Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Estudio y optimización de un sistema de inteligencia artificial para el análisis de tráfico marítimo y aéreo empleando datos AIS y ADS-B

Grado en Ingeniería Mecánica

ALUMNO: Santiago Martín Marco

DIRECTOR: Miguel Rodelgo Lacruz

CURSO ACADÉMICO: 2016-2017

Universidad de Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Estudio y optimización de un sistema de inteligencia artificial para el análisis de tráfico marítimo y aéreo empleando datos AIS y ADS-B

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

La inteligencia artificial está en el momento más fructífero desde su origen con avances en diversos ámbitos. En el presente trabajo se pretende crear un sistema de inteligencia artificial para analizar el comportamiento de buques y aeronaves y así obtener sus patrones de movimiento. El objetivo es detectar aquellos que presenten una conducta anómala con respecto al resto del tráfico para aumentar la seguridad de la navegación identificando aquellos que tengan fallos mecánicos, se encuentren en peligro o puedan estar realizando actividades ilegales. Esto nos permitirá actuar prontamente con las medidas oportunas.

Para desarrollarlo se ha creado una aplicación en Python utilizando la inteligencia artificial de Numenta, NuPIC, que implementa los algoritmos de aprendizaje HTM, un nuevo concepto de red neuronal artificial basada en la memoria temporal y jerárquica que recrea el funcionamiento del neocórtex del cerebro. La recopilación de datos de buques se realiza a través de la API de MarineTraffic *for AIS Data* y los datos de aeronaves recibiendo señales ADS-B mediante un receptor SDR. Una vez obtenidos los datos se realiza la optimización de los parámetros de la red neuronal empleando distintas técnicas, como el F-Score. Finalmente se presentan los resultados obtenidos del análisis en archivos KML para visualizarlos con Google Earth.

PALABRAS CLAVE

Inteligencia artificial, Numenta, HTM, Python, Automatic Dependent Surveillance – Broadcast (ADS-B), Automatic Information System (AIS).

AGRADECIMIENTOS

A mis padres por su apoyo y ayuda durante estos intensos meses y, especialmente, a mi padre, por ayudarme con la recepción de datos.

A Francisco Enguix López por sus soluciones en momentos cruciales.

A mi tutor Don Miguel Rodelgo Lacruz por su ayuda e implicación en el proyecto.

A Matthew Taylor, *Open Source Community Flag-Beare* de Numenta, por la resolución de diferentes problemas que surgieron durante el desarrollo.

Por último a una persona, porque sin su ayuda este trabajo no habría sido igual.

Contenido

Índice de Figuras

Índice de Tablas

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contextualización y motivación de este trabajo

A lo largo de la historia el transporte ha sido el motor del desarrollo económico y social. En la actualidad, con el sistema económico mundial cada vez más enfocado hacia la globalización, la dependencia del transporte no ha hecho otra cosa que aumentar.

Hasta la invención del avión y su generalización a nivel mundial, el transporte por vía marítima era la única vía capaz de unir continentes. La proliferación de estos dos medios de comunicación lleva consigo la necesidad de una coordinación y control de ambos espacios, aéreo y marítimo.

En este contexto, el transporte marítimo es en la actualidad una parte esencial para el correcto funcionamiento del mundo tal cual lo conocemos. Como dijo Kitack Lim, secretario general de la Organización Marítima Internacional (OMI) "el transporte marítimo es la espina dorsal del comercio internacional" [1]. De acuerdo con la OMI, en la actualidad, se fija que alrededor del 80 % del volumen de transporte de mercancías a nivel mundial y más del 70 % de su valor se realiza por vía marítima [2]. Por lo tanto, actualmente sería imposible satisfacer las necesidades de todas las regiones del mundo sin el transporte marítimo.

En los últimos años la importancia de este sector continúa aumentado (véase Figura 1-1). En 2015 la flota mundial contaba con más de 90.000 buques de transporte de mercancías y más de un millón de marineros. La tendencia en un futuro es que continúe aumentando tanto el tamaño de la flota mercante como el número de personas que trabajan en el sector.

Por ello, a lo largo del siglo XIX se crearon diferentes organismos internacionales, como la ya citada OMI, que establecieron un marco legal común a nivel internacional para homogeneizar el transporte marítimo mediante distintos convenios y tratados. Sin embargo, como no podía ser de otra manera, el control y vigilancia del correcto desarrollo de las operaciones marítimas se ejerce por los propios estados ribereños en sus áreas de responsabilidad.

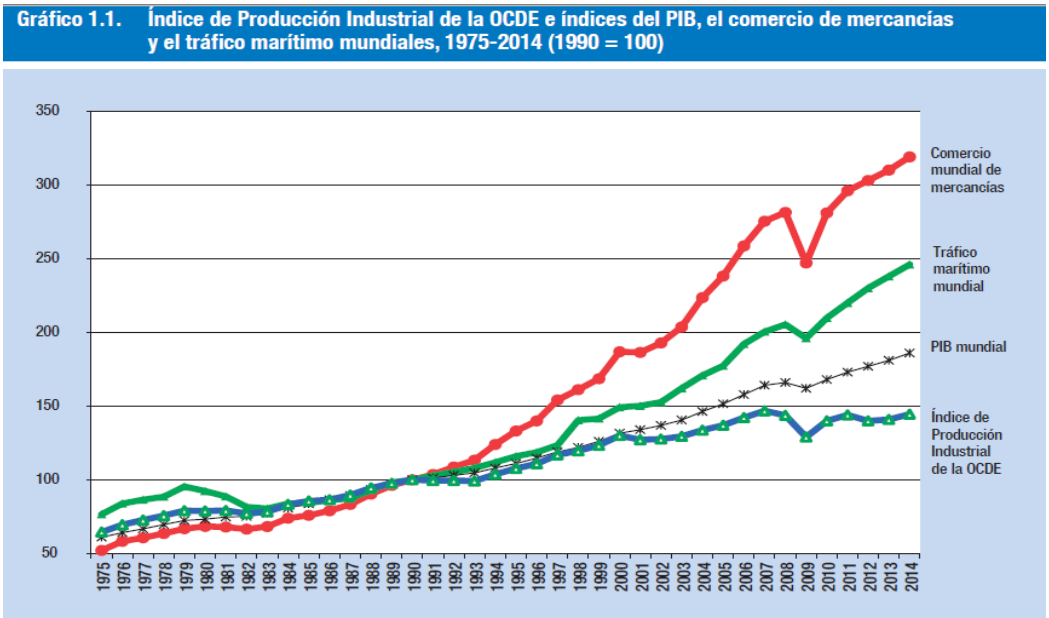

Figura 1-1 Comercio de mercancías y tráfico marítimo mundiales, 1975-2014 (fuente: [3]).

Como podemos ver en la Figura 1-2, España es un país notablemente importante en el control del transporte marítimo a nivel mundial, no solo por los buques que recibe sino también por los que pasan por las zonas de responsabilidad españolas.

