, si entrenamos el algoritmo con una muestra determinada, luego no ponerle a prueba con esa misma muestra.

3.3.2 Valores anómalos

Se ha descubierto que, en algunos casos determinados, ciertos campos se cubren incorrectamente. Por ejemplo, hay mensajes AIS en los que el calado es cero. Por lo tanto, se suprimen las filas con la variable '*draught*'=0. No obstante, es importante recalcar que, aunque sea considerado como un parámetro estático, el calado de los buques varía constantemente en función del peso embarcado en el mismo. Es por esto que se mantendrán aquellas filas que tengan todas sus variables idénticas excepto el calado, que puede cambiar algunos metros dependiendo de la situación. Si estas filas fuesen eliminadas se perdería información real y el modelo se entrenaría con menos datos. En la Figura 3-1 se muestran algunos ejemplos de entradas de datos estáticos, representándose en cada una de las columnas, respectivamente, los valores: MMSI, A, B, C, D, *draught* (calado), y tipo de barco. Se puede observar que, en numerosos casos, el campo 'calado' está relleno con valor 0. Además, este parámetro es muy importante ya que se utiliza en el cálculo de bastantes atributos estáticos, como se muestra en la Figura 3-1, por lo que si empleamos un valor erróneo estamos alterando la predicción del modelo.

9635315	222	59	31	13	0	PETROLERO
9635315	222	59	31	13	118	PETROLERO
9635315	222	59	31	13	119	PETROLERO

9630884	60	15	9	5	0	PETROLERO
9630884	60	15	9	5	36	PETROLERO

9599896	65	65	10	10	0	PASAJEROS
9599896	65	65	10	10	45	PASAJEROS

9644706	20	13	4	5	0	OCIO
9644706	20	13	4	5	30	OCIO

Figura 3-1 Ejemplos de datos estáticos de buques con el campo *draught* = 0 (elaboración propia)

print(df)										
0	mmsi	a	b	c	d	draught	shiptype	len	wid	ldivw \
1	0	0	0	0	0	0	ESTADO	0	0	NaN
2	0	0	0	0	0	0	MERCANTE	0	0	NaN
3	0	0	0	0	0	0	MILITAR	0	0	NaN
4	0	0	0	0	0	0	OCIO	0	0	NaN
...	...	...	...	...	...	...	...	...	...	...
287131	99999999	4	14	3	2	0	PESQUERO	18	5	3.600000
287132	999999999	0	0	0	0	20	PESQUERO	0	0	NaN
287133	999999999	1	19	14	1	25	PASAJEROS	20	15	1.333333
287134	999999999	16	10	3	2	0	ESTADO	26	5	5.200000
287135	999999999	161	44	15	15	109	MILITAR	205	30	6.833333
...	...	...	...	...	...	...	...	...	...	...
0	ldivd	wdivd	area	grith	aml	amt	vs	aol		
1	NaN	NaN	0	0	0	0	0	NaN		
2	NaN	NaN	0	0	0	0	0	NaN		
3	NaN	NaN	0	0	0	0	0	NaN		
4	NaN	NaN	0	0	0	0	0	NaN		
...	...	...	...	...	...	...	...	...		
287131	inf	inf	90	23	0	0	0	0.222222		
287132	0.000000	0.000000	0	0	0	0	0	NaN		
287133	0.800000	0.600000	300	35	500	375	7500	0.050000		
287134	inf	inf	130	31	0	0	0	0.615385		
287135	1.880734	0.275229	6150	235	22345	3270	670350	0.785366		
[287136 rows x 18 columns]										

df = df[df['draught'] != 0]										
print(df)										
9	mmsi	a	b	c	d	draught	shiptype	len	wid	ldivw \
10	0	0	0	0	0	100	REMOLCADOR	0	0	NaN
11	0	0	0	0	0	102	REMOLCADOR	0	0	NaN
12	0	0	0	0	0	106	PESQUERO	0	0	NaN
13	0	0	0	0	0	10	ESTADO	0	0	NaN
...	...	...	...	...	...	...	...	...	...	...
287127	99999999	0	0	19	7	62	MILITAR	0	26	0.000000
287128	99999999	0	0	6	5	58	MILITAR	0	11	0.000000
287132	999999999	0	0	0	0	20	PESQUERO	0	0	NaN
287133	999999999	1	19	14	1	25	PASAJEROS	20	15	1.333333
287135	999999999	161	44	15	15	109	MILITAR	205	30	6.833333
...	...	...	...	...	...	...	...	...	...	...
9	ldivd	wdivd	area	grith	aml	amt	vs	aol		
10	0.000000	0.000000	0	0	0	0	0	NaN		
11	0.000000	0.000000	0	0	0	0	0	NaN		
12	0.000000	0.000000	0	0	0	0	0	NaN		
13	0.000000	0.000000	0	0	0	0	0	NaN		
...	...	...	...	...	...	...	...	...		
287127	0.000000	0.419355	0	26	0	1612	0	NaN		
287128	0.000000	0.189655	0	11	0	638	0	NaN		
287132	0.000000	0.000000	0	0	0	0	0	NaN		
287133	0.800000	0.600000	300	35	500	375	7500	0.050000		
287135	1.880734	0.275229	6150	235	22345	3270	670350	0.785366		
[266940 rows x 18 columns]										

Figura 3-2 Comparativa del tamaño del *dataset* de datos estáticos tras eliminar *draught*=0 (elaboración propia)

En la Figura 3-2 se observa la comparativa entre la cantidad de datos presentes en el experimento antes de eliminar las filas cuyo campo *draught* sea igual a cero y después de eliminarlas. Podemos comprobar que más de veinte mil filas han sido suprimidas debido a esta condición, lo que puede afectar positivamente al resultado de la predicción.

3.3.3 Filtrado de tipos de buque

Por último, basándonos en un documento remitido por el COVAM donde define las situaciones anómalas o de principal interés para labores de vigilancia marítima, eliminamos aquellos buques no prioritarios. Los tipos de buque a predecir con el algoritmo serán mercantes, petroleros, pesqueros, remolcadores, barcos de pasajeros, barcos de ocio, buques de estado, militares y subacuáticos.

3.3.4 Eliminación de outliers

Los *outliers* [23] son valores anómalos, que se escapan del rango en donde se encuentran la mayor parte de las muestras. Estos datos atípicos pueden afectar de manera considerable a los resultados de los distintos ensayos en un modelo de *Machine Learning* como nuestro proyecto. Sin embargo, el término *outlier* no supone una connotación negativa, ya que puede tener varios significados:

- Error: la situación más habitual al encontrarnos con un dato muy destacado es que esté causado por un error en los datos. En una lista con las edades de un conjunto de gente, un valor de 150 años representa un error. Detectar y eliminar los posibles errores es muy importante, ya que los resultados se pueden ver claramente afectados. En caso de necesitar realizar una recta de regresión con el conjunto de datos en donde están incluidos los valores atípicos, vemos cómo puede afectar al resultado en la Figura 3-3, en donde la primera gráfica no elimina los *outliers*, obteniendo una recta con una pendiente mucho mayor a la segunda gráfica, donde sí se eliminan estos valores.

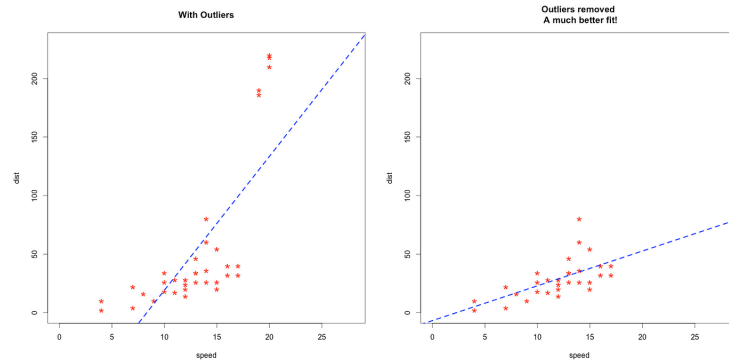


Figura 3-3 Influencia de *outliers* en una recta de regresión [23]

- Límites: comprobado que el dato anómalo no es un error, no se elimina para no afectar a los resultados de la predicción, pudiendo establecer los límites del experimento.
- Agrupación de los datos: representando de manera gráfica los datos podemos observar los posibles conjuntos en los que se agrupan los datos, destacando los más diferenciados, como podemos observar en la Figura 3-4.

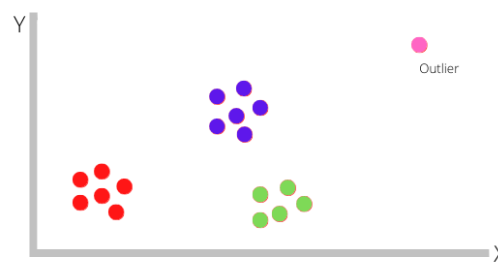


Figura 3-4 Agrupación de datos [23]

- Punto de interés: los *outliers* pueden alterar el transcurso de los experimentos, por lo que su detección es vital. Asimismo, estos datos extremos pueden ser el objetivo a detectar, como ocurre en este proyecto, en el que se trata de modelar el comportamiento “normal” de los barcos, para luego poder detectar como anómalo aquello que se salga de la normalidad.

La representación de los valores permite la detección de anomalías. Estos datos anómalos se pueden clasificar en función del número de variables con las que estén relacionadas. Si únicamente trabajamos con una variable, como en el ejemplo anterior, la edad, y uno de los datos se destaca de manera excepcional, hablaremos de un *outlier* de una sola dimensión, como se puede ver en la Figura 3-5.

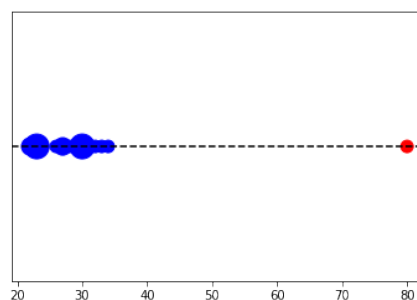


Figura 3-5 *Outlier* de una dimensión [23]

Si los datos a estudiar relacionan dos variables, como puede ser la edad y las ganancias, serán *outliers* de dos dimensiones. En estas gráficas se establece un círculo que delimita el área donde está incluido el bloque de datos, y excluidos los valores anormales. Este círculo, representado en la Figura 3-6, recibe el nombre de frontera.

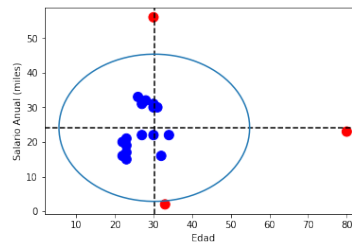


Figura 3-6 *Outliers* de dos dimensiones [23]

Por último, si los datos relacionan tres variables, como pueden ser la edad, las ganancias y las compras mensuales, estaremos ante un valor extremo de tres dimensiones como se puede contemplar en la Figura 3-7.

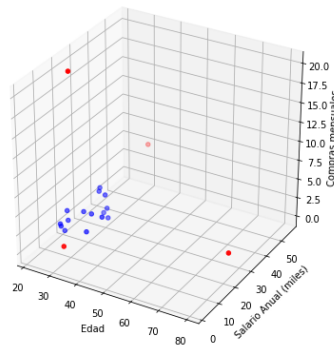


Figura 3-7 *Outliers* de tres dimensiones [23]

No obstante, puede ocurrir el caso en el que un dato analizado respecto a una variable resulte ser un *outlier*, pero al estudiarlo en función de dos o más variables no genere el mismo resultado, y esté incluido en el bloque de datos principal. Debido a esto, es importante conocer el análisis y estudio que se busca realizar sobre los datos.

Una de las maneras de visualizar estos *outliers* con mayor facilidad es a través de las gráficas *boxplot* o diagramas de caja y bigotes [24]. Estos permiten visualizar los valores numéricos y su distribución utilizando el método de los cuartiles, en el que se dividen todos los datos en cuatro grupos con la misma cantidad, delimitados por los valores clave de la gráfica: mínimo, primer cuartil, mediana, tercer cuartil y máximo. En la Figura 3-8 se muestra la representación de un valor anómalo, fuera de cualquiera de estos grupos, fuera de los comúnmente llamados bigotes.

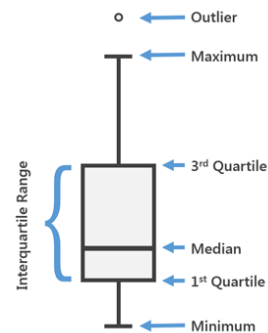


Figura 3-8 Representación *boxplot* [25]

En el diagrama de la Figura 3-8, la caja representa el cincuenta por ciento de los datos, lo que se conoce como el IQR [26] o rango intercuartílico. Este concepto hace referencia a la variabilidad de los datos. Si tenemos un rango intercuartílico grande, los datos estarán muy esparcidos sin concentrarse en una zona o valor aproximado, mientras que si es pequeño ocurrirá lo contrario. En la Figura 3-9 se muestra la relación del diagrama *plotbox* con la gráfica de los datos del experimento. La línea que separa la caja en dos partes es la mediana, por lo que la mitad de los datos serán menores a este valor, mientras que la segunda parte serán superiores.

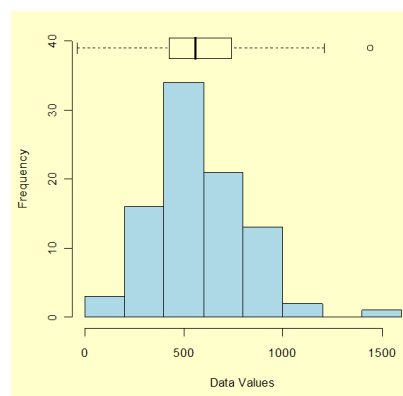


Figura 3-9 Representación gráfica *plotbox* [27]

En este TFG se ha procedido a representar los valores de los datos de cada tipo de buque en diagramas de caja y bigotes con el objetivo de detectar posibles datos atípicos y, posteriormente, proceder a su eliminación.

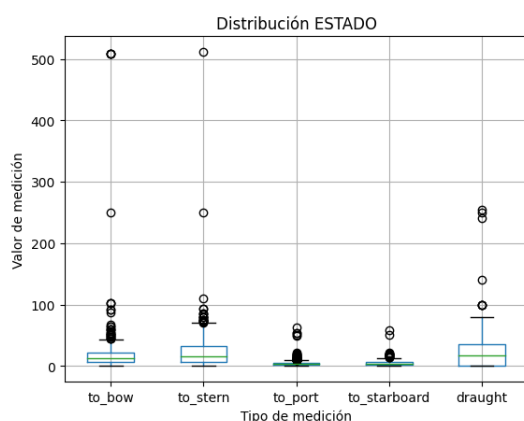


Figura 3-10 Distribución barcos de tipo ESTADO

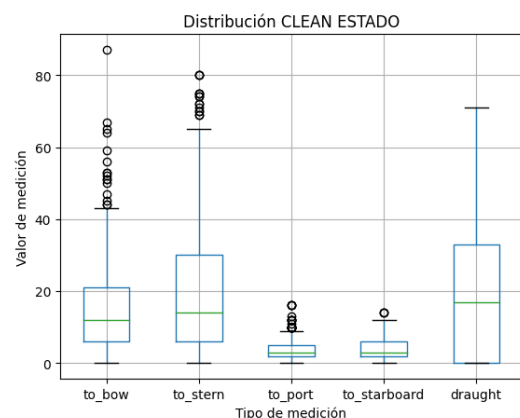


Figura 3-11 Distribución barcos de tipo ESTADO eliminando *outliers*

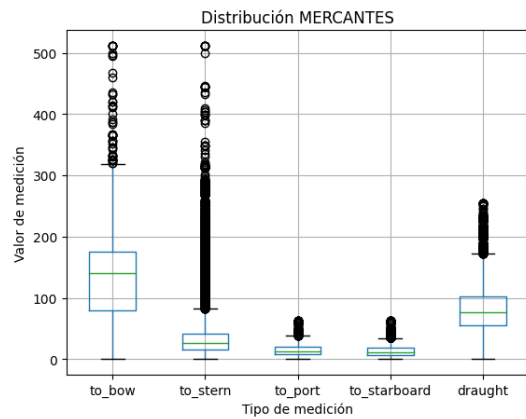


Figura 3-12 Distribución barcos de tipo MERCANTE

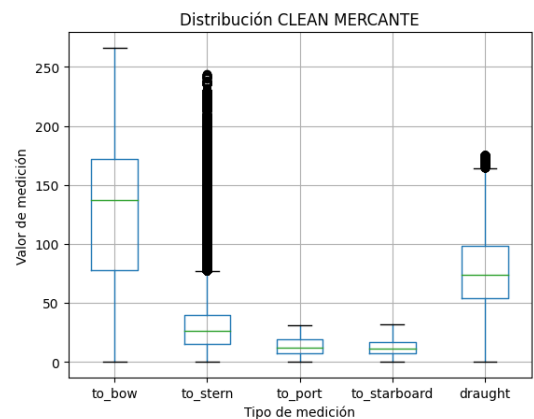


Figura 3-13 Distribución barcos de tipo MERCANTE eliminando outliers

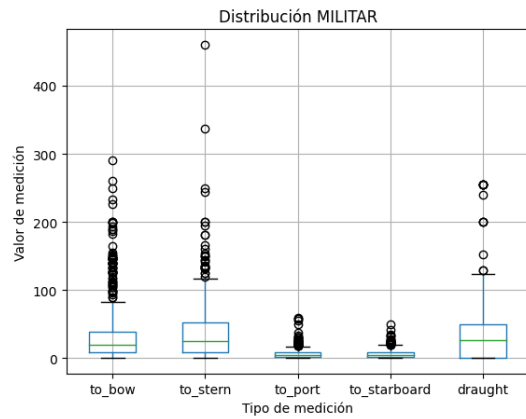


Figura 3-14 Distribución barcos de tipo MILITAR

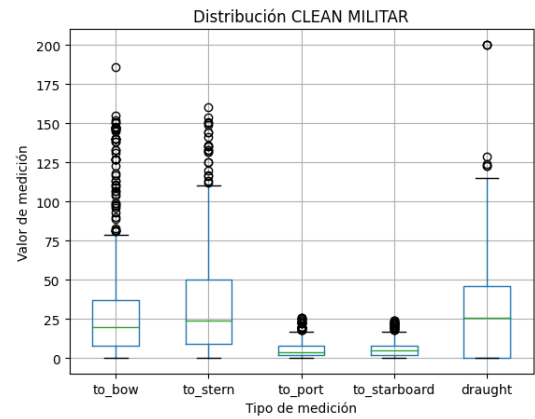


Figura 3-15 Distribución barcos de tipo MILITAR eliminando outliers

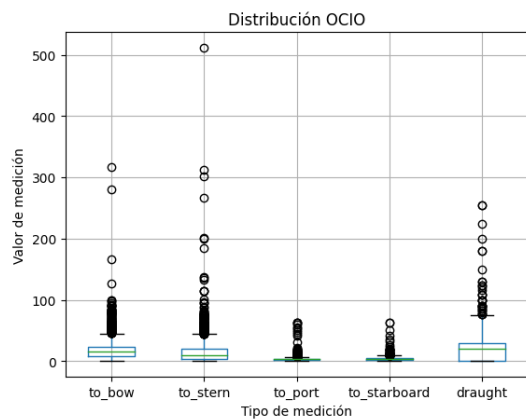


Figura 3-16 Distribución barcos de tipo OCIO

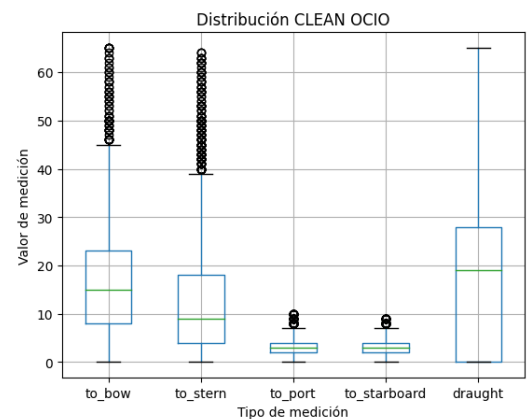


Figura 3-17 Distribución barcos de tipo OCIO eliminando outliers

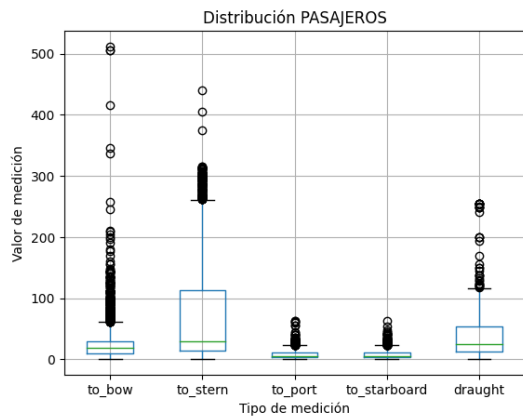


Figura 3-18 Distribución barcos de tipo PASAJEROS

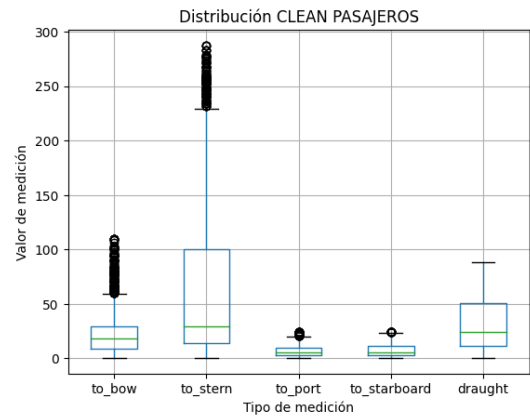


Figura 3-19 Distribución barcos de tipo PASAJEROS eliminando outliers

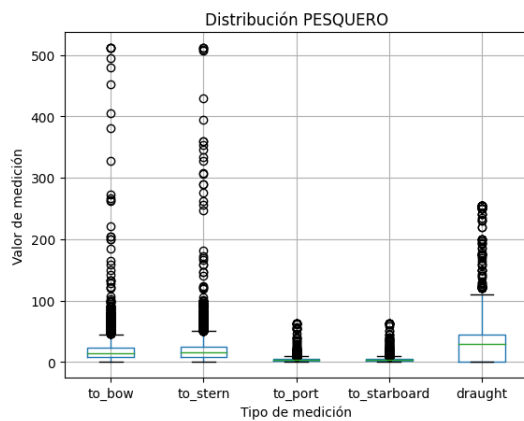


Figura 3-16 Distribución barcos de tipo PESQUERO

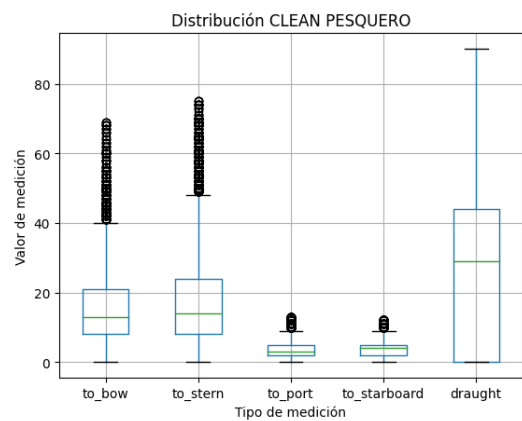


Figura 3-17 Distribución barcos de tipo PESQUERO eliminando outliers

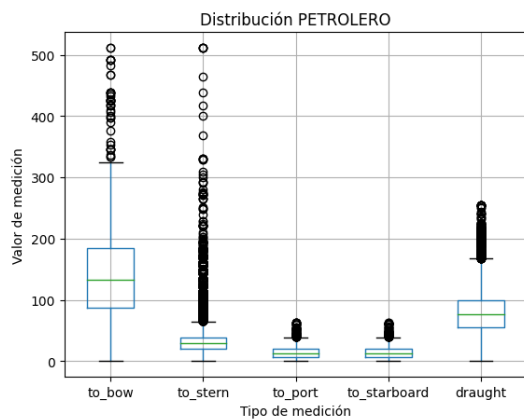


Figura 3-18 Distribución barcos de tipo PETROLEROS

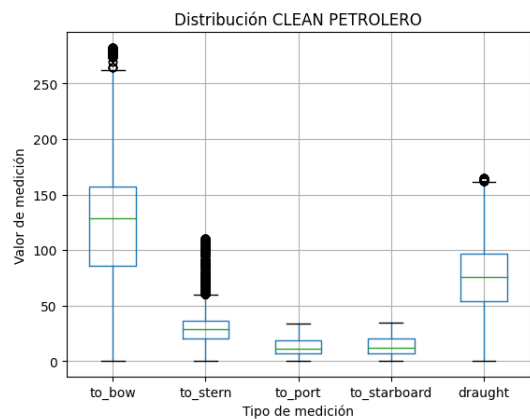


Figura 3-19 Distribución barcos de tipo PETROLEROS eliminando outliers

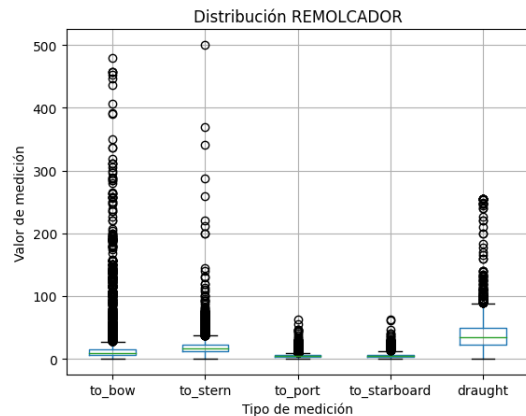


Figura 3-20 Distribución barcos de tipo REMOLCADOR

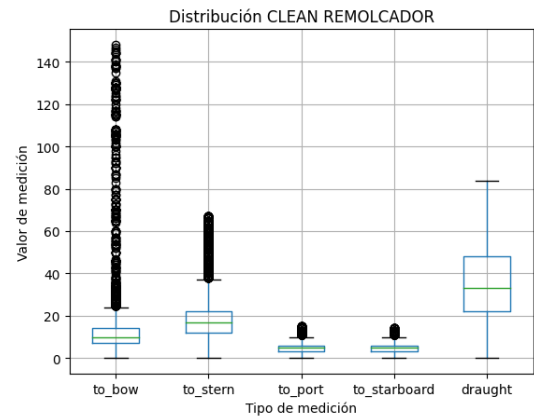


Figura 3-21 Distribución barcos de tipo REMOLCADOR eliminando outliers

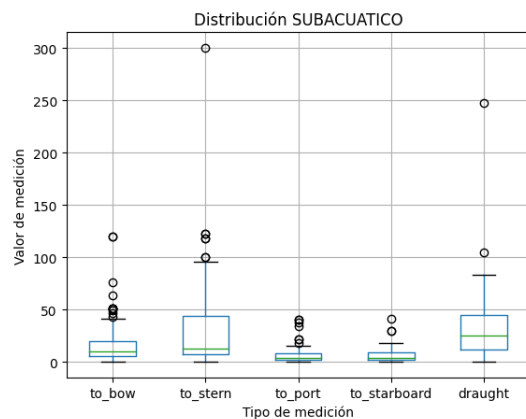


Figura 3-23 Distribución barcos de tipo SUBACUATICO

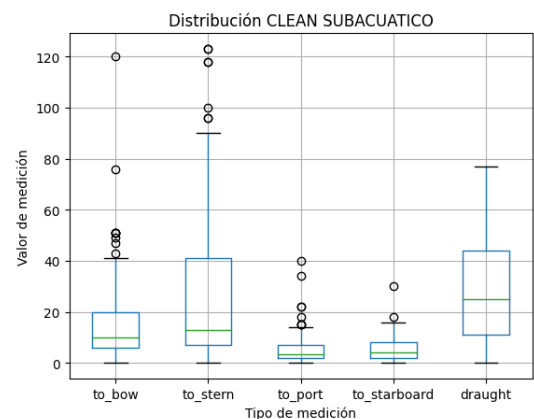


Figura 3-22 Distribución barcos de tipo SUBACUATICO eliminando outliers

Estas gráficas permiten detectar visualmente la existencia de *outliers* en los datos recopilados. Los datos recogen los valores de los atributos estáticos, exceptuando el dato del *MMSI*, y del tipo de barco, ya que es el atributo objetivo de la predicción.

El porcentaje de datos eliminados en este proceso de filtrado viene definido en la función de la Figura 3-24 Función para eliminar *outliers*.

```
#Si queremos eliminar todos los outliers de los parámetros estáticos:
ship_types = ['MERCANTE', 'PETROLERO', 'REMOLCADOR', 'PESQUERO', 'PASAJEROS', 'OCIO', 'ESTADO', 'MILITAR', 'SUBACUATICO']
ship_dimensions = ['a', 'b', 'c', 'd', 'draught']
for type in ship_types:
    for dimension in ship_dimensions:
        q = df[df['shiptype']==type][dimension].quantile(0.99)
        df[(df['shiptype']==type) & (df[dimension] > q)] = None
df_clean=df.dropna()
```

Figura 3-24 Función para eliminar *outliers* (elaboración propia)

En ella se puede observar que la función elimina todos los valores por encima del .99 del cuantil de la dimensión (a, b, c, d, draught) de todos los tipos de barco. Este dato podría cambiar dependiendo de las necesidades y objetivos del estudio.

Desde la Figura 3-10 hasta la Figura 3-22 se representan los datos de cada tipo de barco. Las gráficas de la columna izquierda muestran la distribución de los parámetros estáticos tras la eliminación de valores erróneos e inválidos, mientras que las de la derecha muestran la misma distribución añadiendo la eliminación de los *outliers*.

3.4 Fase de generación de atributos adicionales

Finalizada la fase de preprocesado de los datos, se procede a generar los atributos que serán empleados por el algoritmo para poder realizar la predicción. Al separar los datos iniciales en estáticos y dinámicos, los atributos generados a partir de esa información también corresponderán a los mencionados conjuntos, obteniendo de esta manera atributos estáticos y atributos dinámicos.

3.4.1 Atributos estáticos

En el primer conjunto de datos, creamos los atributos estáticos a partir de 5 parámetros. Estos son A, B, C y D, además del calado del buque.

Las distintas combinaciones de estas variables permiten la creación de los atributos estáticos a utilizar para el algoritmo de clasificación. Para completar este proceso y crear los atributos mencionados se ha definido la función ‘*statics*’, mostrada en la Figura 3-25.

```
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 # Área de la cubierta aproximada= eslora x manga
    df['grith'] = df.len + df.wid # Suma de eslora y manga
    df['aml'] = df.len * df.draught # área longitudinal sumergida
    df['amt'] = df.wid * df.draught # área 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
```

Figura 3-25 Función ‘*statics*’ para crear los atributos mencionados

En la Figura 3-26 se muestran los atributos estáticos creados y que nos permitirán realizar la predicción. Como mencionamos anteriormente, se puede observar que ciertos de estos atributos están directamente relacionados con el campo *draught*, por lo que el filtrado de este tipo de datos es de vital importancia para el correcto desarrollo del modelo.

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

Figura 3-26 Relación de los atributos estáticos calculados a partir de *draught* que alimentan al modelo (elaboración propia)

3.4.2 Atributos dinámicos

Estos atributos son generados a partir de los datos de rumbo, velocidad y posición transmitidas en cada mensaje AIS. Cada uno de esos mensajes incorpora un sello temporal, lo que permite realizar un estudio de cada buque y sus movimientos lo que nos permitirá crear los atributos dinámicos.

Estos atributos dinámicos se crean a partir del cálculo de la cinemática de esos contactos cuyos movimientos realizados se pueden deducir a través de los mensajes AIS transmitidos. Por lo tanto, es fundamental organizarlos temporalmente antes de proceder a la creación de atributos. En la Figura 3-27 se calculan las variables cinemáticas para posteriormente calcular los atributos.

```
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
    #lineal
    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
```

Figura 3-27 Función *cinematics* para la creación de atributos

A continuación, se muestra en la Figura 3-28 los estadísticos empleados en el cálculo de los atributos.

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.

Figura 3-28 Estadísticos que conformarán los atributos dinámicos

3.5 Fase de tratamiento de datos (previo a experimentos)

En los modelos de predicción cuya finalidad es clasificar los datos de entrada en salidas preestablecidas, puede darse la situación en la que dispongamos de cantidades dispares de muestras de las distintas clases de datos. Esto puede significar que las entradas de una clase sean escasas, o insuficientes si se compara con las de otra clase, convirtiéndose en minoritaria. Esto puede generar un posible desbalanceo [28] en el conjunto de datos utilizado en el entrenamiento del modelo.

En nuestro caso se puede apreciar, independientemente del número de días de la muestra o época del año, que el número de entradas de los distintos tipos de barco difieren mucho unos de otros de manera evidente, siendo los mercantes el tipo de barco de mayor frecuencia y, los subacuáticos, los de menor. Sin embargo, no solo puede existir una clase minoritaria, ya que, en nuestro estudio, el menor es el tipo subacuático, pero con entradas muy parecidas a buques de estado y barcos militares.

En la Figura 3-29 se muestra el conjunto de valores del mes de junio y los resultados del experimento. En la gráfica observamos una representación de la cantidad de datos recopilados de cada una de las categorías, donde cabe destacar el dominio de la clase mercantes. En la tabla de la derecha podemos visualizar los datos utilizados para el entrenamiento del algoritmo de cada uno de los tipos de buque y se puede advertir que en aquellos *shiptypes* con pocos datos de entrada los resultados de *precision* y *recall*, y por lo tanto de *F1-score* (estas métricas de evaluación se describirán al inicio del siguiente capítulo) se ven drásticamente perjudicados, lo que afecta a los resultados de la predicción, disminuyendo de esta manera la calidad del modelo.

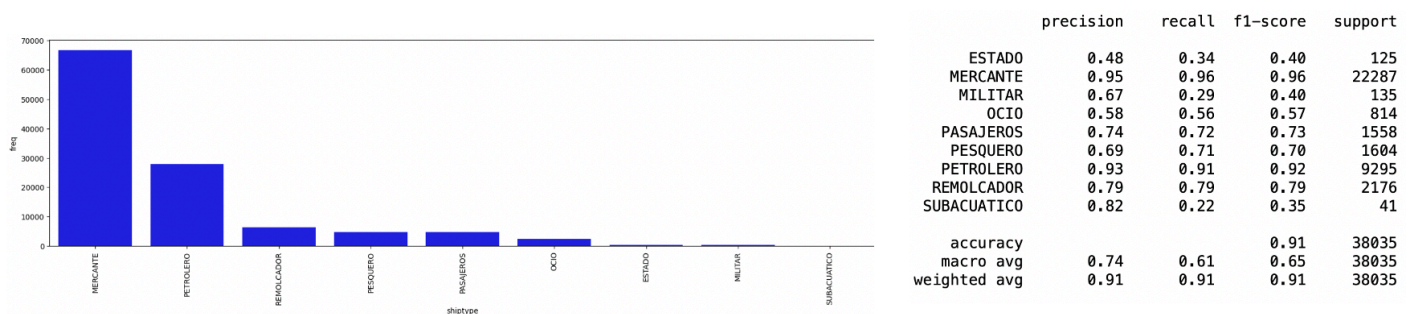


Figura 3-29 Ejemplo de resultados de predicción del tipo de buque con clases desbalanceadas (elaboración propia)

Si al realizar un experimento partimos de un conjunto de datos desequilibrado, los resultados serán altos en precisión para el tipo de barco más habitual, pero bajos en *recall* en las clases minoritarias [29].

Ante este problema, surgen distintas soluciones que pueden amortiguar los efectos de la aparición de clases minoritarias. El empleo de técnicas [30] como el *oversampling* o *undersampling* pueden resultar ventajosas para mantener la calidad del modelo, y su uso depende exclusivamente de los objetivos del estudio y las condiciones del análisis. En la Figura 3-30 se puede observar la comparativa entre ambas técnicas [31] [32].

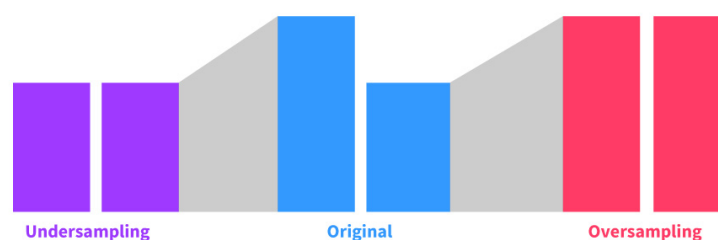


Figura 3-30 Comparativa *undersampling* vs *oversampling* [29]

3.5.1 OverSampling

La primera de las soluciones consiste en crear, de manera artificial, datos sintéticos en aquellas clases minoritarias hasta alcanzar en todos los tipos de buque la misma cantidad de entradas, siendo esta cifra objetivo el número de datos de la clase con mayor frecuencia. Esta técnica es utilizada en situaciones en la que los datos de una de las clases no son suficientes para proceder con el experimento.

3.5.2 UnderSampling

Por el contrario, la técnica de *undersampling* reduce las entradas de todas las clases presentes en el modelo hasta igualar el número de datos al de la clase minoritaria con el objetivo de conseguir un conjunto de datos balanceado. Para ello, elimina aleatoriamente datos de los diferentes tipos de barcos hasta igualarse con el de menor frecuencia. El principal inconveniente de este recurso es la eliminación de un gran número de muestras útiles para el modelo, por lo que será necesario considerar la información perdida en el proceso [33].

3.5.3 MinMaxScaler

Esta función [34] de la librería *Sklearn* realiza una normalización de los datos, habitualmente dentro del rango 0 y 1, manteniendo la proporcionalidad de los datos entre los márgenes establecidos. De esta manera, todos los valores originales se transforman en nuevos datos normalizados. El valor máximo se convierte en 1 y el mínimo en 0. Al mantener la proporcionalidad, se mantienen las características del conjunto de datos y los resultados no se ven alterados.

Esta técnica suele ser utilizada en procesos de *Machine Learning* cuando se trabaja con conjuntos de datos con diferentes características, unas con rangos de valores grandes y otras con un rango mucho menor. Emplear esta función facilita la comparación entre estos conjuntos al adaptarlos a un mismo rango de valores, manteniendo la distribución de los datos [35]. Además, puede mejorar el rendimiento del algoritmo, ya que facilita la tarea de comparar atributos con datos comprendidos entre los mismos valores.

3.6 Fase de experimentación

En este apartado se indican los experimentos realizados con los datos estáticos, los dinámicos y la combinación de ambos. Estos se han realizado empleando fundamentalmente tres *scripts* diferentes, correspondiendo a los conjuntos mencionados, en los que se trabaja en el preprocesado de datos, el entrenamiento del algoritmo y la predicción.

3.6.1 Experimentos con datos estáticos

La parte de experimentos con datos estáticos es, probablemente, la de mayor importancia, ya que es en la que hemos realizado un mayor número de ensayos y donde se estudian posibles variaciones aplicables al resto de *scripts*. Esto es debido a los resultados obtenidos en el trabajo que vamos a validar, donde quedó reflejada la gran importancia de los datos estáticos para la correcta predicción del tipo de buque. En este tipo de pruebas estudiaremos los siguientes campos:

1. Volumen de datos: el principal inconveniente del trabajo anterior es la falta de experimentación o pruebas que permitan validar el proyecto. En nuestro caso contamos con un volumen de datos mucho mayor debido a la constante recopilación de mensajes AIS. Esto conlleva ciertas ventajas, como pueden ser realizar pruebas con datos de diferentes épocas del año, con distinto tamaño del conjunto de datos para comprobar si el comportamiento se mantiene con más entradas o emplear nuevas técnicas de estudio.

2. OS (*oversampling*): realizaremos pruebas con esta técnica. En caso de no utilizar esta técnica, irá marcada con un 'NO'. Si se aplica, en esta columna se dará información de a qué tipos de datos se aplica, es decir, *train*, *test* o ambos.
3. US (*undersampling*): mismo concepto que OS.
4. N° arb: esta columna indica el número de árboles utilizados en el algoritmo *Random Forest*.
5. De/Dt: porcentaje de datos utilizados para la parte de entrenamiento y para la parte de test.
6. N° de tipos de barco: en muchos experimentos comprobaremos el comportamiento del modelo variando el número de tipos de buque, observando cómo afectan a los más representativos. Esta columna indicará el número de tipos de barco estudiados en ese experimento.