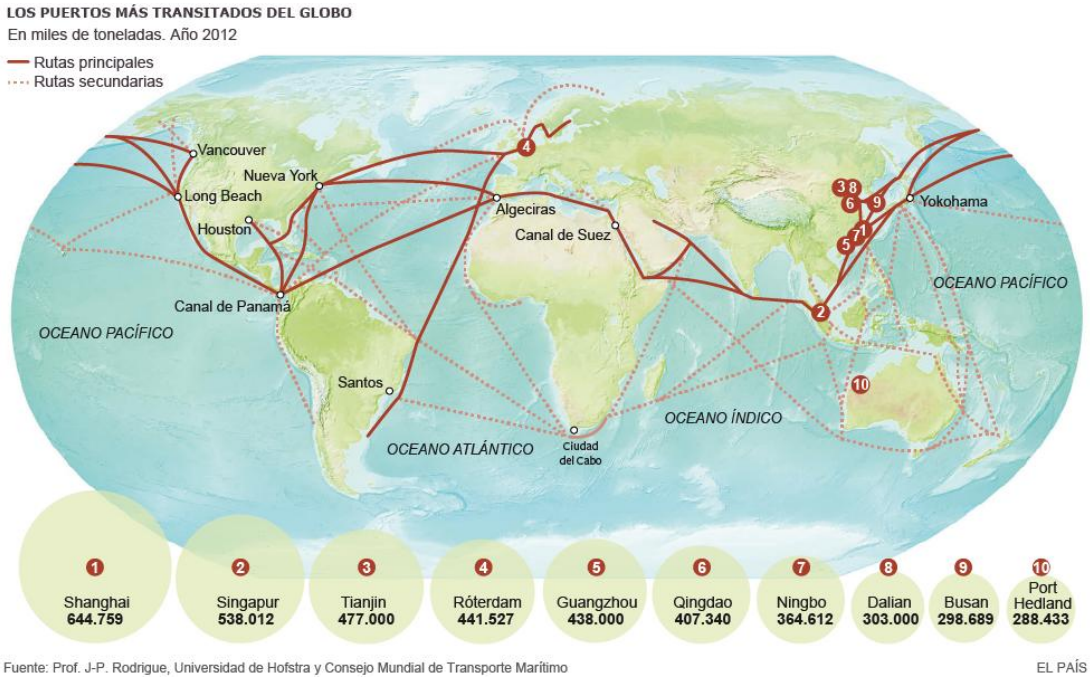

Figura 1-2 Principales rutas marítimas año 2012 (fuente: [4]).

España tiene una tradición marítima muy importante. Una de estas razones es su condición geográfica. Cuenta con casi 8000 km de costas en las cuales encontramos 46 puertos dentro del Sistema Portuario de Titularidad Estatal (véase Figura 1-3), es decir, sin tener en cuenta los puertos menores, como los deportivos. Según el Ministerio de Fomento [5], en el año 2014 se registraron 139.000 entradas de buques entre estos puertos.

Además, se estima que los Centros de Coordinación de Salvamento Marítimo [6] controlan anualmente en torno a 390.000 buques y atienden miles de emergencias a lo largo del año (en 2012 se atendieron 5.067, véase Figura 1-3).

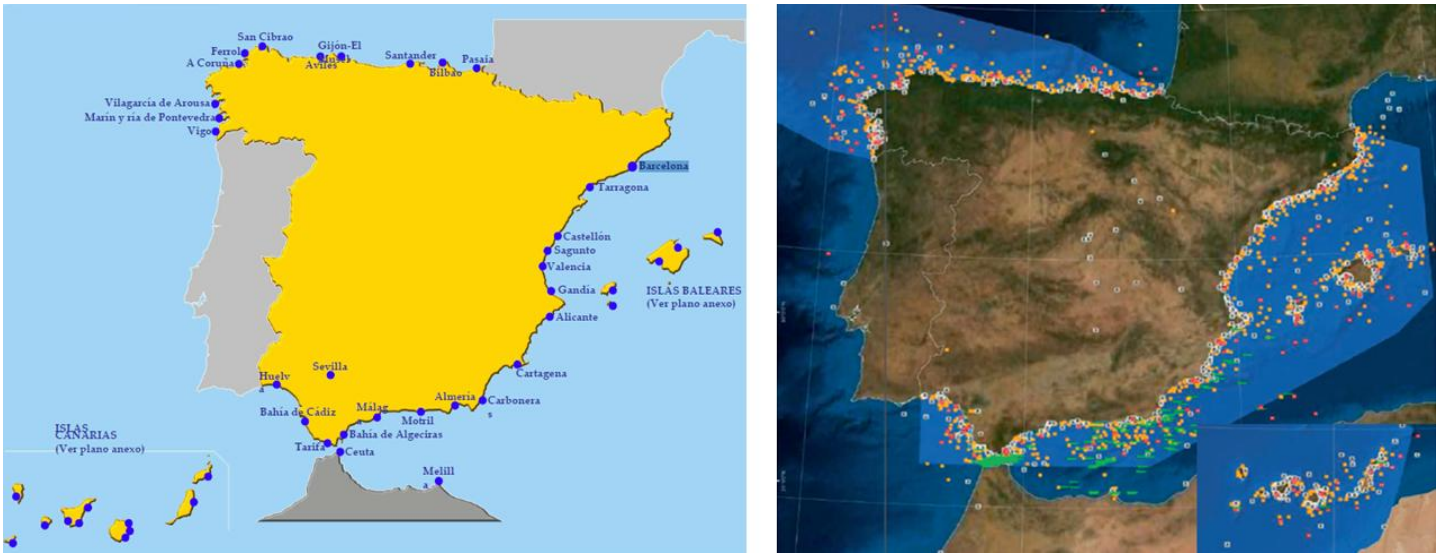

Figura 1-3 Sistema Portuario Titularidad Estatal (izquierda) y Emergencias Salvamento Marítimo 2012 (derecha) (fuente: [7] y [8]).

Debido a la importancia económica y estratégica del medio marino es necesaria una supervisión exhaustiva y permanente. El control y vigilancia en España se realiza, principalmente, por 5 organismos de manera paralela: Salvamento Marítimo mediante los 19 Centros de Coordinación y Salvamento ubicados por toda la costa española; la Guardia Civil del Mar, cuerpo específico de la Guardia Civil; el Servicio de Vigilancia Aduanera, perteneciente a la Agencia Tributaria; la Dirección General de la Marina Mercante, a través de las diferentes Capitanías Marítimas; y por último la Armada Española, mediante sus buques coordinados por el Centro de Operaciones y Vigilancia de Acción Marítima (COVAM) situado en Cartagena. En el espacio marítimo, excepto en contados puntos, no existen vías obligatorias de circulación.

A diferencia de esto, en el espacio aéreo todas las aeronaves deben circular por unas rutas prefijadas denominadas aerovías.

En España las competencias de control del espacio aéreo están mucho menos divididas. Se diferencian dos partes en el concepto de control y vigilancia [9]. Por una parte la seguridad, ordenación y fluidez de la circulación aérea, consistente en dirigir las aeronaves por las citadas rutas hacia los aeropuertos además de proporcionar servicios de información de vuelo y alerta. De estas funciones se encarga la entidad pública empresarial adscrita al Ministerio de Fomento ENAIRE [10] a través de sus 5 centros de control en Madrid, Barcelona, Gran Canaria, Palma de Mallorca y Sevilla y las 22 torres de control distribuidas por el territorio nacional.

La segunda parte es la que se refiere a ejercer la soberanía en el espacio aéreo español, misión que es llevada a cabo por el Ejército del Aire. Para ello cuenta con 13 Escuadrones de Vigilancia Aérea

(EVA) distribuidos por el territorio [11], once de los cuales se localizan en la península ibérica e islas Baleares (véase Figura 1-4) y dos en las islas Canarias. Los EVA dependen del Mando Aéreo de Combate (MACOM) y su función es la detección y seguimiento de las aeronaves en su zona de responsabilidad y la transmisión de estos datos tanto a los dos centros de Defensa Aérea (Grupo Central de Mando y Control y Grupo Norte de Mando y Control), como a los centro de aviación civil mencionados anteriormente.

Figura 1-4 EVA en la península ibérica (fuente: [12]).

A continuación se presentan dos imágenes (véase Figura 1-5) del día 15 de febrero de 2017 a la hora del mediodía de dos sistemas de información automática, uno marítimo y otro aéreo (AIS y ADS-B respectivamente), como prueba de la saturación del tráfico en la península ibérica.

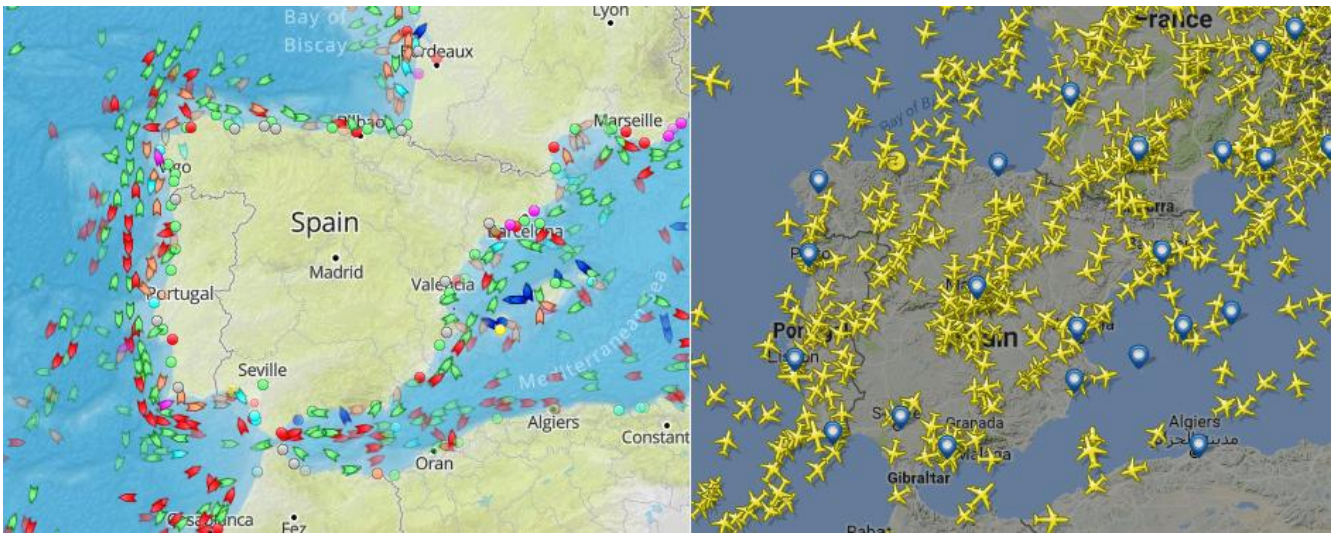

Figura 1-5 Distribución del tráfico marítimo (izquierda) y aéreo (derecha) en la península ibérica (fuente: [30] y [29]).

El papel que realizan estos organismos es, por tanto, crucial para detectar y minimizar cualquier problemática en ambos medios. Pero su gran extensión y el número de buques y aviones que

desarrollan diferentes actividades hacen compleja su vigilancia y control. Por ello en este trabajo se propone una forma de mejorarlo y optimizarlo.

Tanto en aeronaves como Hay multitud de sistemas que envían información como la posición, rumbo y velocidad a los sistemas de control tanto en aeronaves como en buques, los cuales están siendo implantados por distintas naciones y en un futuro lo estarán a nivel mundial. El problema reside en cómo procesar toda esta información de una manera rápida y eficiente de forma que nos permita obtener conclusiones e información de interés prácticamente en tiempo real. Actualmente esta función se realiza en los centros de control por personas especialmente dedicadas a ello. Una forma de mejorar y optimizar la forma en que se procesa esta información es por medio de la inteligencia artificial.

Aunque han transcurrido más de seis décadas desde que se formalizo el concepto, se podría decir que actualmente la inteligencia artificial aún se encuentra "en pañales". Sin embargo, la nueva tecnología ya supera ampliamente las capacidades de los seres humanos en determinadas funciones. Uno de estos aspectos es el reconocimiento de patrones y la capacidad de adelantarse a los mismos, por ejemplo, a los movimientos de un barco o aeronave en base a lo que se considera como un patrón de comportamiento normal en una determinada zona.

Los sistemas de inteligencia artificial nos permiten procesar de manera rápida, segura, y lo que es más importante, de una forma útil estos datos, para así obtener información valiosa.

1.2 Objetivos

En [13] se estudia la viabilidad de aplicar un sistema de inteligencia artificial al análisis del tráfico marítimo. Los datos se obtienen a partir de mensajes AIS que se reciben a través de la *API for AIS Data* de MarineTraffic y posteriormente se representan para su visualización. El estudio, de carácter preliminar, obtiene como resultado la posibilidad de aplicar sistemas de inteligencia artificial, aunque el desarrollo del mismo es limitado. Este trabajo pretende optimizar este sistema de inteligencia artificial para el ámbito marítimo y a continuación aplicarlo al análisis de las rutas de aviones.

En primer lugar, la optimización del sistema mejorará el aprendizaje automático de los diferentes patrones que rigen el tráfico marítimo en general y en particular aquellos referidos al transporte de mercancías internacional. Esto permitirá detectar rápidamente aquellos buques que saliéndose de los patrones de comportamiento puedan estar en peligro o puedan representar una amenaza potencial, como por ejemplo transporte ilegal de mercancías.

Para ello se utiliza la aplicación de inteligencia artificial (IA) desarrollada por Numenta denominada NuPIC (*Numenta Platform for Intelligent Computing*) que implementa los algoritmos HTM (*Hierarchical Temporal Memory*). La obtención de datos se realiza gracias a la información que cada buque envía mediante el sistema de información automática (AIS, por sus siglas en inglés) y se tiene acceso a ellos a través del servidor de MarineTraffic.

El análisis de los datos presenta diversas complicaciones como son el preprocesado de los mismos y la selección de los parámetros de la red neuronal. Por esto, se pretende realizar un análisis previo de los datos con el fin de seleccionar aquellos válidos y utilizar una técnica conocida como *swarming* para la selección óptima de los parámetros. Así se solucionarán diversos errores a la hora de detectar las anomalías para obtener unas ratios de detección más elevados, es decir, disminuir los falsos positivos y los falsos negativos.

Por otra parte, se pretende ampliar el uso de este sistema adaptándolo para su aplicación en el ámbito aéreo, un entorno tridimensional, aunque con la ventaja de que las rutas que siguen los aviones son mucho más regulares que las de los buques. En este caso, los datos se obtienen a partir de un sistema civil de identificación y vigilancia aérea denominado ADS-B.