Los primeros experimentos servirán para corroborar las elecciones realizadas durante el trabajo anterior, y comprobar las posibles variaciones en los resultados con diferentes datos. Partiendo del *dataset* original utilizado en el proyecto anterior, con datos procedentes de un periodo de catorce días, realizamos los experimentos mostrados en Tabla 3-1:

N° exp.	Datos	OS	US	N° arb	De / Dt	N° tipos de barco	Observaciones
1	Dataset original (14 días)	NO	NO	100	0,75/0,25	9	-
2	Dataset original	Train	NO	100	0,75/0,25	9	Experimento original
3	Dataset original	Test	NO	100	0,75/0,25	9	-
4	Dataset original	Train y test	NO	100	0,75/0,25	9	-
5	Dataset original	NO	Train	100	0,75/0,25	9	-

Tabla 3-1 Experimentos base

Realizamos los experimentos básicos con el *dataset* original, manteniendo los ajustes seleccionados en el proyecto anterior, simplemente para comprobar que los resultados obtenidos coinciden con los del trabajo anterior. A partir de aquí, estudiaremos posibles variaciones para determinar cuál de los métodos de la Tabla 3-1 dispone de las mejores características para proporcionar mejores resultados. Utilizamos las técnicas mencionadas anteriormente, como *oversampling* y *undersampling*.

En la Tabla 3-2, mostramos los experimentos realizados con algunas variaciones respecto a los anteriores. Con el objetivo de validar los ajustes seleccionados, realizamos pruebas variando esos datos en cuestión y observamos los resultados. En este caso, cambiamos el número de árboles para encontrar el mayor rendimiento entre la precisión resultante y el tiempo de ejecución del modelo, y la relación de entre datos de entrenamiento y datos test. Estas variaciones se han realizado sobre el experimento cuatro, lo que no implica que sea el seleccionado para continuar con los ensayos.

Nº exp.	Datos	OS	US	Nº arb	De / Dt	Nº tipos de barco	Observaciones
6	Dataset original	Train y test	NO	200	0,75/0,25	7	Variación nº arb
7	Dataset original	Train y test	NO	100	0,65/0,35	7	Variación De/Dt
8	Dataset original	Train y test	NO	100	0,85/0,15	7	Variación De/Dt

Tabla 3-2 Experimentos realizando ajustes en el número de árboles y la relación De/Dt

La Tabla 3-3 Experimentos con los tipos de buque más representativos muestra los experimentos realizados reduciendo los tipos de buque a los cinco más representativos. A continuación, se procede a realizar un estudio para elegir cuál de los experimentos base (del uno al cinco) es el adecuado para el proyecto. Para ello realizaremos una serie de ensayos con dos *datasets* diferentes con entradas de catorce días. El número de árboles y la repartición de datos entre entrenamiento y test se mantendrá igual en todas las variaciones: nº arb = 100 y De/Dt = 0,75/0,25. La Tabla 3-4 muestra los experimentos realizados.

Nº exp.	Datos	OS	US	Nº arb	De / Dt	Nº tipos de barco	Observaciones
9	Dataset original	Train y test	NO	100	0,75/0,25	5	Tipos de barco más representativos con OS
10	Dataset original	NO	NO	100	0,75/0,25	5	Tipos de barco más representativos sin OS

Tabla 3-3 Experimentos con los tipos de buque más representativos

Nº exp.	Datos	OS	US	Nº tipos de barco	Observaciones
11	Recopilación CSVs (14 días)	NO	NO	9	-
12	Recopilación CSVs	Train	NO	9	Experimento original con diferentes datos
13	Recopilación CSVs	NO	NO	5	-
14	Recopilación CSVs	Train	NO	5	-
15	Recopilación CSVs	Train y test	NO	5	-
16	Recopilación CSVs	NO	Train	5	-

17	Recopilación CSVs	NO	Train y test	5	-
----	-------------------	----	--------------	---	---

Tabla 3-4 Experimentos con datos estáticos de 14 días de igual tamaño

Procedemos a realizar la misma serie de experimentos con otro conjunto distinto de datos, pero de otro momento temporal. La Tabla 3-5 muestra los experimentos realizados.

Nº exp.	Datos	OS	US	Nº tipos de barco	Observaciones
18	Base datos Statics (14 días)	NO	NO	9	-
19	Base datos Statics	Train	NO	9	Experimento original con diferentes datos
20	Base datos Statics	NO	NO	5	-
21	Base datos Statics	Train	NO	5	-
22	Base datos Statics	Train y test	NO	5	-
23	Base datos Statics	NO	Train	5	-
24	Base datos Statics	NO	Train y test	5	-

Tabla 3-5 Experimentos con otro conjunto de datos estáticos de 14 días

Una vez seleccionado el experimento que proporciona mejor resultado y optimiza la calidad del modelo, procedemos a comprobar que estos resultados obtenidos se mantienen en diferentes momentos, analizando los datos recopilados durante los meses de junio y enero, cumpliendo con el requisito de estar suficientemente distanciados en el tiempo para evitar cualquier parecido en los datos. En la Tabla 3-6 se muestran las pruebas realizadas con los *datasets* de ambos meses.

Nº exp.	Datos	OS	US	Nº tipos de barco	Observaciones
25	Junio 2022	NO	NO	9	-
26	Junio 2022	NO	NO	5	-
27	Enero 2023	NO	NO	9	-
28	Enero 2023	NO	NO	5	-

Tabla 3-6 Experimentos con datos estáticos de los meses de enero y junio

Una vez se ha comprobado la similitud de los resultados del modelo en diferentes momentos temporales, durante un periodo de tiempo considerablemente prolongado, descartamos la idea de que la predicción pueda estar alterada de manera significativa debido a las distintas épocas del año.

Analizando el método de separación de los datos antes de alimentar al algoritmo de *Random Forest* se observa que, con el método actual, es decir, como se ha ejecutado hasta ahora, podemos separar entradas con iguales *MMSI* y enviarlas tanto a la parte de entrenamiento como a la de test. Esto significa que de una manera no voluntaria falseamos el modelo obteniendo mejores resultados, ya que examinamos al algoritmo con una entrada con la que se ha entrenado y, por lo tanto, ya conoce. De esta manera no puede fallar en esa predicción.

Esto ocurre debido a que hay entradas estáticas con el mismo *MMSI*, pero que no tienen toda la fila de campos rellena con los mismos datos. Esta situación es habitual, ya que el dato del calado puede variar según la situación en la que se encuentre el buque, como se muestra en la Figura 3-31. También puede darse el caso en el que el buque transmisor cambie la posición de la antena AIS entre la transmisión de distintos mensajes, de modo que las distancias establecidas desde las partes características de un buque hasta el transmisor se vean alteradas. Sin embargo, esta opción no es tan común.

En resumen, en el trabajo previo y en los experimentos anteriores el algoritmo se entrenaba con datos estáticos de un buque y después se probaba con datos estáticos del mismo buque, variando tan solo el campo de *draught*, por lo que en cierta medida pervertíamos el modelo.

9635315	222	59	31	13	0	PETROLERO	9732228	199	40	24	18	126	PETROLERO
9635315	222	59	31	13	118	PETROLERO	9732228	199	40	24	18	130	PETROLERO
9635315	222	59	31	13	119	PETROLERO	9732228	199	40	24	18	132	PETROLERO
9599901	160	30	12	20	60	MERCANTE	9732228	199	40	24	18	133	PETROLERO
9599901	160	30	12	20	68	MERCANTE	9732228	199	40	24	18	136	PETROLERO
9599901	160	30	12	20	76	MERCANTE	9732228	199	40	24	18	140	PETROLERO
							9732228	199	40	24	18	144	PETROLERO

Figura 3-31 Ejemplos variaciones en el calado con mismo *MMSI* (elaboración propia)

Para evitar esto establecemos un método de separación que garantiza que todas las entradas con un *MMSI* vayan o bien a los datos de entrenamiento o bien a los de test, pero todas al mismo. Este procedimiento consiste en, a partir de un *dataset* original, crear otro con únicamente la columna de *MMSI*, denominado 'df0'. En este momento se eliminan los datos repetidos (entradas con este dato repetido) para a continuación hacer la separación entre entrenamiento (df_tr) y test (df_ts). El siguiente paso es utilizar la función *merge* de la biblioteca de pandas, que nos permite unir *dataframes* usando un elemento en común como clave para la combinación, por lo que uniremos df_tr y df_ts con el dataset original de manera separada, obteniendo df_train y df_test respectivamente. Esta función relaciona el dato común de todas las entradas de ambos *datasets*, por lo que al unir el archivo df0 con el original, recuperamos todas las entradas de ese *MMSI* en una de las partes (entrenamiento o test).

Asimismo, como se explica en el apartado 3.3.2, se procede a eliminar aquellos datos anómalos que pueden alterar negativamente los resultados del modelo. Por lo tanto, suprimimos las filas con el campo *draught* igual a cero.

Nº exp.	Datos	OS	US	Nº tipos de barco	Observaciones
29	Junio 2022	NO	NO	9	Se realiza la separación correcta de los datos y se eliminan filas con <i>draught</i> =0
30	Junio 2022	NO	NO	5	Se realiza la separación correcta de los datos y se eliminan filas con <i>draught</i> =0
31	Septiembre 2022	NO	NO	9	Se realiza la separación correcta de los datos y se eliminan filas con <i>draught</i> =0
32	Septiembre 2022	NO	NO	5	Se realiza la separación correcta de los datos y se eliminan filas con <i>draught</i> =0
33	Enero 2023	NO	NO	9	Se realiza la separación correcta de los datos y se eliminan filas con <i>draught</i> =0
33	Enero 2023	NO	NO	5	Se realiza la separación correcta de los datos y se eliminan filas con <i>draught</i> =0

Tabla 3-7 Experimentos con separación correcta con datos de entrenamiento y test, y *draught*=0

Se puede observar en la Tabla 3-7 cómo alternamos el número de tipos de buque, eliminamos los valores anómalos y realizamos una separación correcta de los datos. También se ha incorporado el mes de septiembre al conjunto de experimentos.

Finalizamos los experimentos de los datos estáticos eliminando los valores atípicos o *outliers*. Los experimentos realizados se muestran en la Tabla 3-8.

Nº exp.	Datos	OS	US	Nº tipos de barco	Observaciones
34	Junio 2022	NO	NO	9	<i>Outliers</i> eliminados
35	Junio 2022	NO	NO	5	<i>Outliers</i> eliminados

Tabla 3-8 Experimentos de datos estáticos del mes de junio 2022 eliminando *outliers*

3.6.2 Experimentos datos dinámicos

Los experimentos realizados con los datos dinámicos son menos que los estáticos, ya que partimos de las conclusiones obtenidas en el apartado anterior. Además, en este caso no tenemos la casuística de que los mismos datos se utilicen en el conjunto de entrenamiento y test, ya que las cinemáticas son siempre diferentes. Se eliminan los valores anómalos y se suprimen los *outliers* que puedan afectar a los resultados de las predicciones.

Al trabajar con este conjunto de datos, estudiaremos los resultados obtenidos, visualizando la importancia de los diferentes datos en gráficas. Uno de esos datos es el *LOA*, correspondiente a la eslora del buque. Este dato estático tiene una gran influencia en la creación de los atributos dinámicos, por lo que su presencia en el proceso de predicción mejora de forma considerable los resultados obtenidos. De esta manera procedemos a realizar los experimentos mostrados en la Tabla 3-9:

Nº exp.	Datos	Incluye LOA	Nº tipos de barco	Observaciones
38	01/02/23 (1 día)	SI	9	-
39	01/02/23	NO	9	-
40	01/02/23	SI	5	-
41	01/02/23	NO	5	-
42	01/02/23 y 02/02/23 (2 días)	SI	9	-
43	01/02/23 y 02/02/23	NO	9	-
44	01/02/23 y 02/02/23	SI	5	-
45	01/02/23 y 02/02/23	NO	5	-
46	01/02/23 - 09/02/23 (9 días)	SI	9	-
47	01/02/23 - 09/02/23	NO	9	-
48	01/02/23 - 09/02/23	SI	5	-
49	01/02/23 - 09/02/23	NO	5	-

Tabla 3-9 Experimentos con datos dinámicos

4 RESULTADOS OBTENIDOS

Mencionados en el capítulo anterior los experimentos a realizar con los distintos conjuntos de datos, se procede a exponer los resultados de los mismos, indicando las conclusiones obtenidas tras su análisis para cada uno de los conjuntos. Antes de comenzar el estudio de los experimentos, procederemos a explicar las métricas utilizadas para evaluar los resultados y que indican la calidad de la predicción.

Para la mejor comprensión y redacción del capítulo, nos referiremos a los experimentos con el número indicado en las tablas de los apartados 3.6.1 y 3.6.2.

4.1 Calidad del modelo

Para la comprobación de los resultados del modelo utilizaremos la función *classification_report* de la librería *Sklearn*, que proporciona unos parámetros con los que se evalúan los numerosos ensayos realizados. La función mencionada proporciona información sobre la calidad de un modelo de clasificación a través del empleo de varios parámetros entre los que se encuentran *precision*, *recall* y *F1-score* [36]. La representación de estos parámetros se realiza a través de la matriz de confusión, mostrada en la Figura 4-1. Esta permite analizar el modelo de clasificación, y relaciona las predicciones realizadas con los resultados que debería haber mostrado.

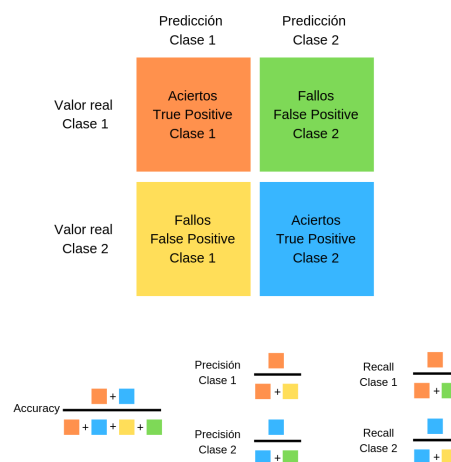


Figura 4-1 Matriz de confusión en un algoritmo de clasificación con datos desbalanceados [29]

Para comprender esta matriz es necesario definir los términos que emplea [37]:

- TP (*true positive*): valor positivo clasificado como positivo.

- FP (*false positive*): valor negativo clasificado como positivo.
- TN (*true negative*): valor negativo clasificado como negativo.
- FN (*false negative*): valor positivo clasificado como negativo.

La precisión es la proporción de verdaderos positivos (TP) sobre el total de predicciones positivas (TP + FP). Esta mide cuántas de las predicciones positivas realizadas por el modelo son verdaderas. Si la precisión es alta (número ligeramente inferior o igual a 1), podemos afirmar que el modelo hace pocas predicciones falsas positivas.

El *recall*, también conocido como sensibilidad, es la relación entre verdaderos positivos (TP) y el total de verdaderos positivos y falsos negativos (TP + FN). De esta manera medimos cuántas de las muestras positivas del conjunto de datos se identifican correctamente. Un *recall* alto indica que el modelo hace pocas predicciones falsas negativas.

El *F1-score* es una medida combinada de precisión y *recall*, que se calcula como la media armónica de ambos. Este se utiliza para evaluar la precisión y el *recall* de manera conjunta, proporcionando una medida única de la calidad del modelo. Un *F1-score* alto indica que el modelo tiene una precisión alta y un *recall* alto.

Además de estos parámetros mencionados, el informe generado por la función añade tanto una media aritmética de las métricas de todas las clases, y una media ponderada que tiene en cuenta la cantidad de entradas procedentes de cada tipo de buque. Sin embargo, a la hora de analizar los resultados priorizaremos la precisión, el *recall* y el *F1-score* debido al gran desbalanceo entre las distintas clases, haciendo que la media ponderada no sea un dato tan representativo.

4.2 Resultados estáticos

Comenzando con el análisis de los resultados, analizamos la primera serie de experimentos reflejados en la Tabla 3-1, en la que replicamos el ensayo original. En el TFG previo, este experimento establecía el ajuste definitivo de sus parámetros. Una vez realizado modificamos estos parámetros indicados para corroborar las decisiones tomadas, y su validación.

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 original TFG previo

	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	0.76	0.68	587
PleasureCraft	0.57	0.57	0.57	282
Tanker	0.88	0.90	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 2

Como se puede observar, somos capaces de replicar el experimento obteniendo los mismos resultados, ya que se aprecian variaciones muy pequeñas en algunas clases.

Sin embargo, al comparar los resultados del experimento original, que conlleva el empleo de la técnica de *oversampling*, con el resto de ensayos de la Tabla 3-1, se puede observar que el uso de esta técnica genera peores resultados, incrementándose el valor de *F1-score* en el experimento 1. Esto se debe a que el sobremuestreo afecta de manera significativa a las clases minoritarias.

	precision	recall	f1-score	support
Cargo	0.93	0.96	0.95	8930
Fishing	0.79	0.80	0.79	1186
MilitaryOps	0.78	0.32	0.46	90
Passenger	0.70	0.72	0.71	587
PleasureCraft	0.62	0.54	0.58	282
Tanker	0.92	0.87	0.89	3268
Tug	0.83	0.82	0.82	1107
accuracy			0.90	15450
macro avg	0.80	0.72	0.74	15450
weighted avg	0.90	0.90	0.90	15450

Experimento 1

En el experimento 1 el recall disminuye en las clases con menos entradas. Sin embargo, la precisión aumenta en gran medida en estas categorías generando de esta manera un incremento en el valor de F1-score. Por lo tanto, analizando estos resultados concluimos que el oversampling favorece la identificación de estas clases minoritarias al aumentar el recall, pero como disminuye la precisión en mayor medida que aumenta el recall, deberá realizar más predicciones falsas de esta clase. Si a esta situación le añadimos la desventaja de que al utilizar sobremuestreo creamos datos ficticios de las clases minoritarias haciendo que el modelo de predicción no se ajuste a la realidad, procedemos a descartar el experimento original y desarrollar el algoritmo evitando el empleo de oversampling.

En la Tabla 3-2 realizamos ciertas variaciones en los campos de número de árboles y porcentaje de datos de entrenamiento y test. Los resultados obtenidos son prácticamente iguales, observando únicamente variación de décimas en la columna de precisión en algunas categorías de barcos. Sin embargo, el proceso es mucho más lento y requiere más tiempo de ejecución a la hora de obtener estos resultados, por lo que como ya estaba establecido anteriormente, el número de árboles se mantendrá en 100 y el porcentaje de datos de entrenamiento y test seguirá siendo de 0,75/0,25.