Para cumplir con estos propósitos generales se plantean los siguientes objetivos específicos:

-Realizar un estudio de las características generales de los datos, para comprobar la idoneidad de utilizar la tecnología de Numenta y posteriormente decidir cuál es la plataforma óptima para la implementación del programa.

-Instalación de la aplicación (NuPIC), adaptación de la misma a los datos que pretendemos analizar, tanto para el caso de los buques como para las aeronaves.

-Llevar a cabo un preprocesado exhaustivo para eliminar las incoherencias que los datos puedan contener.

-Optimizar los parámetros del modelo para mejorar la detección de anomalías utilizando la técnica de *swarming*.

-Obtención de datos AIS de buques. Estudio de las diferentes formas de obtención de datos del sistema ADS-B de aeronaves y obtención de los mismos.

-Representación gráfica de los resultados, como forma más visual de estudio e identificación de las anomalías.

1.3 Marco geográfico

Se pretende que el sistema sea aplicable a cualquier área marítima que se quiera supervisar, pero como es lógico nos hemos centrado en una zona concreta para poder realizar el estudio y optimización.

Para los datos obtenidos desde MarineTraffic se ha decidido utilizar los correspondientes a la zona del Dispositivo de Separación de Tráfico (DST) de Finisterre (véase Figura 1-6) por la regularidad y el comportamiento del tráfico en el mismo. Según [14] los DST son vías de circulación que se establecen donde el tráfico marítimo es muy denso y su objetivo principal es evitar los abordajes. En los DST el control de los buques se realiza más de cerca por los sistemas de control de tráfico. Este DST es el segundo más transitado de España, detrás del DST del estrecho de Gibraltar.

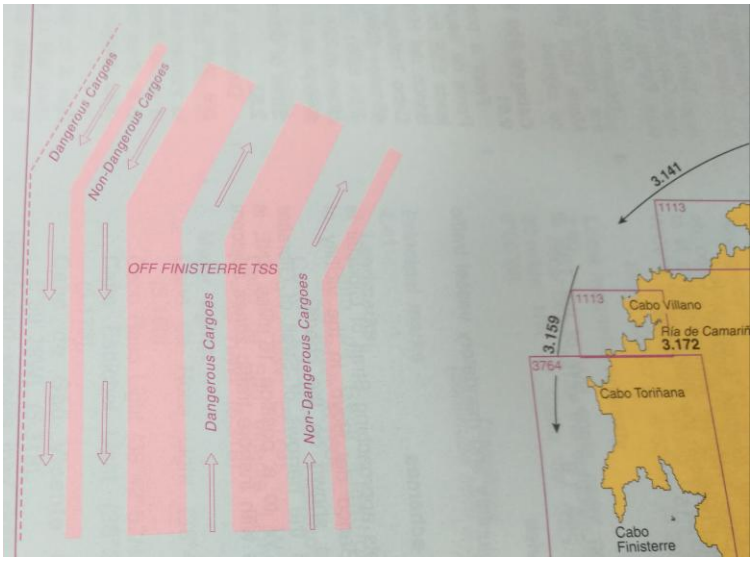

Figura 1-6 DST de Finisterre (fuente: [15]).

En cuanto a los datos de las aeronaves por la propia naturaleza del método de obtención de los mismos, la zona corresponde al alcance del receptor SDR[1]. En concreto la zona abarca hasta unos 230 kilómetros de radio con centro en la Playa de la Lanzada, situada cerca de la entrada de la Ría de Pontevedra. Como se puede ver en la imagen hay cinco aeropuertos en el radio de cobertura, aunque la mayor parte de ellos no tienen un tráfico excesivamente denso.

Figura 1-7 Zona de cobertura receptor SDR (fuente: [16]).

1.4 Organización de la memoria

A continuación se presenta cómo se desarrolla la memoria, explicando brevemente los contenidos de cada sección.

En la Sección 1 se introduce el tema que se va a desarrollar en el TFG y las razones que llevan a su planteamiento. Posteriormente se realiza una explicación de los objetivos que se plantean y su desarrollo.

En la Sección 2 se plasma brevemente la tecnología con la que se va a trabajar: su definición, nacimiento y desarrollo de la IA. Se explica el estado actual de la tecnología y se introduce en concreto

[1] Receptor SDR: sistema de radiocomunicaciones en el cuál parte de los componentes físicos propiamente de las radios tradicionales se han substituido por *software*.

el tipo de redes neuronales que se van a utilizar (HTM) y la aplicación desarrollada por Numenta de código abierto (NuPIC) para implementarlas. Seguidamente se describen los sistemas (AIS y ADS-B) que envían la información desde cada medio de transporte. Y por último, se nombran los diferentes sistemas que se han desarrollado hasta la fecha, primero relacionados con el control del ámbito naval y después sobre el aéreo.

En la Sección 3 se explica por qué se han seleccionado los diferentes sistemas que se van a utilizar y se desarrolla la instalación de la IA (NuPIC) y de la aplicación *geospatial*. Posteriormente se describe cómo se han obtenido datos: en el caso de los buques mediante la API de MarineTraffic y en el de aeronaves a través de un receptor SDR recibiendo datos ADS-B. A continuación se explica cómo se ejecuta el programa, su funcionamiento y los diferentes módulos que intervienen. Tras esto, se describe la implementación del *swarming* y cómo se ha realizado la selección de los parámetros óptimos del modelo. Por último, se describe de qué manera se ha llevado a cabo la representación gráfica de los resultados.

En la Sección 4 se muestran y comentan los resultados obtenidos tras realizar la optimización y seguidamente se presentan visualmente los resultados. Primero referidos al ámbito marítimo y después al aéreo valorándose la aplicación en cada aspecto.

En la Sección 5 se plasman las conclusiones obtenidas del funcionamiento de la aplicación, así como posibles mejoras e implementaciones futuras.

En la Sección 6 se citan referencias bibliográficas y recursos consultados para la realización del trabajo.

Finalmente se adjuntan 12 anexos:

En el Anexo I se muestran las siglas y acrónimos citados y utilizados en el trabajo.
En el Anexo II se realiza una descripción del receptor SDR y la mejora llevada a cabo en la antena.
Del Anexo III al VIII se adjunta el código en Python de los diferentes módulos de la aplicación.
Del Anexo IX al XII se encuentran ejemplos de los ficheros de entrada y salida de ambas aplicaciones.

2 ESTADO DEL ARTE

2.1 Inteligencia Artificial

2.1.1 *Definición*

La inteligencia artificial se encuentra englobada dentro de las ciencias de la computación. Hay diversas formas de definir que es la inteligencia artificial, además, desde que surgió el concepto su definición ha ido evolucionando así como lo hacía la tecnología. Raymond Kurzweil definió la inteligencia artificial en 1990 como: "El arte de crear máquinas que realizan funciones, las cuales requerirían de inteligencia al ser realizadas por una persona" [17]. Raymond Kurzweil es científico especializado en ciencias de la computación e inteligencia artificial y desde el 2012 es el director de ingeniería de Google, empresa puntera en inteligencia artificial y robótica.

Tras esta definición es necesario mencionar un concepto que a veces se utiliza de manera no precisa. El aprendizaje automático, *machine learning* en inglés, es la rama de la inteligencia artificial que se dedica al estudio de los agentes o programas que aprenden o evolucionan basados en la experiencia, para así efectuar una tarea determinada cada vez mejor [18].