En la Tabla 3-3, introducimos la idea de trabajar con las clases más representativas. De esta manera suprimimos las clases minoritarias observando cómo afecta a los resultados. Los tipos de buque con más entradas son mercante, pasajeros, pesqueros, petroleros y remolcadores. Aunque la técnica de *oversampling* ha sido descartada, realizamos este experimento utilizándola para comprobar su comportamiento con las clases más abundantes.

	precision	recall	f1-score	support
Cargo	0.87	0.93	0.90	8893
Fishing	0.83	0.84	0.83	8893
Passenger	0.89	0.82	0.85	8893
Tanker	0.93	0.90	0.91	8893
Tug	0.83	0.85	0.84	8893
accuracy			0.87	44465
macro avg	0.87	0.87	0.87	44465
weighted avg	0.87	0.87	0.87	44465

Experimento 9: 5 más frecuentes con OS

Sin embargo, la técnica de sobremuestreo consiste en crear datos ficticios en aquellos tipos de barcos con menor número de entradas para poder realizar la medición. Es por esto que, si quitamos aquellos tipos de barcos con menos entradas, puede perder un poco de sentido el utilizar esta técnica, por lo que también realizaremos el experimento sin utilizarla.

	precision	recall	f1-score	support
Cargo	0.94	0.95	0.95	8893
Fishing	0.80	0.82	0.81	1161
Passenger	0.78	0.73	0.76	639
Tanker	0.91	0.87	0.89	3296
Tug	0.80	0.81	0.81	1081
accuracy			0.90	15070
macro avg	0.85	0.84	0.84	15070
weighted avg	0.90	0.90	0.90	15070

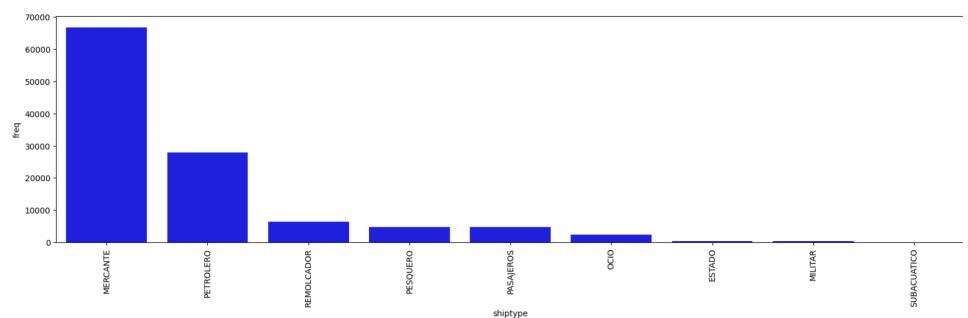
Experimento 10: 5 más frecuentes sin OS

Los resultados son parejos, mejoran ligeramente sin usar la técnica de sobremuestreo. Así evitaríamos la posible contaminación de los datos utilizando la técnica de *oversampling*.

En la Tabla 3-4 y en la Tabla 3-5 se realizan los mismos experimentos que en la Tabla 3-1 con conjuntos de datos diferentes, del mismo tamaño, correspondientes periodos de 14 días.

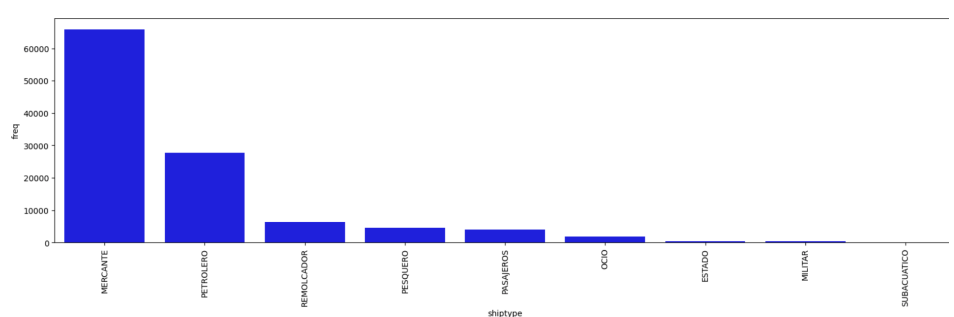
En la Tabla 3-6, finalizados los experimentos con las técnicas de *oversampling* y *undersampling*, habiendo seleccionado los ajustes establecidos para obtener los mejores resultados, se procede a comprobar que los resultados obtenidos se mantienen en diferentes épocas del año. Para ello emplearemos los datos estáticos recopilados de los meses de junio de 2022 y enero de 2023, momentos suficientemente distanciados para evitar cualquier similitud en los datos, o para que al menos, en caso de similitud de los *datasets*, no sea este el motivo.

	precision	recall	f1-score	support
ESTADO	0.48	0.34	0.40	125
MERCANTE	0.95	0.96	0.96	22287
MILITAR	0.67	0.29	0.40	135
OCIO	0.58	0.56	0.57	814
PASAJEROS	0.74	0.72	0.73	1558
PESQUERO	0.69	0.71	0.70	1604
PETROLERO	0.93	0.91	0.92	9295
REMOLCADOR	0.79	0.79	0.79	2176
SUBACUATICO	0.82	0.22	0.35	41
accuracy			0.91	38035
macro avg	0.74	0.61	0.65	38035
weighted avg	0.91	0.91	0.91	38035



Experimento 25

	precision	recall	f1-score	support
ESTADO	0.57	0.37	0.45	134
MERCANTE	0.94	0.96	0.95	21951
MILITAR	0.68	0.30	0.42	113
OCIO	0.56	0.51	0.53	620
PASAJEROS	0.74	0.73	0.74	1388
PESQUERO	0.70	0.73	0.71	1485
PETROLERO	0.93	0.90	0.91	9302
REMOLCADOR	0.79	0.81	0.80	2098
SUBACUATICO	0.50	0.17	0.25	42
accuracy			0.91	37133
macro avg	0.71	0.61	0.64	37133
weighted avg	0.90	0.91	0.90	37133



Experimento 27

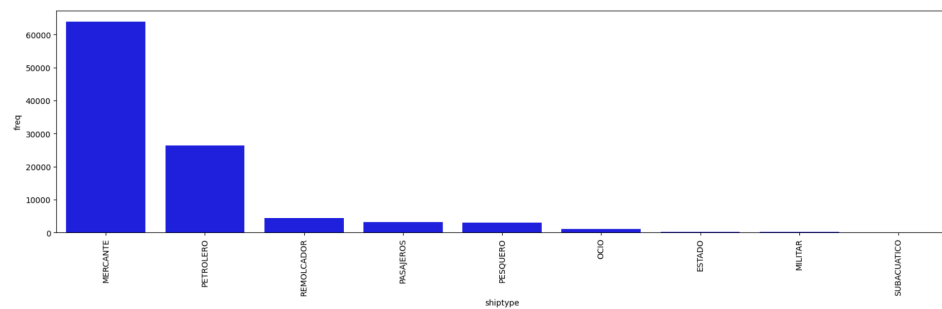
Como podemos observar en ambos experimentos, los resultados utilizando datos estáticos de otras épocas del año se mantienen constantes, por lo que se descarta la idea de que la predicción pueda ser alterada debido a las distintas épocas del año.

En la Tabla 3-7 se realizan experimentos añadiendo la principal implementación de este trabajo respecto al TFG previo. Esta consiste en la correcta separación de los datos de entrenamiento y de test a partir del *MMSI*. De esta manera evitamos falsear el modelo, ya que este estaba siendo examinado con algunas entradas con las que se había entrenado, y conseguimos resultados más realistas.

Además, estos experimentos incluyen otro avance, al realizarse con conjuntos de datos en los que se han eliminado los posibles valores anómalos, como es el caso del campo ‘draught’, que en varias entradas estaba completo con el valor 0.

Por lo tanto, los experimentos de la Tabla 3-7 han supuesto un avance en la fase de preprocesado y depuración de los datos. Se han realizado los experimentos con los conjuntos de datos procedentes de los meses de junio y septiembre de 2022 y enero de 2023.

	precision	recall	f1-score	support
ESTADO	0.25	0.18	0.21	322
MERCANTE	0.87	0.88	0.87	25131
MILITAR	0.34	0.07	0.11	378
OCIO	0.48	0.31	0.38	2114
PASAJEROS	0.56	0.57	0.56	3102
PESQUERO	0.51	0.65	0.58	3473
PETROLERO	0.78	0.73	0.75	10886
REMOLCADOR	0.64	0.73	0.68	4149
SUBACUÁTICO	0.06	0.01	0.02	92
accuracy			0.76	49647
macro avg	0.50	0.46	0.46	49647
weighted avg	0.76	0.76	0.76	49647

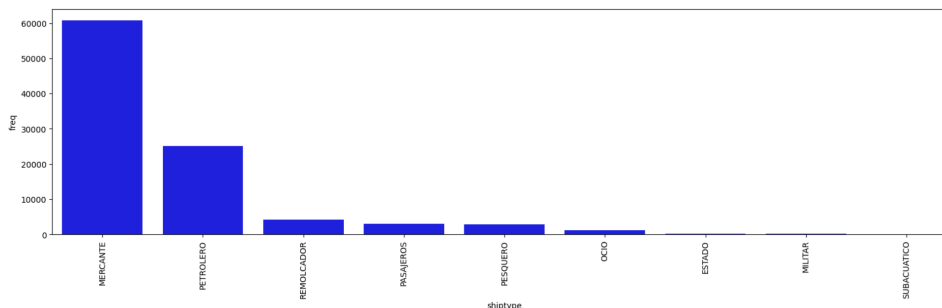


Experimento 29

Se puede observar que tras realizar estos experimentos (ocurre lo mismo en los meses de septiembre y enero), los resultados no solo no mejoran, sino que empeoran. Esto ocurre debido a la previamente mencionada ‘correcta separación de los datos’ para el modelo, donde se subsana el error de realizar las predicciones con posibles datos ya conocidos (falseados). Por lo tanto, los resultados empeoran, pero el modelo mejora, ya que se ciñe más a la realidad. La veracidad de la predicción es imperativa.

Para finalizar con los experimentos de datos estáticos, procedemos a eliminar aquellos datos extremos o atípicos, tal como se refleja en la Tabla 3-8, que pueden alterar los resultados de las predicciones. Estos son los *outliers* previamente explicados. De cara a este experimento final mantenemos el conjunto de datos de junio, de manera que se pueda observar la variación entre el conjunto de datos de entrada del experimento 29, que no elimina los *outliers*, con el conjunto de datos del experimento 34, en donde si se eliminan.

	precision	recall	f1-score	support
ESTADO	0.25	0.18	0.21	312
MERCANTE	0.87	0.89	0.88	24164
MILITAR	0.41	0.04	0.07	378
OCIO	0.45	0.29	0.36	1984
PASAJEROS	0.55	0.57	0.56	3122
PESQUERO	0.51	0.66	0.58	3332
PETROLERO	0.80	0.73	0.76	10386
REMOLCADOR	0.65	0.74	0.69	3978
SUBACUÁTICO	0.07	0.01	0.02	88
accuracy			0.77	47744
macro avg	0.51	0.46	0.46	47744
weighted avg	0.76	0.77	0.76	47744



Experimento 34

Como se ha mencionado, se puede apreciar la diferencia entre el número de entradas en el experimento 29, 49647, por los 47744 del ensayo 34. Asimismo, se puede apreciar en algunos valores del *classification_report* al comparar ambos experimentos, que el último, mejora ligeramente algunos de los resultados del anterior, siendo efectivo el proceso de preprocesado y depurado mencionado.

Es importante recalcar que todos estos experimentos han sido realizados de forma paralela con los 5 tipos de buques más representativos, obteniéndose las mismas conclusiones. Estos quedaran reflejados en el Anexo III: Resultados de otros experimentos realizados

4.3 Resultados dinámicos

Finalizado el análisis de los experimentos con datos estáticos, se procede a reflejar los resultados de aquellos experimentos realizados con los datos dinámicos reflejados en la Tabla 3-9. El análisis de estos experimentos seguirá la misma línea empleada en el apartado anterior, es decir, está basado en las conclusiones obtenidas en la parte de datos estáticos.

Estos serán los ensayos 46 y 47, experimentos con el mayor volumen de datos con 9 tipos de barco. El primero si incluye el atributo ‘loa’ de cara a la predicción del modelo, siendo el que mayor importancia adquiere, tal y como se puede ver en la Figura 4-2.

	precision	recall	f1-score	support
ESTADO	1.00	0.03	0.06	157
MERCANTE	0.71	0.94	0.81	24197
MILITAR	0.67	0.09	0.16	87
OCIO	0.20	0.00	0.01	276
PASAJEROS	0.80	0.59	0.68	3181
PESQUERO	0.79	0.87	0.83	4722
PETROLERO	0.78	0.27	0.40	10333
REMOLCADOR	0.71	0.24	0.36	499
SUBACUATICO	1.00	0.03	0.06	30
accuracy			0.73	43482
macro avg	0.74	0.34	0.38	43482
weighted avg	0.74	0.73	0.69	43482

Experimento 46

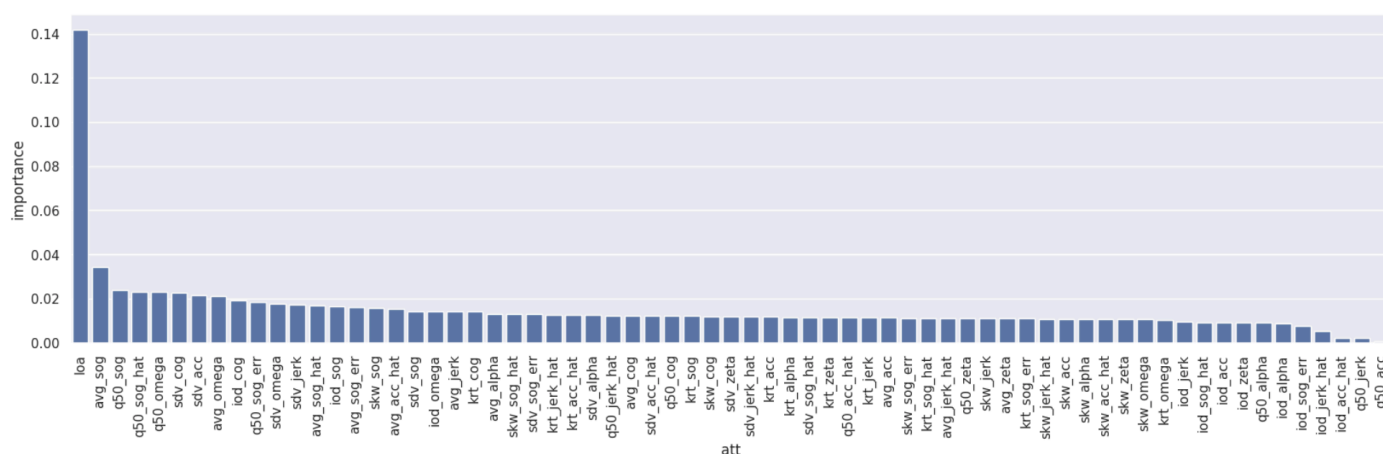


Figura 4-2 Importancia de los atributos empleados con 9 tipos de barcos con ‘LOA’

Sin embargo, si eliminamos el atributo ‘loa’ del proceso de predicción, podemos ver como empeoran los resultados, ya que en la Figura 4-2 se puede apreciar que este atributo es el más influyente, por lo que los resultados se verán mermados.

	precision	recall	f1-score	support
ESTADO	1.00	0.01	0.01	161
MERCANTE	0.64	0.94	0.76	23880
MILITAR	0.00	0.00	0.00	84
OCIO	0.00	0.00	0.00	290
PASAJEROS	0.80	0.56	0.66	3176
PESQUERO	0.77	0.64	0.70	4713
PETROLERO	0.54	0.11	0.18	10400
REMOLCADOR	0.59	0.08	0.14	421
SUBACUATICO	0.00	0.00	0.00	27
accuracy			0.66	43152
macro avg	0.48	0.26	0.27	43152
weighted avg	0.64	0.66	0.59	43152

Experimento 47

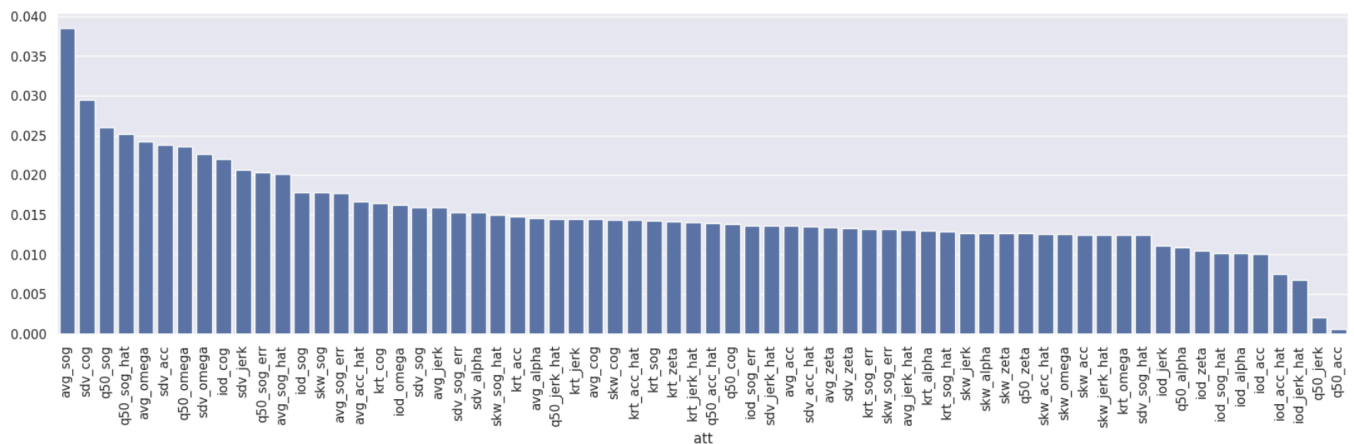


Figura 4-3 Importancia de los atributos empleados con 9 tipos de barcos sin ‘LOA’

En este capítulo se han comentado los resultados más relevantes. Referimos al anexo III para consultar el detalle del resto de los experimentos realizados.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Tras realizar una exhaustiva revisión y análisis del modelo de predicción del tipo de barco, se han obtenido las siguientes conclusiones.

En primer lugar, se ha demostrado la importancia del preprocesado y tratamiento inicial de los datos de entrada para mejorar la calidad de los mismos y, por ende, la precisión de las predicciones. Se ha conseguido un modelo de predicción más fiable.

En segundo lugar, se constató durante la realización de los experimentos una significativa mejora en los tiempos de ejecución del modelo, motivada por la optimización del código utilizado en su implementación. Esto nos ha permitido abordar una variedad mayor de experimentos, no solo en cantidad sino también en complejidad de los mismos. Además, se ha comprobado la validez de la propuesta de parámetros para la optimización de la calidad de la predicción del tipo de barco, pues los resultados se han mostrado consistentes en todos los experimentos realizados.

En resumen, las conclusiones obtenidas en este trabajo permiten establecer una base sólida para la mejora y optimización del modelo de predicción del tipo de barco, contribuyendo así al desarrollo de herramientas más precisas y eficaces para la gestión de la navegación marítima.

5.2 Líneas futuras

Este trabajo ha permitido identificar diferentes aspectos a mejorar y optimizar en el modelo de predicción del tipo de barco a partir de datos náuticos. Como posibles líneas futuras, se podría:

- Proponer la utilización de técnicas de aprendizaje automático para la detección y corrección de datos anómalos en el proceso de preprocesado.
- Asimismo, se podría investigar sobre la utilización de otros algoritmos de clasificación realizando una comparativa con los resultados obtenidos empleando *Random Forest*.
- Se podría tratar de aplicar el algoritmo en diferentes regiones geográficas, para detectar si existen comportamientos distintos dependiendo de la zona geográfica analizada.
- Se podría investigar sobre la inclusión de otras variables relacionadas con las características de la ruta seguida, para mejorar aún más la precisión de las predicciones.

En definitiva, existen múltiples posibilidades para seguir mejorando y optimizando el modelo de predicción del tipo de barco, y este trabajo puede servir como punto de partida para futuras investigaciones en este ámbito.