2.1.2 *Origen e historia*

Los primeros avances que posteriormente dieron lugar a la inteligencia artificial tuvieron lugar en 1943 cuando Warren S. McCulloch y Walter H. Pitts propusieron uno de los primeros modelos matemáticos de la neurona del cerebro humano y animal [19].

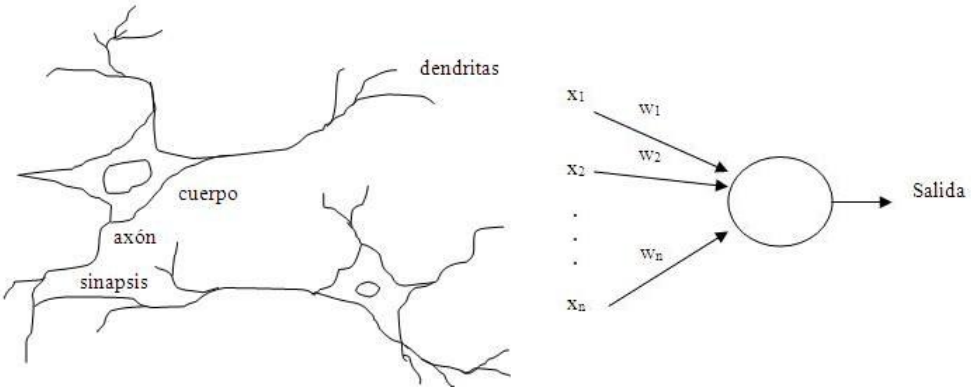

Figura 2-1 Neurona biológica y neurona artificial (fuente: [15]).

En 1950 Alan Turing pública su artículo "Computing Machinery and Intelligence" [20], en el que se profundiza sobre la posibilidad de que una máquina imite el comportamiento de la mente humana. Además, propone el "Test de Turing" para determinar si una máquina es inteligente o no. En el documento se recogían los diferentes conocimientos de la época en este ámbito, pero sería más tarde cuando se acuñaría el término "Inteligencia Artificial".

Esto ocurrió en la Conferencia de Dartmouth que tuvo lugar en 1956 [21] y [22]. En la conferencia, que duró dos meses, se definieron las bases de esta nueva ciencia y las futuras líneas de actuación. La premisa de la misma era: "Todo aspecto de aprendizaje o cualquier otra característica de inteligencia puede ser definido de forma tan precisa que puede construirse una máquina para simularlo".

Tras este gran hito, la nueva ciencia entra en una edad de oro con innumerables avances, lo que lleva cada vez a más organismos y empresas a financiar proyectos en este ámbito. Algunos de estos avances son el "Peceptron" de F. Rosenblatt, un modelo de red neuronal con aprendizaje supervisado en 1958 [23], o "Eliza" [24], que es considerado como uno de los primeros *bots* conversacionales diseñado entre 1964 y 1966 en el Instituto Tecnológico de Massachusetts. En esta dirección de internet [25] se puede probar a tener una conversación con este *chatbot*.

Los avances, aunque fueron rápidos, no eran lo suficientemente importantes, y al no cumplir las expectativas de los inversores, la financiación se fue reduciendo. Esto fue debido a las limitadas capacidades de procesamiento de los ordenadores y a la complejidad de "programar" ciertas habilidades en los sistemas de IA como las capacidades motoras o las sensoriales. De esta manera en torno a 1974 se entra en un periodo de decadencia denominado primer invierno que duraría hasta principios de los ochenta.

En esta década y debido al aumento de la capacidad de computación de los ordenadores se produce un nuevo *boom* de la IA. Esto propicia la aparición de los Sistemas Expertos [26], orientados a la aplicación práctica de los conocimientos, y del ambicioso proyecto llevado a cabo por Japón llamado "la quinta generación". En este proyecto se pretendía el desarrollo de una nueva clase de ordenadores que utilizarían técnicas de IA, tanto en el plano del *software* como en el del *hardware*, obteniendo resultados asombrosos como la traducción automática. Sin embargo, ninguno de estos proyectos tuvo el resultado esperado, lo que sumió de nuevo a la inteligencia artificial en otro periodo de falta de financiación e interés por la comunidad en general. Este nuevo "invierno" se extenderá prácticamente hasta finales del siglo XX.

A pesar de este declive tenemos pruebas de la importancia y transcendencia que estaba teniendo la inteligencia artificial con el conocido hecho de 1997 en el que "Deep Blue" (véase Figura 2-2), un superordenador (para aquella época), ganó al entonces campeón de ajedrez del mundo Gari Kasparov [27] .

Figura 2-2 Deep Blue (izquierda) contra Gari Kasparov (derecha) (fuente: [28]).

2.1.3 *IA en la actualidad*

En la actualidad, como ya plantearon en su momento Warren S. McCulloch y Walter H. Pitts, la corriente de investigación y desarrollo sigue siendo la de intentar copiar el funcionamiento del cerebro humano cada vez de manera más precisa para poder recrear en las máquinas una de las capacidades animales y humanas básicas, el aprendizaje. Es realmente complicado y en ocasiones imposible programar determinadas conductas, como por ejemplo el equilibrio. Por ello, una de las mejores estrategias para lograr que una máquina mantenga el equilibrio es dándole las herramientas (sensores, etc.) y la capacidad de aprender de los intentos realizados. Esto se traduce en mejorar los modelos cognitivos actuales para así aumentar sus capacidades.

Existen diversos enfoques en la aplicación del aprendizaje automático. Uno de ellos son las redes neuronales artificiales (en inglés ANN, *Artificial Neural Networks*) que ahora vuelven a estar de actualidad con el resurgimiento del aprendizaje profundo [29]. Este enfoque se centra en crear "neuronas artificiales" y mediante la conexión o inhibición de multitud de ellas recrear la estructura del cerebro biológico. Copiando la estructura se busca imitar su comportamiento para que, en vez de enseñarle al ordenador una lista inacabable de reglas o normas para resolver un problema o actuar de una determinada manera, se le dé un modelo con el que pueda evaluar las situaciones que se le presenten y una pequeña serie de instrucciones para mejorarlo en base a los errores obtenidos. De esta forma con el tiempo se volverá cada vez más preciso a la hora de resolver el problema, gracias a la habilidad de estos sistemas de aprender de la experiencia [30].

La inteligencia artificial se está aplicando prácticamente en cualquier campo que se nos pueda ocurrir. Desde aplicaciones relacionadas con la seguridad (en criminología para análisis del comportamiento humano) o militares (mediante el uso de vehículos no tripulados) hasta en el ámbito de la música (terminar partituras inacabadas por su autor), pasando por la medicina (para el diagnóstico de enfermedades), la industria (robótica y optimización de los sistemas de producción), las telecomunicaciones, los videojuegos y por último en el ámbito social, como son el reconocimiento de voz, texto, o imágenes, para recomendarnos productos que pueden ser de nuestro agrado, etc.