6 BIBLIOGRAFÍA

- [1] «Armada - Ministerio de Defensa de España». <https://www.defensa.gob.es/fuerzasarmadas/ar/> (accedido 22 de marzo de 2023).
- [2] «COVAM - Ministerio de Defensa de España». https://www.defensa.gob.es/defensa_yo/glosario/C/COVAM.html (accedido 22 de marzo de 2023).
- [3] M. Á. Gómez, «El COVAM, los ojos de la Armada que velan por los buques españoles», *COPE*, 9 de octubre de 2019. https://www.cope.es/actualidad/defensa/noticias/covam-los-ojos-armada-que-velan-por-los-buques-espanoles-20191009_516197 (accedido 22 de marzo de 2023).
- [4] «La Armada usará inteligencia artificial para vigilar el entorno marítimo desde Cartagena - La Opinión de Murcia». <https://www.laopiniondemurcia.es/cartagena/2021/11/10/armada-usara-inteligencia-artificial-vigilar-59387366.html> (accedido 21 de marzo de 2023).
- [5] E. P. S.L, «El COVAM se convierte en la punta de lanza de la Inteligencia Artificial en la Armada», *Murciaplaza*. <https://murciaplaza.com/el-covam-se-convierte-en-la-punta-de-lanza-de-la-inteligencia-artificial-en-la-armada> (accedido 22 de marzo de 2023).
- [6] «Defensa modernizará el centro de operaciones y vigilancia de la Armada». <https://www.infodefensa.com/texto-diario/mostrar/3129114/defensa-modernizara-centro-operaciones-vigilancia-armada> (accedido 21 de marzo de 2023).
- [7] G. Rodríguez Casajús, B. Barragáns Martínez (advisor), y P. Sendín Raña (advisor), «Predicción de tipo de buque utilizando datos AIS y técnicas de inteligencia artificial», Centro Universitario de la Defensa de Marín, Working Paper, abr. 2022. Accedido: 29 de abril de 2023. [En línea]. Disponible en: <http://calderon.cud.uvigo.es/handle/123456789/530>
- [8] «Inteligencia artificial: definición, historia, usos, peligros». <https://datascientest.com/es/inteligencia-artificial-definicion> (accedido 21 de marzo de 2023).
- [9] R.- ASALE y RAE, «inteligencia | Diccionario de la lengua española», «*Diccionario de la lengua española*» - Edición del Tricentenario. <https://dle.rae.es/inteligencia> (accedido 22 de marzo de 2023).
- [10] R. Ayala, «Talos, el mito de la primera criatura autómatas de la historia», *Muy Interesante*, 8 de septiembre de 2022. <https://www.muyinteresante.com.mx/historia/talos-el-mito-de-la-primer-criatura-automata-de-la-historia/> (accedido 23 de marzo de 2023).
- [11] «500 Years Later, da Vinci's Mechanical Lion Is Brought to Life». <https://www.popularmechanics.com/technology/a29020685/leonardo-da-vinci-mechanical-lion-display/> (accedido 21 de marzo de 2023).
- [12] «Eugene, el mayor órdago al test de Turing viene desde Rusia».

<https://www.xataka.com/robotica-e-ia/eugene-el-mayor-ordago-al-test-de-turing-viene-desde-rusia> (accedido 21 de marzo de 2023).

[13] «Qué tipos de problemas podemos resolver con técnicas de Aprendizaje Automático (Machine Learning) | campusMVP.es». <https://www.campusmvp.es/recursos/post/que-tipos-de-problemas-podemos-resolver-con-tecnicas-de-aprendizaje-automatico-machine-learning.aspx> (accedido 21 de marzo de 2023).

[14] «Introducción a Machine Learning: Algunos conceptos básicos – Llipe.com». <https://www.lliipe.com/2017/03/13/introduccion-a-machine-learning/> (accedido 23 de marzo de 2023).

[15] «Aprendizaje por Refuerzo | Aprende Machine Learning». <https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/> (accedido 21 de marzo de 2023).

[16] «Algoritmos de aprendizaje por refuerzo | Interactive Chaos». <https://interactivechaos.com/es/manual/tutorial-de-machine-learning/algoritmos-de-aprendizaje-por-refuerzo> (accedido 22 de marzo de 2023).

[17] «Aprendizaje por refuerzo y ajedrez: Alpha Zero». <https://www.ui1.es/blog-ui1/aprendizaje-por-refuerzo-y-ajedrez-alpha-zero> (accedido 21 de marzo de 2023).

[18] «Random Forest (Bosque Aleatorio): combinando árboles - IArtificial.net», 10 de junio de 2019. <https://www.iartificial.net/random-forest-bosque-aleatorio/> (accedido 23 de marzo de 2023).

[19] «ENCOMAR - Entorno Colaborativo Marítimo de la Armada». <https://encomar.covam.es/> (accedido 21 de marzo de 2023).

[20] «IDE “JupyterLab 3.0” is Released, the successor to Jupyter Notebook | OSDN Magazine». <https://mag-en.osdn.jp/mag/863> (accedido 21 de marzo de 2023).

[21] «El tutorial de Python», *Python documentation*. <https://docs.python.org/3/tutorial/index.html> (accedido 21 de marzo de 2023).

[22] «Seguridad AIS - Para qué sirve». <https://www.azimutmarine.es/seguridad-ais> (accedido 21 de marzo de 2023).

[23] Na8, «Detección de outliers en Python», *Aprende Machine Learning*, 2 de junio de 2020. <https://www.aprendemachinelearning.com/deteccion-de-outliers-en-python-anomalia/> (accedido 21 de marzo de 2023).

[24] «Diagrama BoxPlot - Proyectos Gestión Conocimiento». <https://www.pgconocimiento.com/diagrama-boxplot/> (accedido 21 de marzo de 2023).

[25] «Diagrama de caja—ArcGIS Pro | Documentación». <https://pro.arcgis.com/es/pro-app/latest/help/analysis/geoprocessing/charts/box-plot.htm> (accedido 21 de marzo de 2023).

[26] «Detect and Remove the Outliers using Python - GeeksforGeeks». <https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/> (accedido 21 de marzo de 2023).

[27] «Effective Outlier Detection Techniques in Machine Learning | by Mehul Ved | Medium». <https://medium.com/@mehulved1503/effective-outlier-detection-techniques-in-machine-learning-ef609b6ade72> (accedido 21 de marzo de 2023).

[28] B. A. Miravet, «Mejora de las predicciones en muestras desbalanceadas».

[29] Na8, «Clasificación con datos desbalanceados», *Aprende Machine Learning*, 16 de mayo de 2019. <https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/> (accedido 21 de marzo de 2023).

[30] «7 Techniques to Handle Imbalanced Data», *KDnuggets*. <https://www.kdnuggets.com/7-techniques-to-handle-imbalanced-data.html> (accedido 21 de marzo de 2023).

- [31] «Undersampling», *CORP-MIDS I* (MDS). <https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/> (accedido 21 de marzo de 2023).
- [32] «What Is Undersampling?» <https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/> (accedido 23 de marzo de 2023).
- [33] K. Pykes, «Oversampling and Undersampling», *Medium*, 10 de septiembre de 2020. <https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf> (accedido 21 de marzo de 2023).
- [34] «python - MinMaxScaler de sKlearn solo con Numpy - Stack Overflow en español». <https://es.stackoverflow.com/questions/400007/minmaxscaler-de-sklearn-solo-con-numpy> (accedido 22 de marzo de 2023).
- [35] «MinMaxScaler | Interactive Chaos». <https://interactivechaos.com/es/manual/tutorial-de-machine-learning/minmaxscaler> (accedido 22 de marzo de 2023).
- [36] «Precision, Recall, F1, Accuracy en clasificación - IArtificial.net», 17 de noviembre de 2019. <https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/> (accedido 21 de marzo de 2023).
- [37] «Machine Learning: Seleccin Metricas de clasificacion - sitiobigdata.com». <https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/#> (accedido 21 de marzo de 2023).

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

Por la temática del TFG realizado y su ámbito de aplicación, no se considera necesario ni pertinente incluir ninguna reflexión sobre aspectos sociales, económicos ni ambientales.

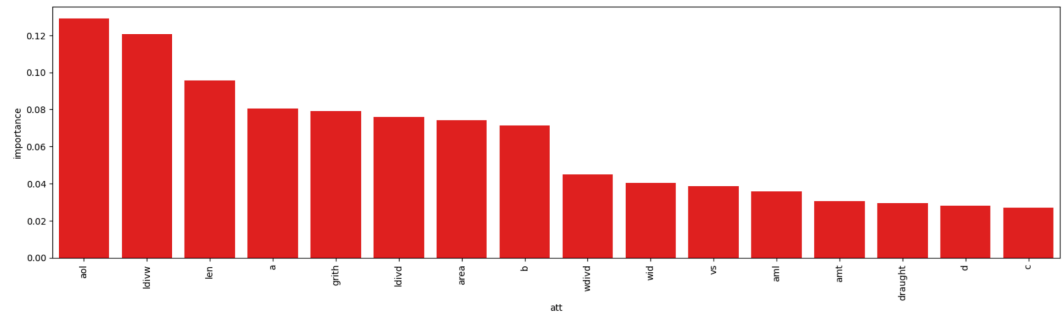
ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

Por la temática de TFG abordado, se podrían analizar las implicaciones éticas que subyacen en el empleo de la Inteligencia Artificial. Se soslaya este análisis en este caso, dado que no se propone suplantar el trabajo del operador humano que haga uso de este sistema, sino simplemente proporcionarle información más completa para la toma de decisiones.

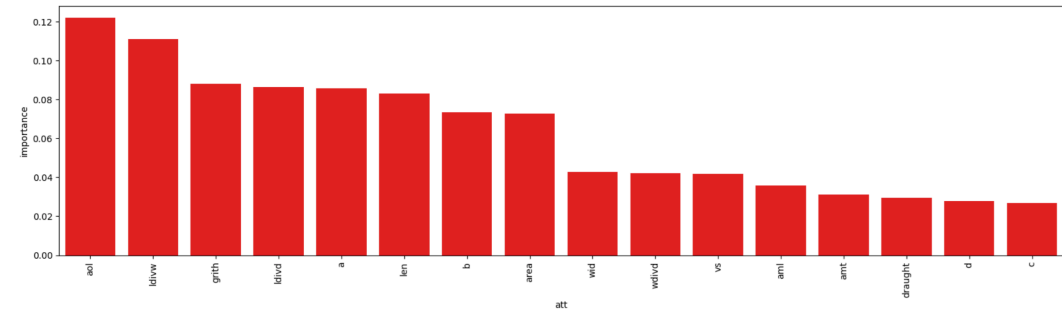
ANEXO III: RESULTADOS DE OTROS EXPERIMENTOS REALIZADOS

IMPORTANCIA DE ATRIBUTOS EN EL MODELO *RANDOM FOREST* CON 5 TIPOS DE BARCOS

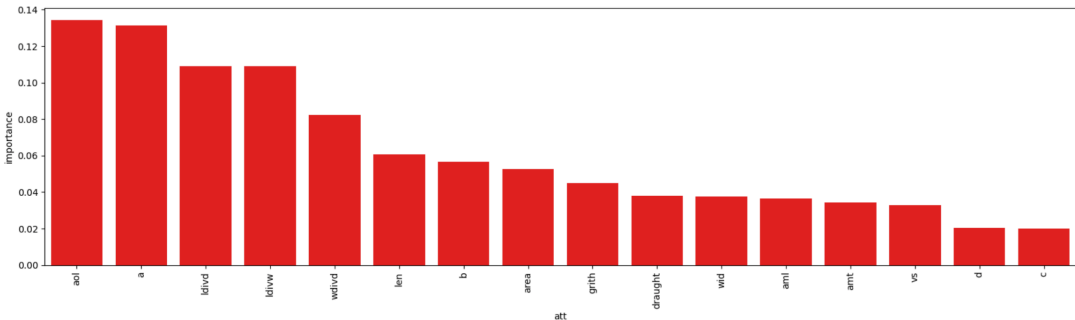
1. Sin OS ni US



2. Con OS

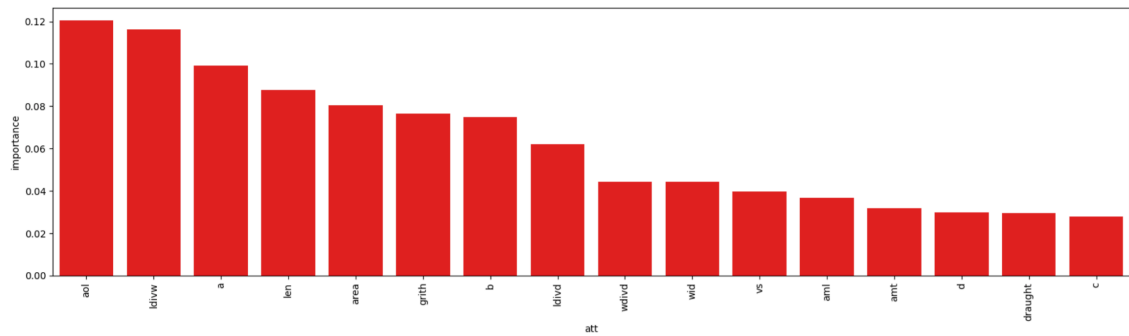


3. Con US

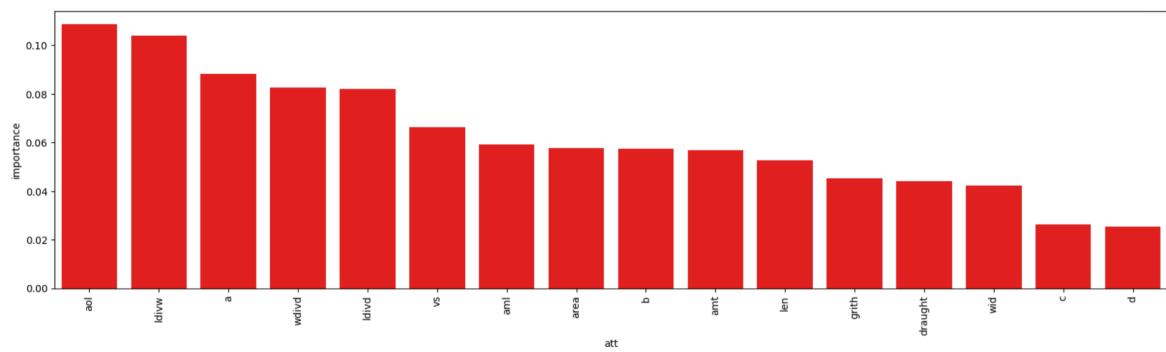


IMPORTANCIA DE ATRIBUTOS EN EL MODELO *RANDOM FOREST* CON 9 TIPOS DE BARCOS

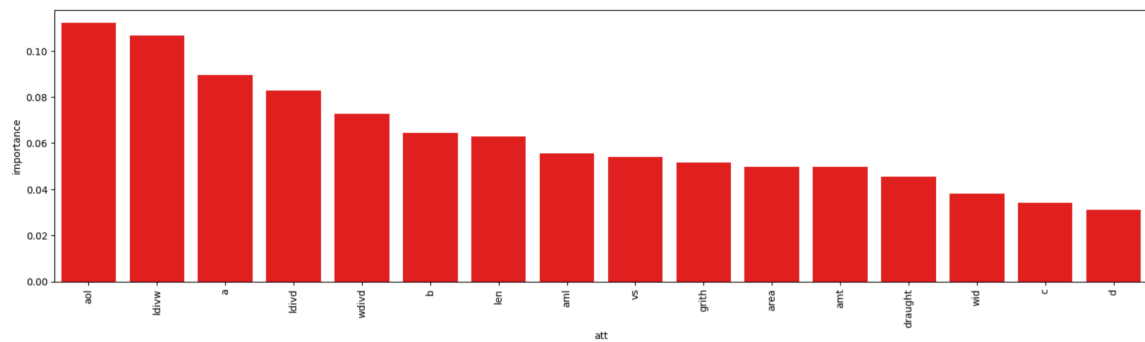
1. Sin OS ni US



2. Con OS

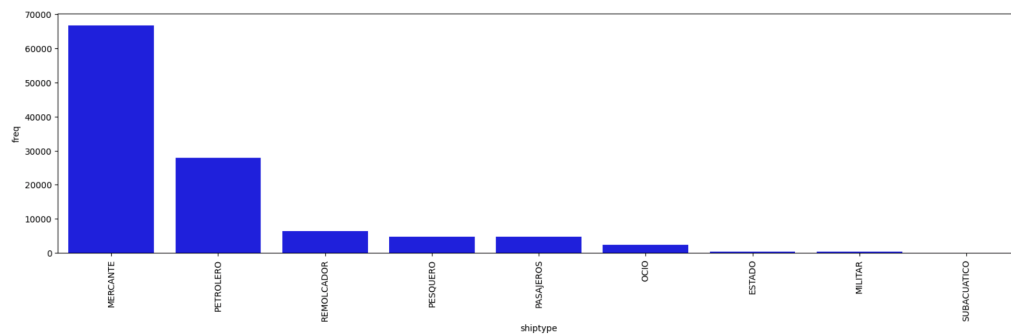


3. Con US



Archivos estáticos 1 mes

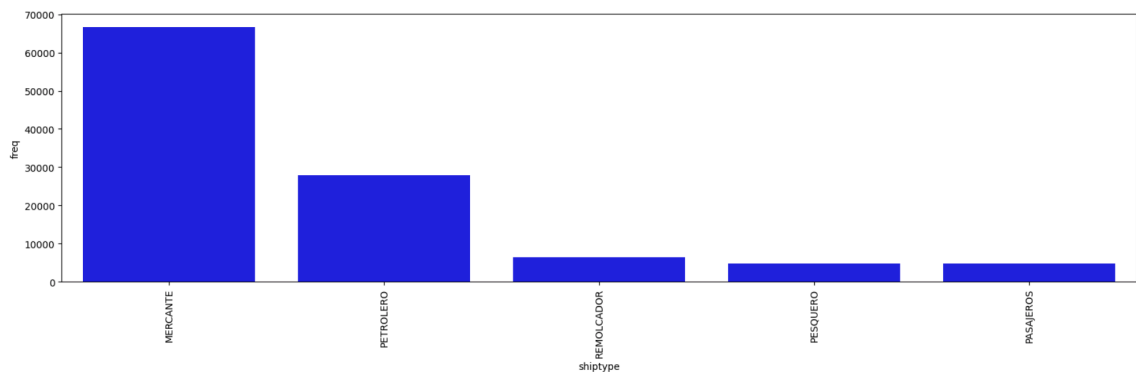
Ais_static_202206.csv sin OS ni US con 9 tipos



```
#Random Forest
pipe = Pipeline([('scaler', MinMaxScaler()), ('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_OS, Y_OS) #entrenamiento del modelo
y_pred = model.predict(X_test_OS) #evaluación del modelo
print(classification_report(Y_test_OS, y_pred, zero_division=0))
```

	precision	recall	f1-score	support
ESTADO	0.48	0.34	0.40	125
MERCANTE	0.95	0.96	0.96	22287
MILITAR	0.67	0.29	0.40	135
OCIO	0.58	0.56	0.57	814
PASAJEROS	0.74	0.72	0.73	1558
PESQUERO	0.69	0.71	0.70	1604
PETROLERO	0.93	0.91	0.92	9295
REMOLCADOR	0.79	0.79	0.79	2176
SUBACUATICO	0.82	0.22	0.35	41
accuracy			0.91	38035
macro avg	0.74	0.61	0.65	38035
weighted avg	0.91	0.91	0.91	38035

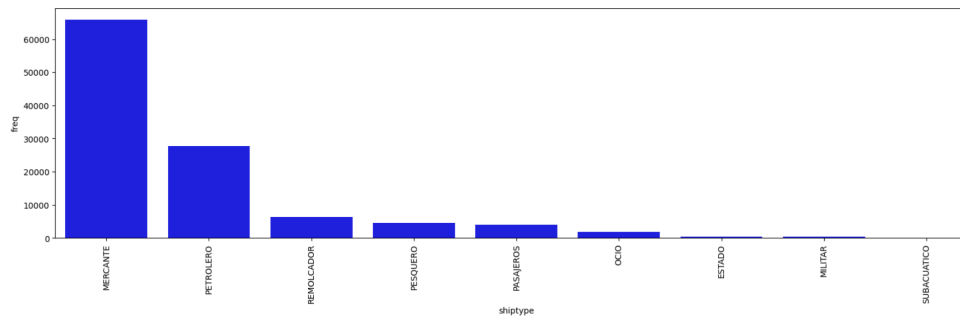
Ais_static_202206.csv sin OS ni US con 5 tipos



```
49]: #Random Forest
pipe = Pipeline([('scaler', MinMaxScaler()), ('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_OS, Y_OS) #entrenamiento del modelo
y_pred = model.predict(X_test_OS) #evaluación del modelo
print(classification_report(Y_test_OS, y_pred, zero_division=0))
```

	precision	recall	f1-score	support
MERCANTE	0.95	0.96	0.96	22363
PASAJEROS	0.81	0.76	0.79	1570
PESQUERO	0.75	0.76	0.76	1570
PETROLERO	0.93	0.90	0.92	9270
REMOLCADOR	0.80	0.80	0.80	2127
accuracy			0.92	36900
macro avg	0.85	0.84	0.84	36900
weighted avg	0.92	0.92	0.92	36900

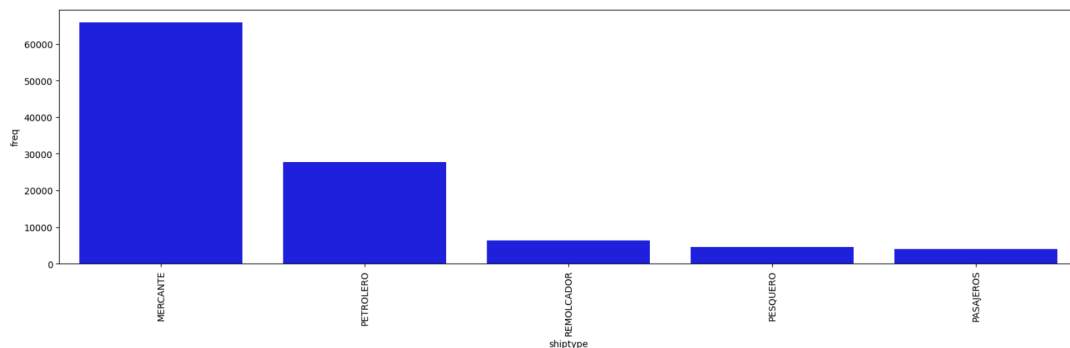
AisStaticEnero2023.csv sin OS ni US con 9 tipos



```
#Random Forest
pipe = Pipeline([('scaler', MinMaxScaler()),('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_OS, Y_OS) #entrenamiento del modelo
y_pred = model.predict(X_test_OS) #evaluación del modelo
print(classification_report(Y_test_OS, y_pred, zero_division=0))
```

	precision	recall	f1-score	support
ESTADO	0.57	0.37	0.45	134
MERCANTE	0.94	0.96	0.95	21951
MILITAR	0.68	0.30	0.42	113
OCIO	0.56	0.51	0.53	620
PASAJEROS	0.74	0.73	0.74	1388
PESQUERO	0.70	0.73	0.71	1485
PETROLERO	0.93	0.90	0.91	9302
REMOLCADOR	0.79	0.81	0.80	2098
SUBACUATICO	0.50	0.17	0.25	42
accuracy			0.91	37133
macro avg	0.71	0.61	0.64	37133
weighted avg	0.90	0.91	0.90	37133

AisStaticEnero2023.csv sin OS ni US con 5 tipos



```
#Random Forest
pipe = Pipeline([('scaler', MinMaxScaler()),('classifier',
RandomForestClassifier(n_estimators=100))])
model = pipe.fit(X_OS, Y_OS) #entrenamiento del modelo
y_pred = model.predict(X_test_OS) #evaluación del modelo
print(classification_report(Y_test_OS, y_pred, zero_division=0))
```

	precision	recall	f1-score	support
MERCANTE	0.95	0.96	0.95	21935
PASAJEROS	0.80	0.76	0.78	1317
PESQUERO	0.77	0.78	0.78	1553
PETROLERO	0.93	0.90	0.92	9272
REMOLCADOR	0.82	0.83	0.82	2161
accuracy			0.92	36238
macro avg	0.85	0.85	0.85	36238
weighted avg	0.92	0.92	0.92	36238

aisStaticEnero2023.csv sin OS ni US con 9 tipos

	precision	recall	f1-score	support
ESTADO	0.59	0.37	0.46	134
MERCANTE	0.94	0.96	0.95	21951
MILITAR	0.64	0.28	0.39	113
OCIO	0.58	0.52	0.55	620
PASAJEROS	0.75	0.73	0.74	1388
PESQUERO	0.69	0.73	0.71	1485
PETROLERO	0.93	0.90	0.92	9302
REMOLCADOR	0.79	0.81	0.80	2098
SUBACUATICO	0.53	0.21	0.31	42
accuracy			0.91	37133
macro avg	0.71	0.61	0.65	37133
weighted avg	0.91	0.91	0.91	37133

aisStaticEnero2023.csv sin OS ni US con 5 tipos

	precision	recall	f1-score	support
MERCANTE	0.95	0.96	0.95	21935
PASAJEROS	0.79	0.75	0.77	1317
PESQUERO	0.77	0.78	0.77	1553
PETROLERO	0.93	0.90	0.92	9272
REMOLCADOR	0.81	0.83	0.82	2161
accuracy			0.92	36238
macro avg	0.85	0.84	0.85	36238
weighted avg	0.92	0.92	0.92	36238

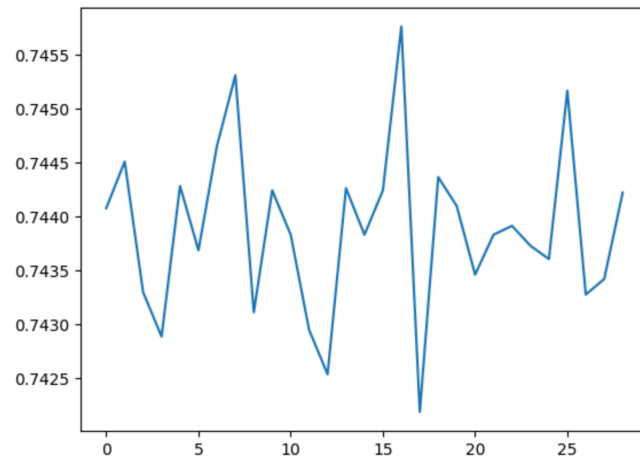
Ais_static_202206.csv sin OS ni US con 9 tipos

	precision	recall	f1-score	support
ESTADO	0.53	0.37	0.43	125
MERCANTE	0.95	0.96	0.96	22287
MILITAR	0.66	0.32	0.43	135
OCIO	0.57	0.56	0.57	814
PASAJEROS	0.75	0.72	0.73	1558
PESQUERO	0.69	0.71	0.70	1604
PETROLERO	0.93	0.91	0.92	9295
REMOLCADOR	0.79	0.79	0.79	2176
SUBACUATICO	0.69	0.22	0.33	41
accuracy			0.91	38035
macro avg	0.73	0.62	0.65	38035
weighted avg	0.91	0.91	0.91	38035

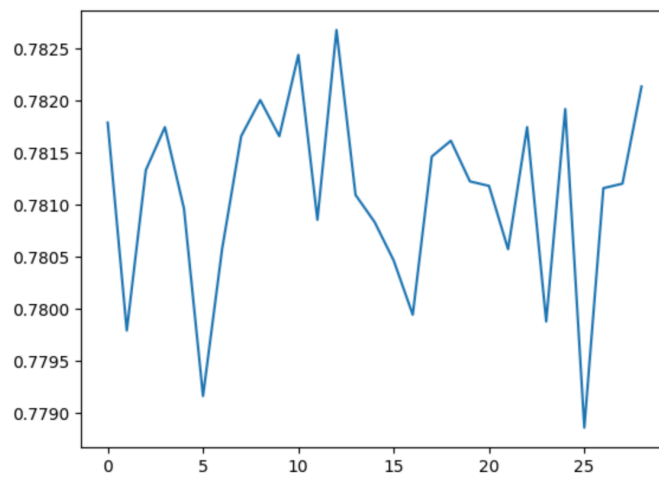
Ais_static_202206.csv sin OS ni US con 5 tipos

	precision	recall	f1-score	support
MERCANTE	0.95	0.96	0.96	22363
PASAJEROS	0.81	0.77	0.79	1570
PESQUERO	0.75	0.76	0.75	1570
PETROLERO	0.93	0.90	0.92	9270
REMOLCADOR	0.80	0.80	0.80	2127
accuracy			0.92	36900
macro avg	0.85	0.84	0.84	36900
weighted avg	0.92	0.92	0.92	36900

Gráfica evolución precisión 30 días datos estáticos 9 tipos sin calado haciendo separación *MMSI*



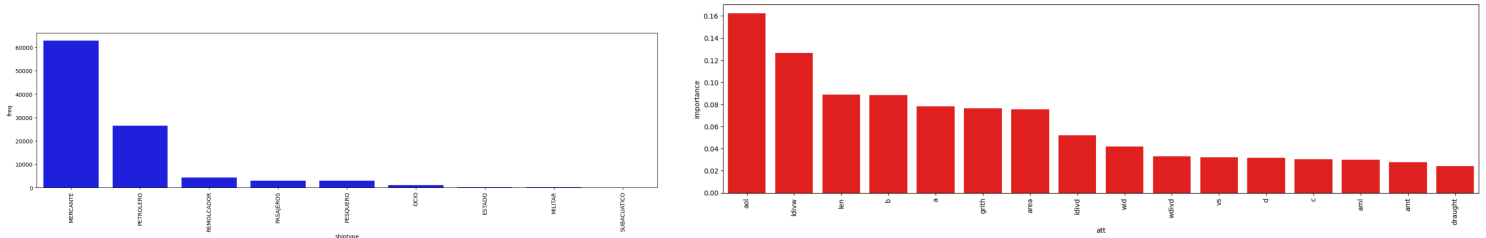
Gráfica evolución precisión 30 días datos estáticos 5 tipos sin calado haciendo separación *MMSI*



Análisis de los meses de septiembre, enero y junio corrigiendo la separación de *MMSI*, sin eliminar *MMSI* duplicados y quitando calado=0

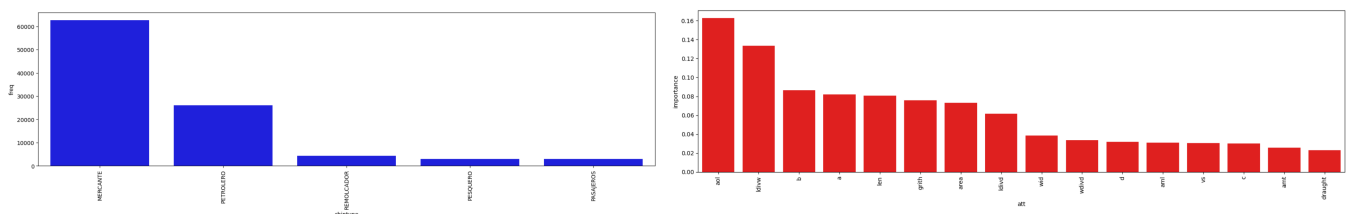
aisStaticEnero2023.csv

Resultados Estáticos quitando calado = 0, 9 tipos de barcos, separando los datos por *MMSI* y sin eliminar duplicados en *MMSI*



	precision	recall	f1-score	support
ESTADO	0.25	0.18	0.21	303
MERCANTE	0.85	0.90	0.87	24913
MILITAR	0.33	0.06	0.10	309
OCIO	0.49	0.29	0.37	1417
PASAJEROS	0.58	0.53	0.55	2347
PESQUERO	0.51	0.68	0.58	3105
PETROLERO	0.81	0.69	0.75	10616
REMOLCADOR	0.66	0.73	0.69	4111
SUBACUATICO	0.04	0.01	0.02	83
accuracy			0.77	47204
macro avg	0.50	0.45	0.46	47204
weighted avg	0.77	0.77	0.77	47204

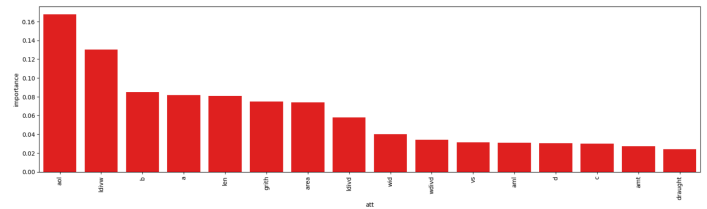
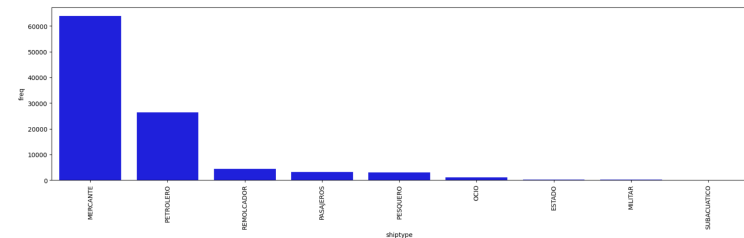
Resultados Estáticos quitando calado = 0, 5 tipos de barcos, separando los datos por *MMSI* y sin eliminar duplicados en *MMSI*.



	precision	recall	f1-score	support
MERCANTE	0.86	0.90	0.88	25128
PASAJEROS	0.65	0.54	0.59	2419
PESQUERO	0.63	0.68	0.65	3073
PETROLERO	0.83	0.71	0.76	10995
REMOLCADOR	0.69	0.76	0.73	4113
accuracy			0.81	45728
macro avg	0.73	0.72	0.72	45728
weighted avg	0.81	0.81	0.81	45728

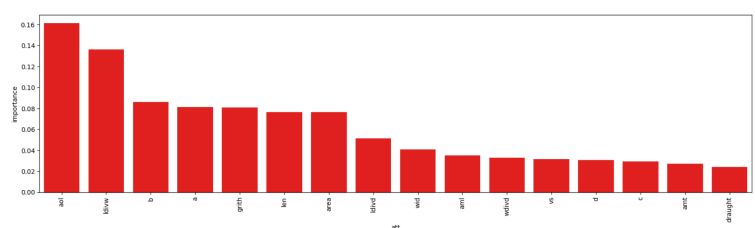
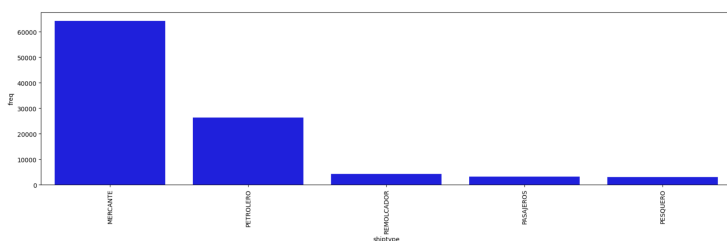
ais_static_202206.csv

Resultados Estáticos quitando calado = 0, 9 tipos de barcos, separando los datos por *MMSI*



	precision	recall	f1-score	support
ESTADO	0.25	0.18	0.21	322
MERCANTE	0.87	0.88	0.87	25131
MILITAR	0.34	0.07	0.11	378
OCIO	0.48	0.31	0.38	2114
PASAJEROS	0.56	0.57	0.56	3102
PESQUERO	0.51	0.65	0.58	3473
PETROLERO	0.78	0.73	0.75	10886
REMOLCADOR	0.64	0.73	0.68	4149
SUBACUATICO	0.06	0.01	0.02	92
accuracy			0.76	49647
macro avg	0.50	0.46	0.46	49647
weighted avg	0.76	0.76	0.76	49647

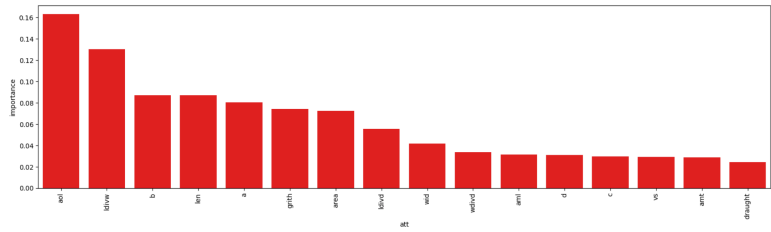
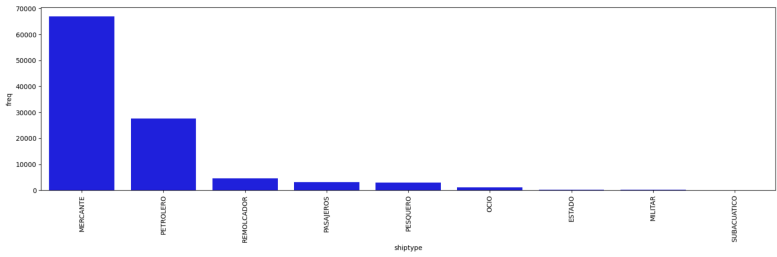
Resultados Estáticos quitando calado = 0, 5 tipos de barcos, separando los datos por *MMSI*



	precision	recall	f1-score	support
MERCANTE	0.86	0.89	0.88	24730
PASAJEROS	0.64	0.62	0.63	3137
PESQUERO	0.61	0.66	0.64	3479
PETROLERO	0.81	0.72	0.76	10906
REMOLCADOR	0.69	0.73	0.71	4229
accuracy			0.80	46481
macro avg	0.72	0.73	0.72	46481
weighted avg	0.80	0.80	0.80	46481

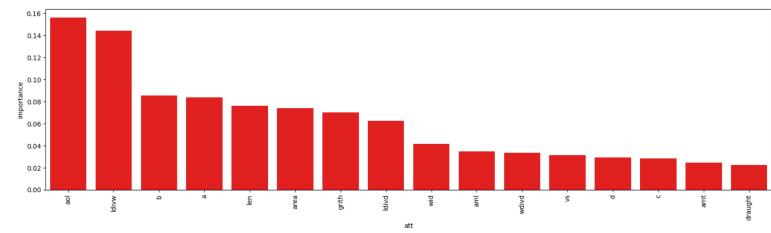
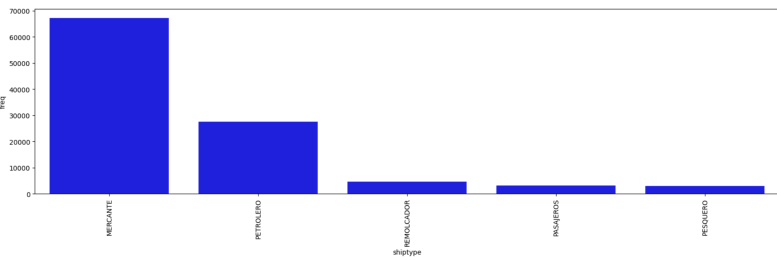
ais_static_202209.csv

Resultados Estáticos quitando calado = 0, 9 tipos de barcos, separando los datos por *MMSI* y sin eliminar duplicados en *MMSI*



	precision	recall	f1-score	support
ESTADO	0.28	0.10	0.15	277
MERCANTE	0.87	0.90	0.88	25754
MILITAR	0.66	0.06	0.10	376
OCIO	0.47	0.36	0.41	1792
PASAJEROS	0.56	0.56	0.56	2658
PESQUERO	0.53	0.64	0.58	3263
PETROLERO	0.81	0.74	0.78	10974
REMOLCADOR	0.66	0.74	0.70	4222
SUBACUATICO	0.04	0.01	0.02	91
accuracy			0.78	49407
macro avg	0.54	0.46	0.46	49407
weighted avg	0.78	0.78	0.78	49407

Resultados Estáticos quitando calado = 0, 5 tipos de barcos, separando los datos por *MMSI* y sin eliminar duplicados en *MMSI*



	precision	recall	f1-score	support
MERCANTE	0.88	0.89	0.88	25550
PASAJEROS	0.64	0.61	0.62	2658
PESQUERO	0.65	0.70	0.67	3276
PETROLERO	0.80	0.75	0.77	11137
REMOLCADOR	0.71	0.75	0.73	4261
accuracy			0.81	46882
macro avg	0.73	0.74	0.74	46882
weighted avg	0.81	0.81	0.81	46882