Relacionado con las numerosas aplicaciones en el ámbito social se puede apreciar un cambio en las empresas punteras en IA. En la década de los noventa estas empresas eran aquellas relacionadas con el mundo de la computación, como IBM. Sin embargo hoy en día el relevo lo están tomado aquellas relacionadas con el ámbito social, que además, tienen un enorme acceso a datos personales de usuarios, como pueden ser Facebook o Google. En este sentido y sin que muchos seamos totalmente conscientes, en nuestra vida diaria interactuamos con sistemas de inteligencia artificial que nos hacen la vida más cómoda.

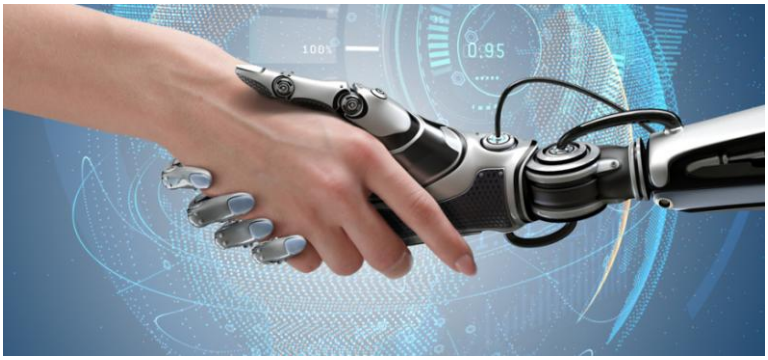

Figura 2-3 La inteligencia artificial en nuestras vidas (fuente: [31]).

2.1.4 *Clasificación de la inteligencia artificial*

Existen múltiples formas de clasificar las diferentes ramas de la IA. Una de las más comunes es teniendo en cuenta los métodos y técnicas que desarrollan y aplican. Así lo clasifican los autores Stuart J. Russell and Peter Norvig en su libro *Artificial Intelligence: A Modern Approach* [17] diferenciando seis técnicas [32]:

- **Resolución de problemas y búsqueda de soluciones:** el objetivo es resolver problemas de índole muy variada. Para ello se centra en cómo poder formalizar estos problemas. Dentro de este grupo encontraríamos diferentes aplicaciones como la de crear una IA que jugará al ajedrez (Deep Blue), encontrar la ruta más corta para ir a un determinado lugar o encontrar la ruta más eficiente teniendo que parar en un número determinado de sitios.

- **Actuación lógica y planeamiento:** este caso sería similar a la resolución de problemas pero serviría para aquellos muchos más complejos donde los diversos *inputs* se van modificando con el tiempo y dependiendo de cómo se actúe.

- **Sistemas basados en conocimiento y razonamiento lógico**: es frecuente que se necesite aplicar conocimientos del medio para poder resolver el problema. Se englobarían aquí todas aquellas aplicaciones que necesiten un razonamiento lógico. Tiene aplicación en el campo de la medicina, en el diagnóstico de enfermedades, y en las finanzas, para saber si un cliente cumple los estándares para recibir un crédito. Un ejemplo muy básico de esta técnica sería: si Lassie es un perro y un perro es un animal, entonces Lassie es un animal.

- **Conocimiento y razonamiento bajo incertidumbre:** como suele ocurrir mediante una decisión o un planeamiento no siempre se consiguen los objetivos preconcebidos. Este método puede englobar a los anteriores y tendría en cuenta estos factores.

- **Comunicación, percepción y acción:** aquí se englobarían todas las técnicas relacionadas con reconocimiento de imágenes y texto, reconocimiento de voz y todas aquellas relacionadas con el movimiento y la robótica.

- **Aprendizaje:** se refiere a la capacidad autónoma por la que la máquina aprende de la experiencia, de esta forma el programa aprendería de sus propios errores para cada vez responder de manera más correcta. También se conoce como *machine learning* o aprendizaje automático.

Por lo tanto es común que las aplicaciones que se desarrollan utilicen simultáneamente varias de estas técnicas para realizar lo que se pretende que hagan.

Dentro del aprendizaje automático encontramos dos tipos en función de la forma de entrenar la red neuronal:

-Aprendizaje supervisado: en estos sistemas se realiza el aprendizaje sobre un conjunto de datos pre-etiquetados. De manera que el sistema ajusta los pesos de sus parámetros internos en función de ellos para devolver la respuesta deseada (véase Figura 2-4) [33]. Tras este entrenamiento se pretende que el sistema actúe correctamente al recibir los datos del conjunto a estudiar.

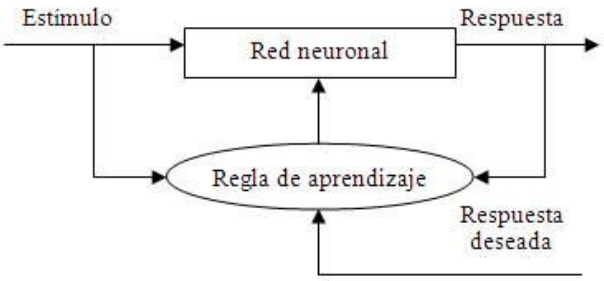

Figura 2-4 Aprendizaje supervisado (fuente: [34]).

-Aprendizaje no supervisado: en este aprendizaje no se dispone de un conjunto de datos etiquetados. Se le pasa al sistema el conjunto de datos para que inicie un proceso iterativo de categorización de la información (*clustering*).

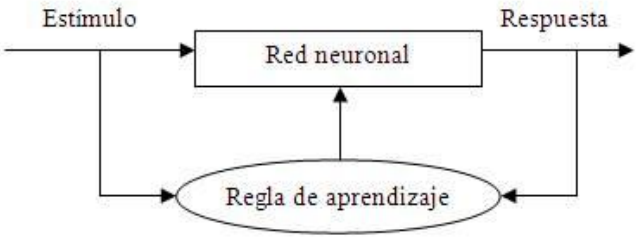

Figura 2-5 Aprendizaje no supervisado (fuente: [34]).

2.2 Numenta y HTM (*Hierarchical Temporal Memory*)

Numenta es una empresa, con base en California, fundada en 2005 por Jeff Hawkins, Donna Dubinsky y por Dileep George. Jeff Hawkins es el fundador de Palm, compañía de PDA's (*Personal Digital Assistants*) y al año siguiente de publicar su libro "On Intelligence" [35] fundó Numenta. Esta compañía define su misión como: "ser los líderes en la nueva era de la inteligencia de las maquinas" [36].

Jeff Hawkins explica su teoría en el libro, y más tarde funda Numenta para ponerla en práctica creando los algoritmos HTM. Su teoría está basada en que entender cómo funciona el neocórtex, es la manera más rápida de crear máquinas inteligentes. Pretende realizar ingeniería inversa del mismo, para poder entender cómo funciona exactamente y crear *software* que tenga estos principios como forma de funcionamiento.

Como el propio Jeff Hawkins señala, las HTMs son un tipo de red neuronal artificial con aprendizaje supervisado. Programarlas es diferente a la programación de ordenadores tradicional. Actualmente los programadores crean programas específicos para resolver problemas específicos. Por el contrario, las HTMs son entrenadas a través de la exposición a una corriente de datos sensoriales. Por lo tanto, las capacidades de las HTMs están determinadas en gran parte por los datos a los que han estado expuestas [37].

2.2.1 *HTM (Hierarchical Temporal Memory)*

La tecnología HTM se ha desarrollado como modelo de referencia para su inteligencia artificial de *online machine learning* (aprendizaje automático en línea). En su núcleo se encuentran los algoritmos de aprendizaje para patrones basados en el tiempo, que a diferencia de otros modelos no requieren de conjuntos de datos de entrenamiento o que estos estén marcados [38].

HTM es un sistema basado en memoria. La forma en que los datos se almacenan y cómo se accede a ellos es lógicamente diferente al modelo estándar usado por los programadores. La memoria tradicional de un ordenador tiene una organización plana y no tiene noción inherente del tiempo. Un programador puede implementar cualquier tipo de estructura y organización de datos sobre la memoria "plana" (tradicional) de un ordenador. Además, tienen control sobre cómo y dónde se almacena la información. Por el contrario, la memoria HTM es más restrictiva. Tiene una organización jerárquica y está inherentemente basada en el tiempo. Además, esta siempre se almacena de manera distribuida. Un usuario de HTM especifica el tamaño de la jerarquía y para qué entrenar el sistema, pero HTM controla dónde y cómo se almacena la información [37].

A continuación se presentan las características que deben cumplir los datos con los que mejor funciona HTM [36]:

- Transmisión continúa de datos en lugar de datos por lotes. Quiere decir aprendizaje en línea, los datos se transmiten en orden secuencial de manera que la IA aprende de forma continua haciéndose cada vez más precisa. Además, esto permite que el algoritmo se adapte dinámicamente a los patrones de los datos en caso que estos se modifiquen.

- Datos con patrones basados en el tiempo. Como ya se dijo anteriormente por la manera en que se obtienen los patrones, la variable temporal es sumamente importante. Esto hace que los datos que utilicemos deban tener un patrón relacionado con el tiempo, lo que no es lo mismo que saber el tiempo en que se obtuvo cada dato.

- Multitud de fuentes de datos individuales en los que elaborar manualmente modelos separados no es práctico. Debido a la forma de los datos es necesario crear modelos relativamente independientes para cada uno de los conjuntos de ellos. Por lo tanto, no es viable realizarlo manualmente.

- Datos con patrones sutiles que no siempre pueden ser vistos por las personas. Las máquinas son mucho más precisas a la hora de realizar un determinado análisis.

- Datos para los cuales técnicas simples como umbrales producen falsos positivos y falsos negativos. La complejidad de los datos debido a la relación que las diferentes variables tienen no es conocida o es compleja, lo que imposibilita aplicar simples umbrales para estas.

A continuación se explican las bases y funcionamiento de las HTMs.

Sus autores en [37] definen las HTMs como un tipo de red neuronal, ya que intenta modelar la arquitectura del neocórtex. Los modelos de neurona HTM se denominan células (*cells*) y están organizadas por columnas, capas, regiones y según una jerarquía. Las células HTM se organizan y funcionan en base a diferentes principios.

Una red HTM consiste en regiones organizadas jerárquicamente. La región es la unidad principal de memoria y predicción y cada región representa un nivel en la jerarquía. Conforme se asciende en la jerarquía hay convergencia, es decir, diversos elementos de una región inferior convergen en un elemento de la región superior (véase Figura 2-6). A la vez, debido a las conexiones de retroalimentación, al ir descendiendo en la jerarquía la información diverge.

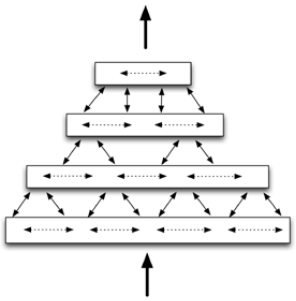

Figura 2-6 Red HTM con 4 regiones (fuente: [37]).

El beneficio de usar jerarquía es que se reduce el tiempo de entrenamiento y el uso de memoria, debido a que los patrones aprendidos en regiones bajas se transmiten y reutilizan en regiones más altas en la jerarquía combinándolos con los propios de ese nivel.

Las regiones están formadas por células interconectadas organizadas en columnas (véase Figura 2-7). En cada región la información se representa mediante un pequeño número de células activas, lo que los autores denominan representación distribuida dispersa (*Sparse Distributed Representation*, SDR) (véase Figura 2-7). Cada célula activa porta cierta información pero debe ser interpretada con el conjunto de las células activas para obtener un significado completo. Lo primero que hace una región HTM al recibir la información de entrada es transformarla en una SDR donde solo un pequeña parte de las células de entrada están activas.

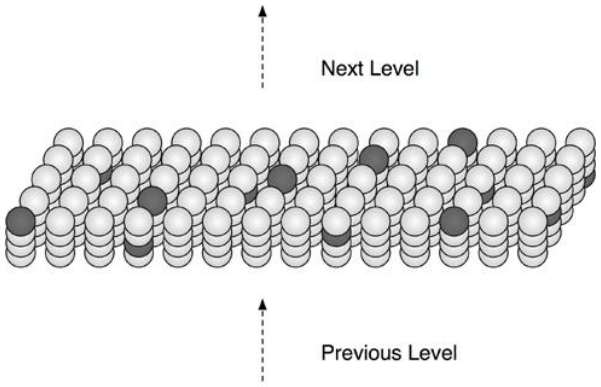

Figura 2-7 Una región HTM que contiene información distribuida dispersamente (fuente: [37]).

Computacionalmente una SDR es una lista de bits en la cual existen muy pocos 1 (células activas). Por ejemplo [39]:

0100000000000000000000010000000000000000000000000000000000000010000.……..01000

Donde solo estarán activos los bits [1, 23, 75,..., 2045].

Una región HTM aprende encontrando patrones. La región no sabe qué es cada input, actúa de manera estadística, buscando combinaciones en los datos de entrada que ocurran recurrentemente: a esto se le llaman patrones espaciales. Posteriormente, busca cómo estos patrones aparecen secuencialmente a lo largo del tiempo, lo que se llama patrones temporales o secuencias.

Cuando HTM ha aprendido los patrones puede realizar inferencias en los nuevos datos de entrada. Cada región HTM almacena las secuencias de patrones y al relacionarlas con los datos de entrada actuales, una región forma una o más predicciones (dependiendo de las secuencias de patrones aprendidas) de los próximos inputs que espera recibir. Además de esto, una misma entrada puede obtener predicciones diferentes, dependiendo de los datos de entrada recibidos con anterioridad. A continuación, la región compara las predicciones realizadas con el siguiente dato de entrada, en caso de que ninguna de ellas coincida con el dato, HTM sabrá que ha ocurrido una anomalía.

2.2.2 *NuPIC (Numenta Platform for Intelligent Computing)*

NuPIC es la plataforma de código abierto creada por Numenta que implementa los algoritmos HTM. Pero además es una comunidad donde aparte del código de NuPIC propiamente dicho se pueden encontrar aplicaciones de NuPIC a diversos casos o conjuntos de datos, tutoriales y videos explicativos de cómo realizar aplicaciones y foros de debate [36].

La implementación de NuPIC está escrita en lenguaje C++ y Python. NuPIC consta de diversos componentes [40]:

-Componentes sensoriales: es decir, los datos de entrada. Generalmente NuPIC recibe estos datos en forma de archivos CSV (del inglés *Comma Separated Values*, valores separados por comas).

A partir de aquí encontraríamos propiamente la implementación de los algoritmos HTM, denominados también a veces CLA (*Cortical Learning Algorithm*, algoritmo de aprendizaje cortical).