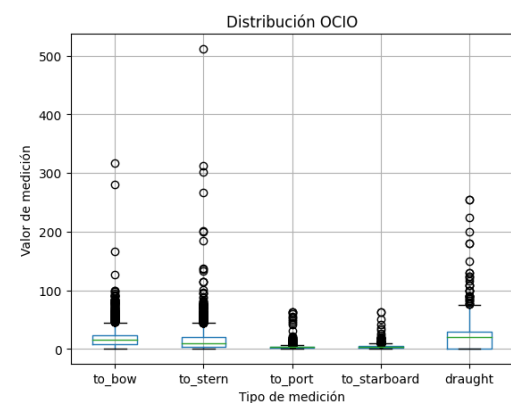
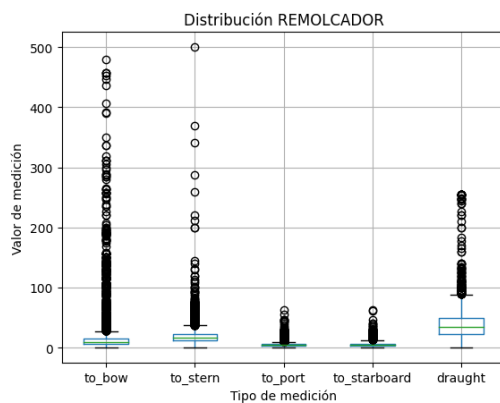
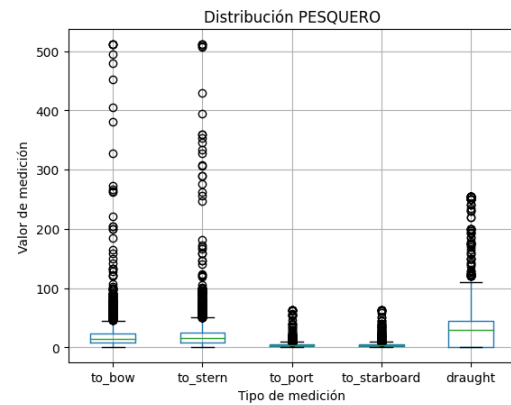
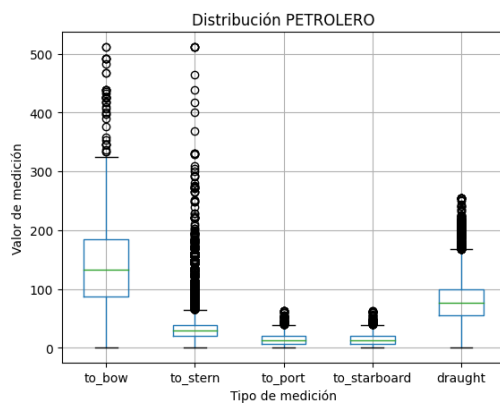
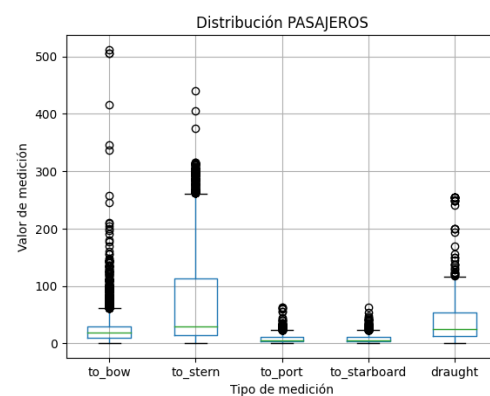
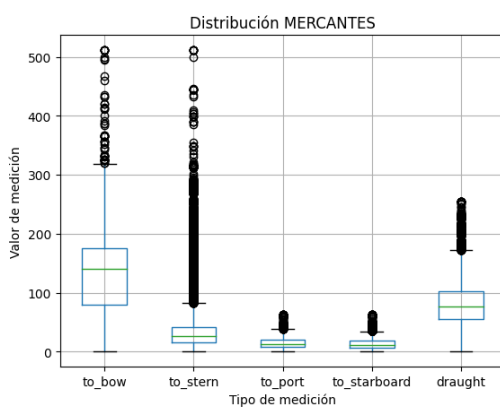
STATIC ELIMINAR OUTLIERS

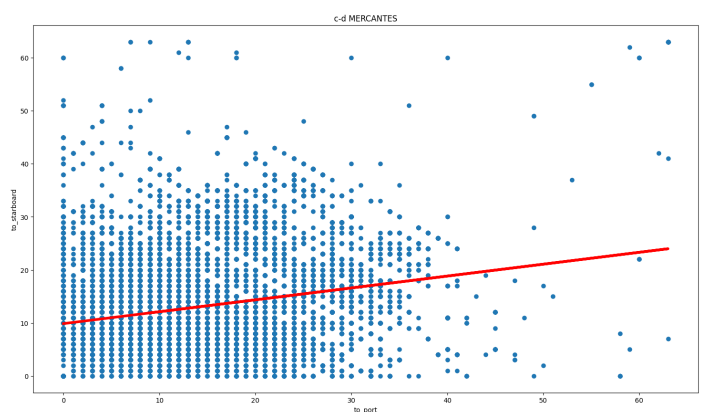
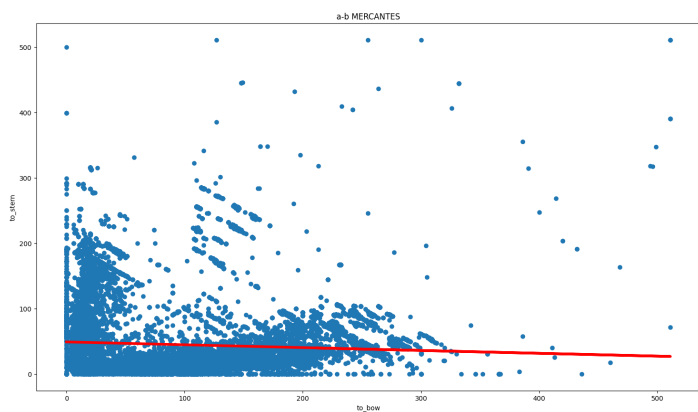
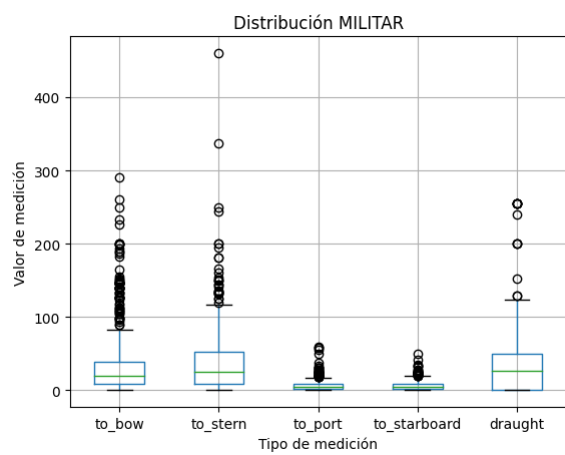
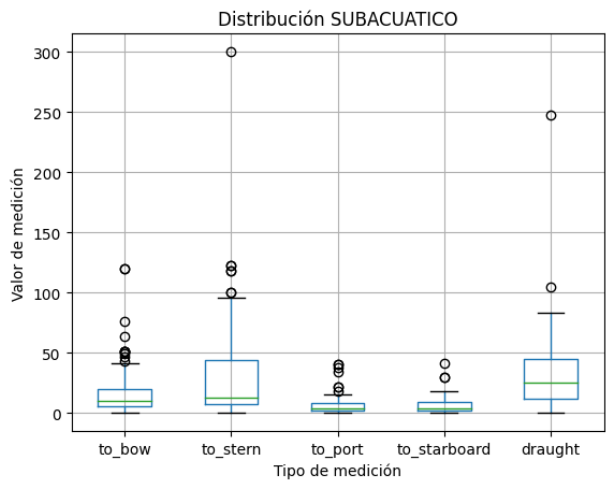
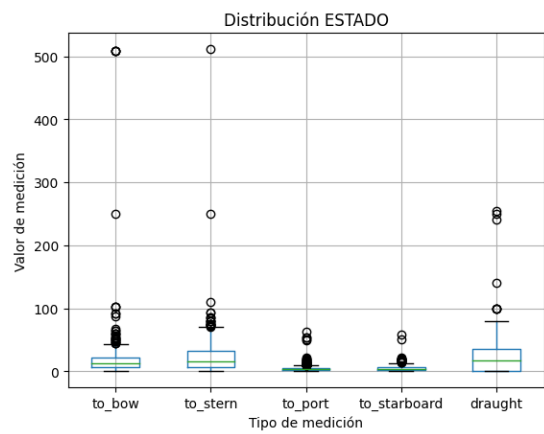
Con esto represento la distribución de los datos originales en los distintos *csv* observando la posible aparición de valores anómalos o outliers

ais_static_202206.csv

Resultados quitando calado = 0, 9 tipos de barcos, separando datos entreno *Random Forest* por *MMSI*

Columnas datos estáticos: *MMSI*, *to_bow*, *to_stern*, *to_port*, *to_starboard*, *draught*, *shiptype*. Como en las columnas *MMSI* y *shiptype* no puede haber outliers, no las tendremos en cuenta.

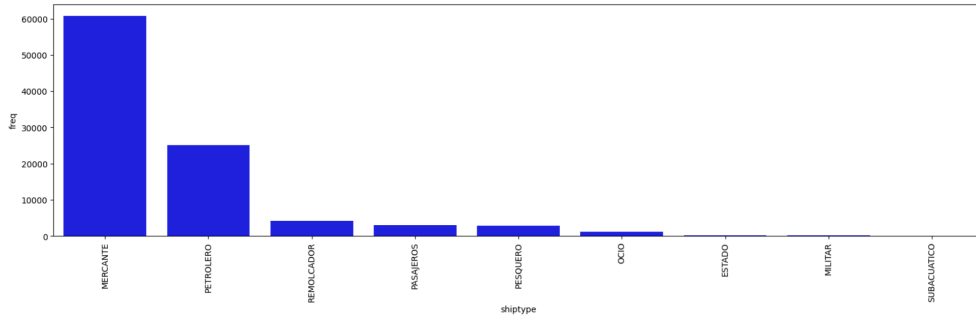




RESULTADOS ESTÁTICOS ELIMINANDO *OUTLIERS*

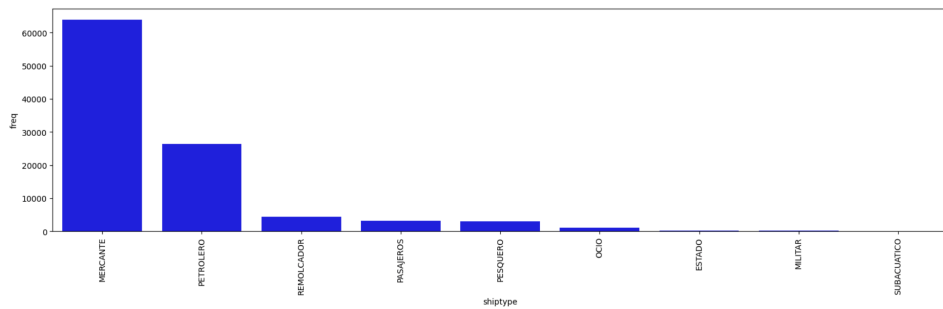
ais_static_202206.csv eliminando outliers 9 *shiptypes*

	precision	recall	f1-score	support
ESTADO	0.25	0.18	0.21	312
MERCANTE	0.87	0.89	0.88	24164
MILITAR	0.41	0.04	0.07	378
OCIO	0.45	0.29	0.36	1984
PASAJEROS	0.55	0.57	0.56	3122
PESQUERO	0.51	0.66	0.58	3332
PETROLERO	0.80	0.73	0.76	10386
REMOLCADOR	0.65	0.74	0.69	3978
SUBACUATICO	0.07	0.01	0.02	88
accuracy			0.77	47744
macro avg	0.51	0.46	0.46	47744
weighted avg	0.76	0.77	0.76	47744



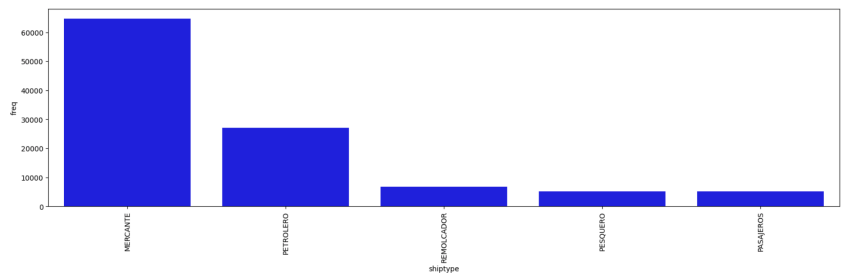
ais_static_202206.csv sin eliminar outliers 9 *shiptypes*

	precision	recall	f1-score	support
ESTADO	0.25	0.18	0.21	322
MERCANTE	0.87	0.88	0.87	25131
MILITAR	0.34	0.07	0.11	378
OCIO	0.48	0.31	0.38	2114
PASAJEROS	0.56	0.57	0.56	3102
PESQUERO	0.51	0.65	0.58	3473
PETROLERO	0.78	0.73	0.75	10886
REMOLCADOR	0.64	0.73	0.68	4149
SUBACUATICO	0.06	0.01	0.02	92
accuracy			0.76	49647
macro avg	0.50	0.46	0.46	49647
weighted avg	0.76	0.76	0.76	49647



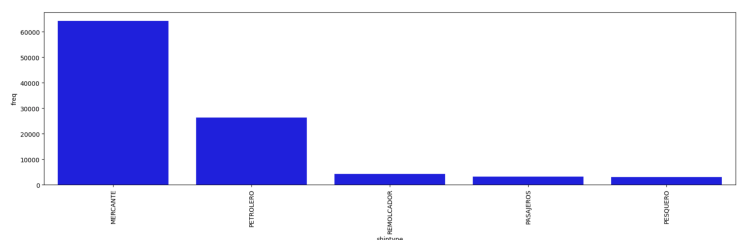
ais_static_202206.csv eliminando outliers 5 *shiptypes*

	precision	recall	f1-score	support
MERCANTE	0.91	0.94	0.92	20249
PASAJEROS	0.71	0.73	0.72	984
PESQUERO	0.77	0.79	0.78	964
PETROLERO	0.88	0.80	0.84	8408
REMOLCADOR	0.82	0.82	0.82	1354
accuracy			0.89	31959
macro avg	0.82	0.82	0.82	31959
weighted avg	0.89	0.89	0.89	31959



ais_static_202206.csv sin eliminar outliers 5 *shiptypes*

	precision	recall	f1-score	support
MERCANTE	0.86	0.89	0.88	24730
PASAJEROS	0.64	0.62	0.63	3137
PESQUERO	0.61	0.66	0.64	3479
PETROLERO	0.81	0.72	0.76	10906
REMOLCADOR	0.69	0.73	0.71	4229
accuracy			0.80	46481
macro avg	0.72	0.73	0.72	46481
weighted avg	0.80	0.80	0.80	46481



DATOS DINÁMICOS

DatosCinematicosRaul.sql. PERIODO DE TIEMPO: más de 1 día, menos de 2

	precision	recall	f1-score	support
ESTADO	0.00	0.00	0.00	36
MERCANTE	0.66	0.93	0.77	3786
MILITAR	0.00	0.00	0.00	32
OCIO	0.58	0.36	0.44	200
PASAJEROS	0.70	0.43	0.53	508
PESQUERO	0.66	0.65	0.65	806
PETROLERO	0.73	0.15	0.26	1549
REMOLCADOR	0.68	0.74	0.71	769
SUBACUATICO	1.00	0.06	0.12	16
accuracy			0.67	7702
macro avg	0.56	0.37	0.39	7702
weighted avg	0.67	0.67	0.62	7702

Sin LOA

	precision	recall	f1-score	support
ESTADO	0.00	0.00	0.00	30
MERCANTE	0.59	0.91	0.71	3609
MILITAR	0.00	0.00	0.00	32
OCIO	0.34	0.08	0.13	164
PASAJEROS	0.73	0.41	0.52	486
PESQUERO	0.62	0.47	0.54	779
PETROLERO	0.42	0.07	0.12	1486
REMOLCADOR	0.58	0.45	0.51	763
SUBACUATICO	0.00	0.00	0.00	12
accuracy			0.59	7361
macro avg	0.36	0.27	0.28	7361
weighted avg	0.55	0.59	0.52	7361

Con 5 tipos de barcos con LOA

	precision	recall	f1-score	support
MERCANTE	0.69	0.93	0.80	3873
PASAJEROS	0.81	0.46	0.59	504
PESQUERO	0.68	0.66	0.67	826
PETROLERO	0.67	0.16	0.25	1444
REMOLCADOR	0.71	0.73	0.72	785
accuracy			0.70	7432
macro avg	0.71	0.59	0.61	7432
weighted avg	0.70	0.70	0.65	7432

Con 5 tipos de barcos sin LOA

	precision	recall	f1-score	support
MERCANTE	0.61	0.91	0.73	3678
PASAJEROS	0.73	0.39	0.51	480
PESQUERO	0.64	0.50	0.56	766
PETROLERO	0.43	0.06	0.11	1433
REMOLCADOR	0.57	0.45	0.50	746
accuracy			0.61	7103
macro avg	0.60	0.46	0.48	7103
weighted avg	0.58	0.61	0.55	7103

DatosCinematicosRaul20230201.sql. PERIODO DE 1 DIA

	precision	recall	f1-score	support
ESTADO	0.00	0.00	0.00	12
MERCANTE	0.68	0.95	0.79	2648
MILITAR	0.00	0.00	0.00	9
OCIO	0.50	0.04	0.08	23
PASAJEROS	0.75	0.51	0.60	324
PESQUERO	0.77	0.84	0.80	567
PETROLERO	0.68	0.14	0.23	1128
REMOLCADOR	0.50	0.04	0.08	48
SUBACUATICO	0.00	0.00	0.00	7
accuracy			0.69	4766
macro avg	0.43	0.28	0.29	4766
weighted avg	0.69	0.69	0.63	4766

SIN LOA

	precision	recall	f1-score	support
ESTADO	0.00	0.00	0.00	10
MERCANTE	0.63	0.94	0.75	2590
MILITAR	0.00	0.00	0.00	9
OCIO	0.00	0.00	0.00	31
PASAJEROS	0.77	0.49	0.60	351
PESQUERO	0.76	0.59	0.66	536
PETROLERO	0.48	0.07	0.13	1131
REMOLCADOR	0.00	0.00	0.00	39
SUBACUATICO	0.00	0.00	0.00	2
accuracy			0.64	4699
macro avg	0.29	0.23	0.24	4699
weighted avg	0.61	0.64	0.57	4699

Con 5 tipos de barcos con LOA

	precision	recall	f1-score	support
MERCANTE	0.67	0.94	0.78	2597
PASAJEROS	0.79	0.54	0.64	362
PESQUERO	0.80	0.84	0.82	550
PETROLERO	0.62	0.12	0.20	1147
REMOLCADOR	0.67	0.07	0.13	56
accuracy			0.69	4712
macro avg	0.71	0.50	0.51	4712
weighted avg	0.68	0.69	0.62	4712

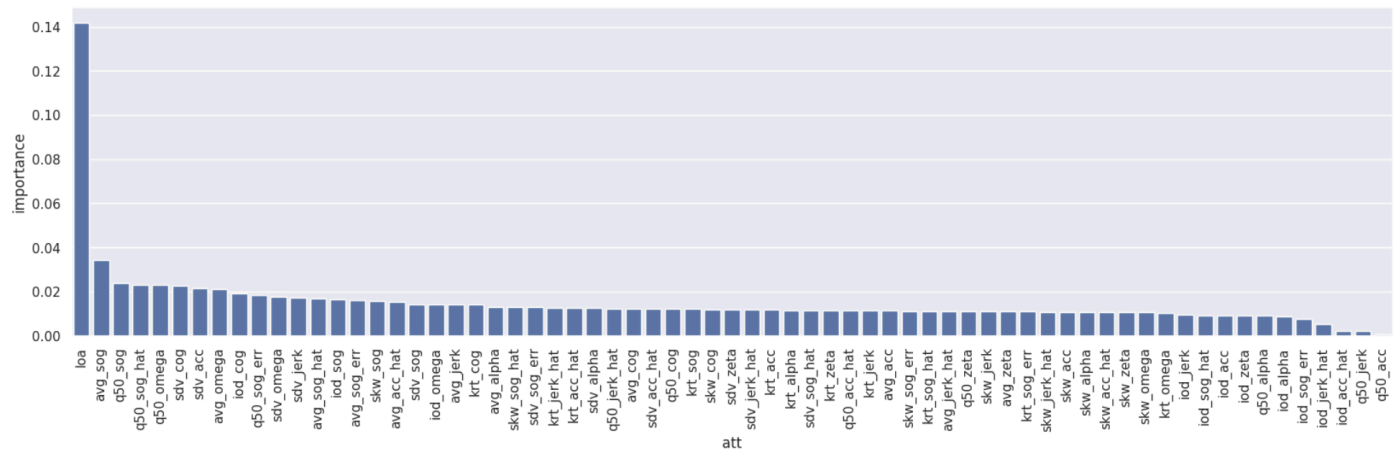
Con 5 tipos de barcos sin LOA

	precision	recall	f1-score	support
MERCANTE	0.64	0.94	0.76	2614
PASAJEROS	0.87	0.54	0.67	364
PESQUERO	0.75	0.62	0.68	525
PETROLERO	0.41	0.06	0.11	1101
REMOLCADOR	0.00	0.00	0.00	42
accuracy			0.66	4646
macro avg	0.54	0.43	0.44	4646
weighted avg	0.61	0.66	0.58	4646

DATA= cinemáticos del 1 al 9 de febrero, periodo de 9 días

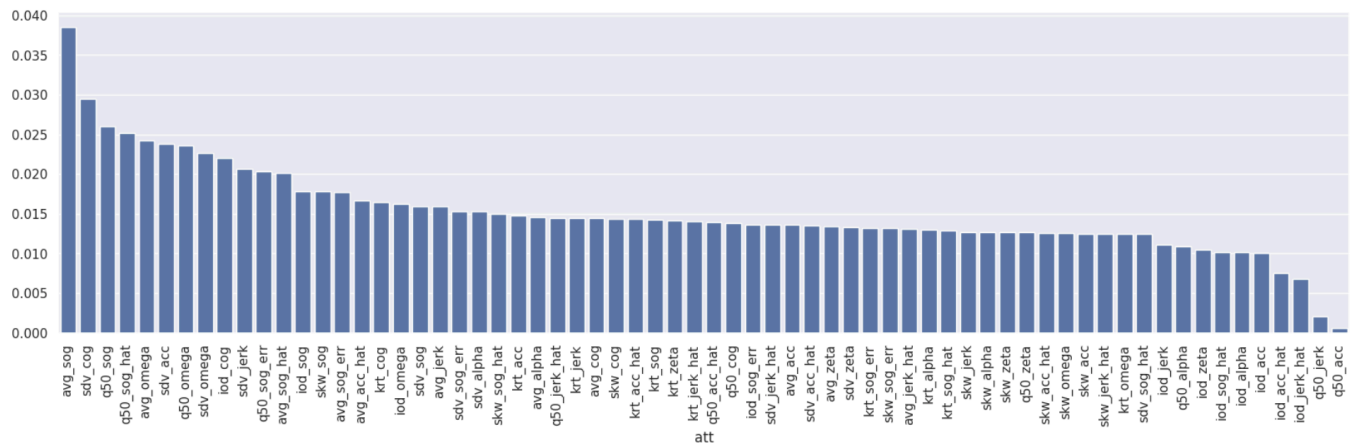
Con 9 tipos de barcos

	precision	recall	f1-score	support
ESTADO	1.00	0.03	0.06	157
MERCANTE	0.71	0.94	0.81	24197
MILITAR	0.67	0.09	0.16	87
OCIO	0.20	0.00	0.01	276
PASAJEROS	0.80	0.59	0.68	3181
PESQUERO	0.79	0.87	0.83	4722
PETROLERO	0.78	0.27	0.40	10333
REMOCADOR	0.71	0.24	0.36	499
SUBACUATICO	1.00	0.03	0.06	30
accuracy			0.73	43482
macro avg	0.74	0.34	0.38	43482
weighted avg	0.74	0.73	0.69	43482



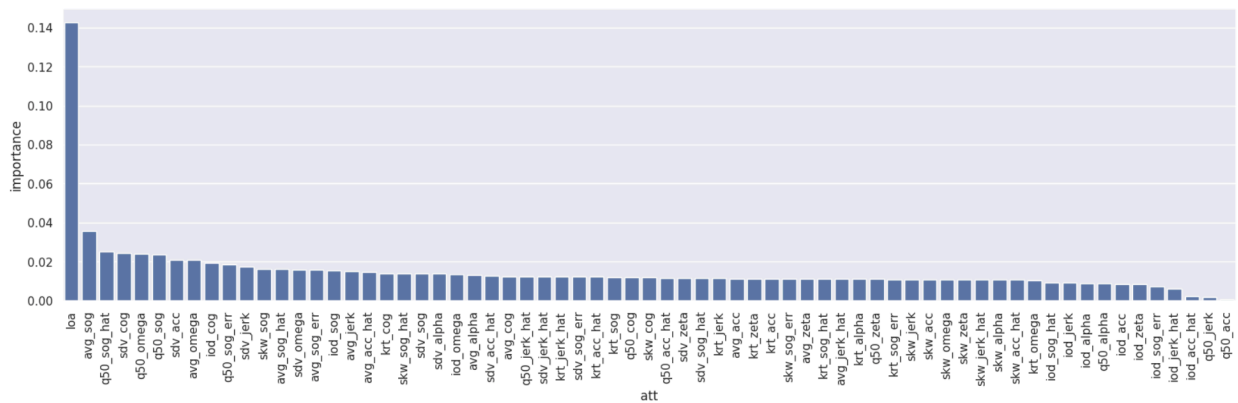
Con 9 tipos de barcos SIN LOA

	precision	recall	f1-score	support
ESTADO	1.00	0.01	0.01	161
MERCANTE	0.64	0.94	0.76	23880
MILITAR	0.00	0.00	0.00	84
OCIO	0.00	0.00	0.00	290
PASAJEROS	0.80	0.56	0.66	3176
PESQUERO	0.77	0.64	0.70	4713
PETROLERO	0.54	0.11	0.18	10400
REMOCADOR	0.59	0.08	0.14	421
SUBACUATICO	0.00	0.00	0.00	27
accuracy			0.66	43152
macro avg	0.48	0.26	0.27	43152
weighted avg	0.64	0.66	0.59	43152



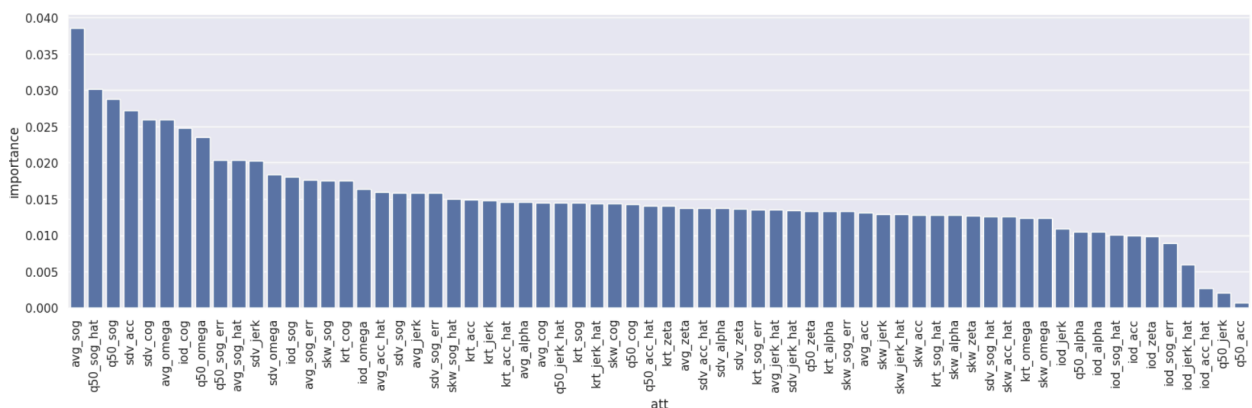
Con 5 tipos de barcos

	precision	recall	f1-score	support
MERCANTE	0.71	0.95	0.81	24019
PASAJEROS	0.84	0.59	0.69	3231
PESQUERO	0.82	0.87	0.85	4707
PETROLERO	0.79	0.28	0.41	10508
REMOLCADOR	0.78	0.26	0.39	468
accuracy			0.74	42933
macro avg	0.79	0.59	0.63	42933
weighted avg	0.76	0.74	0.71	42933



Con 5 tipos de barcos sin LOA

	precision	recall	f1-score	support
MERCANTE	0.64	0.94	0.76	23868
PASAJEROS	0.84	0.55	0.67	3205
PESQUERO	0.77	0.62	0.69	4652
PETROLERO	0.54	0.10	0.17	10421
REMOLCADOR	0.67	0.08	0.14	465
accuracy			0.66	42611
macro avg	0.69	0.46	0.48	42611
weighted avg	0.65	0.66	0.60	42611



ANEXO IV: CÓDIGO STATIC

```
import pandas as pd
import sqlite3
import numpy as np
from tqdm import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import GroupShuffleSplit
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

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

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

path_data = 'DatosStaticos.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()
data=pd.read_csv('aisStaticEnero2023.csv', sep=',',chunksize=200000)

for df in data:
    chunk=df[cols]
    chunk.columns=ncol
    atributos=statics(chunk) #Cálculo de los atributos
    atributos=atributos.drop_duplicates()
    atributos.to_sql('static_data', con=conn, if_exists='replace', index=False)
cur.close()
conn.close()

path_data = 'DatosStaticos.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()
df = pd.read_sql(f"SELECT * FROM static_data """, con=conn)
cur.close()
conn.close()
df=df.drop_duplicates()
```

```
path_data = 'DatosStaticos.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()
df.to_sql('static_data', con=conn, if_exists='replace', index=False)
cur.close()
conn.close()
```

#la base de datos es DatosStaticos.sql y static_data es una tabla donde se guardan los datos de la base de datos

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

#Ejecuto esta o la de abajo, dependiendo de donde quiero que coja los datos, de la base de datos o de los csv's

```
conn = sqlite3.connect('DatosStaticos.sql')
cur = conn.cursor()
df = pd.read_sql("""SELECT * FROM static_data""", con=conn)
cur.close()
conn.close()
```

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

#Ejecuto esta o la de arriba, dependiendo de donde quiero que coja los datos, de la base de datos o de los csv's

```
path_data = 'DatosStaticos14dias.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()
df = pd.read_sql(f"""SELECT * FROM estaticos """, con=conn)
cur.close()
conn.close()
```

```
df = df[df['draught'] != 0]
df=df[df.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
df=df.dropna()
df=df.drop_duplicates()
df=df[df.shiptype.isin(['MERCANTE','PETROLERO','PESQUERO','REMOLCADOR','PASAJEROS'])]
#df=df.drop('mmsi',axis=1) #eliminación de datos que no se usan como atributo (lo hago luego despues del merge)
#X=df.drop('shiptype', axis=1) #Variables independientes
#y=df.shiptype #Variable dependiente
#df
```

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

```
outlier_df=df
outlier_df = df[df['shiptype'] == 'MERCANTE']
```

```
cols_to_plot = ['a', 'b', 'c', 'd', 'draught']
```

```
outlier_df[cols_to_plot].boxplot()
```

```
plt.title('Distribución MERCANTE')
plt.xlabel("Tipo de medición")
plt.ylabel("Valor de medición")
```

```
plt.show()
```

#Si queremos eliminar todos los outliers de los parámetros estáticos:

```
ship_types =
['MERCANTE','PETROLERO','REMOLCADOR','PESQUERO','PASAJEROS','OCIO','ESTADO','MILITAR','SUBACUATICO']
ship_dimensions = ['a','b','c','d','draught']
for type in ship_types:
    for dimension in ship_dimensions:
        q = df[df['shiptype']==type][dimension].quantile(0.99)
        df[(df['shiptype']==type) & (df[dimension] > q)] = None

df_clean=df.dropna()

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

outlier_clean = df[df['shiptype'] == 'MERCANTE']

cols_to_plot = ['a', 'b', 'c', 'd', 'draught']

outlier_clean[cols_to_plot].boxplot()

plt.title('Distribución CLEAN MERCANTE')
plt.xlabel('Tipo de medición')
plt.ylabel('Valor de medición')

plt.show()

#ship_types =
['MERCANTE','PETROLERO','REMOLCADOR','PESQUERO','PASAJEROS','OCIO','ESTADO','MILITAR','SUBACUATICO']

df0=df_clean.filter(items=['mmsi'])
df0=df0.drop_duplicates()
df0

df_tr, df_ts = train_test_split(df0, random_state = 42)
df_tr
df_train=pd.merge(df_tr, df_clean)
df_train=df_train.drop('mmsi',axis=1)
df_train

df_test=pd.merge(df_ts, df_clean)
df_test=df_test.drop('mmsi',axis=1)
df_test

#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, train_size=0.75, random_state = 42)
#print(y_test)

X_OS=df_train.drop('shiptype', axis=1)
Y_OS=df_train['shiptype']
X_test_OS=df_test.drop('shiptype', axis=1)
Y_test_OS=df_test['shiptype']

#Representación de los datos de entrenamiento
gr = df_train['shiptype'].value_counts().reset_index()
gr = Y_OS.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_OS, Y_OS) #entrenamiento del modelo
y_pred = model.predict(X_test_OS) #evaluación del modelo
print(classification_report(Y_test_OS, y_pred, zero_division=0))

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=df_train.drop('shiptype', axis=1).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='r')
g.figure.set_size_inches(20, 5)
g.set_xticklabels(g.get_xticklabels(), rotation=90);
```

ANEXO IV: CÓDIGO DINAMIC

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

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

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
    #lineal
    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

#Funciones mas importantes para la creación de estadisticos
def statistics(df):
    #aceleración
    parametros = ['acc', 'sog', 'cog', 'sog_hat', 'sog_err', 'acc_hat', 'jerk_hat', 'jerk', 'alpha', 'zeta', 'omega']
    for x in parametros:
        df['avg_'+x] = df[x].mean()
        df['sdv_'+x] = df[x].std()
        if df[x].mean()!=0:
            df['iod_'+x] = df[x].var()/df[x].mean()
        else:
            df['iod_'+x] = 0
        df['q50_'+x] = df[x].quantile(0.50)
        df['skw_'+x] = df[x].skew()
        df['krt_'+x] = df[x].kurt()

df=df.drop('dtime', axis=1)
df=df.drop('nmi', axis=1)
```

```
df=df.drop('sog_hat', axis=1)
df=df.drop('sog_err', axis=1)
df=df.drop('acc_hat', axis=1)
df=df.drop('jerk_hat', axis=1)
df=df.drop('acc', axis=1)
df=df.drop('jerk', axis=1)
df=df.drop('omega', axis=1)
df=df.drop('alpha', axis=1)
df=df.drop('zeta', axis=1)
df=df.drop('lat', axis=1)
df=df.drop('lon', axis=1)
return df
```

```
if os.path.exists("DatosDinamicos.sql"):
    os.remove("DatosDinamicos.sql")
```

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

```
#establezco conexion con la base de datos (aqui cambio la base de datos)
path_data = r'DatosDinamicos.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()
```

```
#leo el archivo csv y reader produce dataframes con el tamaño de elementos establecido en chunksize
#Luego, se utiliza un bucle for para iterar sobre cada uno de estos DataFrames y realizar operaciones en ellos.
```

```
reader = pd.read_csv('datos_dinamicos1m.csv', sep=',', chunksize=1000000)
for df in tqdm(reader):
    chunk=df[orig]
    chunk.columns = subs
    chunk=chunk.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
    chunk=chunk.dropna()
    df = chunk[chunk['imo'] != 0]
    df = df[df['imo'] != "NULL"]
    df = df[df['sog'] > 0]
    df = df[df['sog'] < 130]
    df.to_sql('dinamic', con=conn, if_exists='append', index=False)
cur.close()
conn.close()
1it [00:08, 8.02s/it]
```

```
path_data = r'DatosDinamicos.sql'
conn = sqlite3.connect(path_data)
cur = conn.cursor()
data = pd.read_sql("""SELECT * FROM dinamic """, con=conn)
cur.close()
conn.close()
```

```
data.drop_duplicates()
sns.set(rc={'figure.figsize':(11.7,8.27)})
gr = data.shiptype.value_counts().reset_index()
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='b')
print(gr)
```

```
if os.path.exists("DatosCinematicos.sql"):
    os.remove("DatosCinematicos.sql")
path_sql_cinematics = r'DatosCinematicos.sql'
```

```
connection = sqlite3.connect(path_sql_cinematics)
curCinematics = connection.cursor()

imos = data['imo'].unique()
for i in tqdm(imos):
    ship = data[data['imo'] == i]
    cinematics_ship = cinematics(ship)
    cinematics_ship=cinematics_ship[cinematics_ship.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
    cinematics_ship=cinematics_ship.dropna()
    cinematics_ship=statistics(cinematics_ship)
    cinematics_ship=cinematics_ship.drop('tim', axis=1)
    cinematics_ship=cinematics_ship.drop('cog', axis=1)
    cinematics_ship=cinematics_ship.drop('sog', axis=1)
    cinematics_ship=cinematics_ship.drop_duplicates()
    #print(cinematics_ship)

    cinematics_ship.to_sql('cinematics', con=connection, if_exists='append', index=False)

curCinematics.close()
connection.close()

#opcion 1

path_sql_cinematics = r'DatosCinematicosRaul20230201.sql'
connection = sqlite3.connect(path_sql_cinematics)
curCinematics = connection.cursor()
data = pd.read_sql("""SELECT * FROM cinematics """, con=connection)
data.to_sql('cinematics', con=connection, if_exists='replace', index=False)
curCinematics.close()
connection.close()

path_sql_cinematics = r'DatosCinematicos2.sql'
connection = sqlite3.connect(path_sql_cinematics)
curCinematics = connection.cursor()
data = pd.read_sql("""SELECT * FROM cinematics """, con=connection)
data.to_sql('cinematics', con=connection, if_exists='replace', index=False)
curCinematics.close()
connection.close()

# lista de archivos de base de datos
db_files = ['DatosCinematicosRaul20230201.sql',
            'DatosCinematicosRaul20230202.sql',
            'DatosCinematicosRaul20230203.sql',
            'DatosCinematicosRaul20230204.sql',
            'DatosCinematicosRaul20230205.sql',
            'DatosCinematicosRaul20230206.sql',
            'DatosCinematicosRaul20230207.sql',
            'DatosCinematicosRaul20230208.sql',
            'DatosCinematicosRaul20230209.sql']

# lista vacía para almacenar los dataframes
df_list = []

# bucle para leer cada archivo de base de datos
for db_file in db_files:

    # establecer la conexión y leer los datos de la tabla "cinematics"
    connection = sqlite3.connect(db_file)
    df = pd.read_sql("""SELECT * FROM cinematics """, con=connection)
```

```
# cerrar la conexión y agregar el dataframe a la lista
connection.close()
df_list.append(df)

# combinar todos los dataframes en un solo dataframe
data = pd.concat(df_list)

data.drop_duplicates()
sns.set(rc={'figure.figsize':(11.7,8.27)})
gr = data.shiptype.value_counts().reset_index()
gr.columns = ['shiptype', 'freq']
g = sns.barplot(data=gr, x='shiptype', y='freq', color='b')
print(gr)

data=data[data.replace([np.inf, -np.inf], np.nan).notnull().all(axis=1)]
data=data[data.shiptype.isin(['MERCANTE','PETROLERO','PESQUERO','REMOLCADOR','PASAJEROS','MILITAR','OCI
O','SUBACUATICO','ESTADO'])]
X=data.drop('mmsi', axis=1) #Quitamos MMSI
X=X.drop('imo', axis=1) #Quitamos IMO
X=X.drop('loa', axis=1) #Variable estática (poner o no)
X=X.drop_duplicates()
y=X.shiptype #Variable dependiente
X=X.drop('shiptype', axis=1) #Variables independientes

#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, train_size=0.75, random_state = 42)
print(y_test)

X_OS=X_train
Y_OS=y_train
X_test_OS=X_test
Y_test_OS=y_test
#OverSampling

oversample=SMOTE(random_state = 42)
X_OS, Y_OS=oversample.fit_resample(X_train, y_train)

#Undersampling
under = RandomUnderSampler()
X_OS, Y_OS=under.fit_resample(X_train, y_train)

#Random Forest
pipe = Pipeline([(['scaler', MinMaxScaler()]),(['classifier',
RandomForestClassifier(n_estimators=100)])])
model = pipe.fit(X_OS, Y_OS) #entrenamiento del modelo
y_pred = model.predict(X_test_OS) #evaluación del modelo
print(classification_report(Y_test_OS, y_pred, zero_division=0))

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_train.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);
```