-Codificadores o *encoders* [41]: son los encargados de transformar los diferentes datos que podemos dar a la aplicación en SDRs. En función del tipo de datos y de lo que queramos que haga con ellos deberemos utilizar un codificador u otro.

-Jerarquía de regiones: aquí encontramos la "red de neuronas". Como ya se explicó anteriormente, una red está formada por diversas regiones en jerarquía y cada región está formada por células organizadas en columnas y capas. Se identifican los patrones espaciales mediante el *spatial pooler* (SP) y los patrones temporales con el *temporal pooler* (TP). Tras identificar los patrones, entrega como salida las predicciones para cada dato concreto en forma de SDRs.

-Clasificador CLA o *CLA classifier* [42]: intenta aprender una función de la SDR para cada tiempo *t*, produciendo una distribución de probabilidad para el campo predicho, *k* pasos hacia el futuro. El CLA toma los siguientes parámetros:

- *Alpha*: valor utilizado para calcular la media móvil
- *K*: número de pasos hacia el futuro que se quiere realizar la predicción.

Básicamente toma las SDRs de salida y las interpreta obteniendo como producto las predicciones que buscábamos.

Por último se calcula el grado de anomalía del dato que se está analizando. Este grado se mide con una valor de 1 a 0, siendo 1 totalmente anómalo y 0 no anómalo. El grado de anomalía se calcula como el porcentaje de columnas activas en el SP que fueron predichas incorrectamente por el TP:

$$anomalyScore = \frac{|A_t - (P_{t-1} \cap A_t)|}{|A_t|}$$

$P_{t-1} = Columnas\ predichas\ en\ el\ momento\ t$

$A_t = Columnas\ activadas\ en\ el\ momento\ t$

2.2.3 *Geospatial Coordinate Encoder*

Dentro de los diferentes codificadores el más útil para los datos que vamos a tratar es el denominado codificador geoespacial. A continuación se explicará brevemente el codificador geoespacial que se utiliza para convertir posiciones GPS en representaciones distribuidas dispersas (SDRs). Este codificador tiene las siguientes propiedades [43] y [44]:

-Las posiciones relativamente juntas tienen bits comunes al codificarse.

-La resolución de la codificación de los datos de entrada es proporcional a la velocidad de movimiento. Es decir, a menor velocidad mayor resolución y también a la inversa, a mayor velocidad menor resolución.

Vamos a proceder a explicar cómo se realiza la codificación:

Primero el codificador debe inicializar el espacio en el que vamos a trabajar, de manera que virtualmente divide este espacio en cuadrados a los que vamos a llamar celdas (cuadrados pequeños, véase Figura 2-8), la extensión que abarcan estos cuadrados dependerá de la escala, que es uno de los parámetros que se le da como entrada al codificador geoespacial. Posteriormente asigna a cada una de estas celdas dos valores "aleatorios" (son aleatorios en el sentido que cada valor no tiene ninguna relación con la posición espacial que ocupa la celda) en función de la latitud y longitud del mismo mediante una función *hash*[2]. El primer valor corresponde al valor que tendrá la celda y varía entre 0 y

[2] Función *hash*: algoritmo matemático que representa cualquier bloque arbitrario de datos de manera unívoca [76].

1, al cual llaman "orden". El segundo valor es la posición del bit, por lo tanto dependerá del número total de bits en la SDR. Cada celda concreta siempre tendrá los mismo dos valores (orden y bit) [45].

Se procede a explicar cómo se realiza la codificación concreta del ejemplo en la Figura 2-8. La estrella azul representa la posición a codificar. Se crea una cuadrícula (cuadrado grande verde) centrada en la posición a codificar y de tamaño dependiente de la velocidad de movimiento.

Después se seleccionan los *k* valores más altos (cinco en el ejemplo, la cantidad de valores dependerá del número de bits activos de la SDR), y se coloca un 1 en los lugares correspondientes de la matriz de bits, obteniendo el vector de la posición.

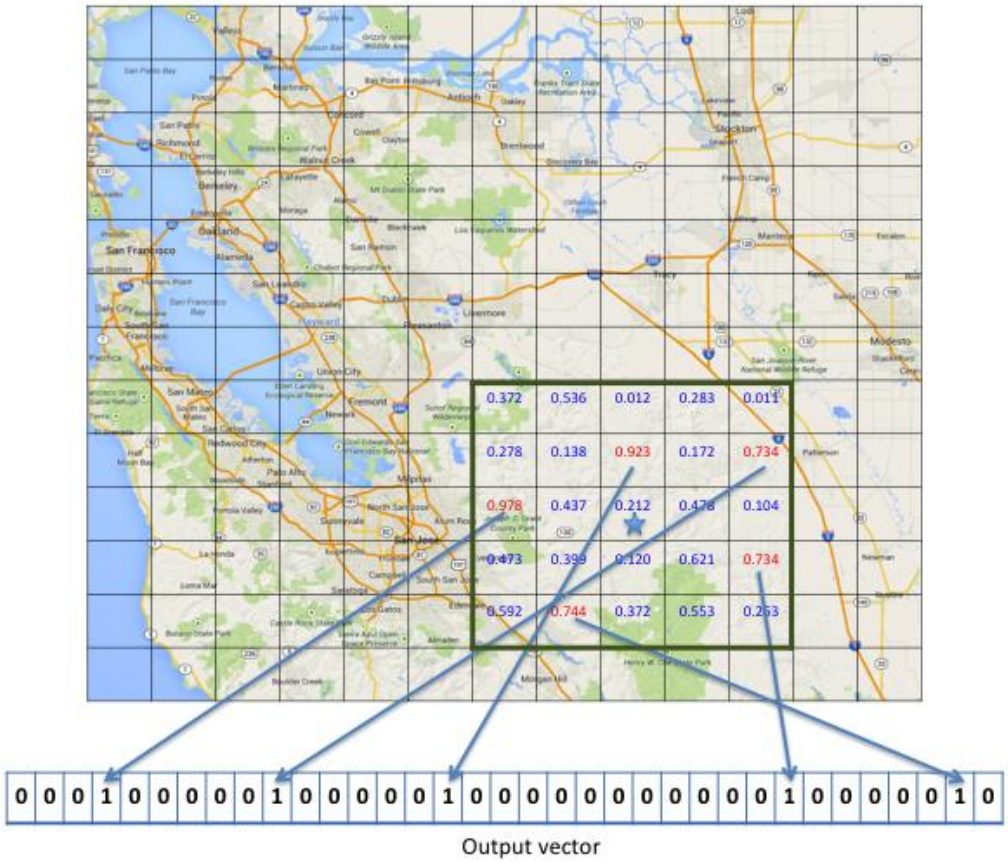

Figura 2-8 Generación de una SDR a partir de una entrada de latitud, longitud y velocidad por el codificador de coordenadas geoespaciales (fuente: [43]).

2.2.4 *Swarming*

La palabra *swarming* (procedente del inglés, en español "enjambre") tiene su origen en el comportamiento que algunos seres vivos desarrollan cuando se unen en enjambres, bandadas, etc. (véase Figura 2-9). Estos seres vivos se mueven o desarrollan su actividad con un objetivo común, normalmente relacionado con el lugar a donde se dirigen o la función que están desarrollando.

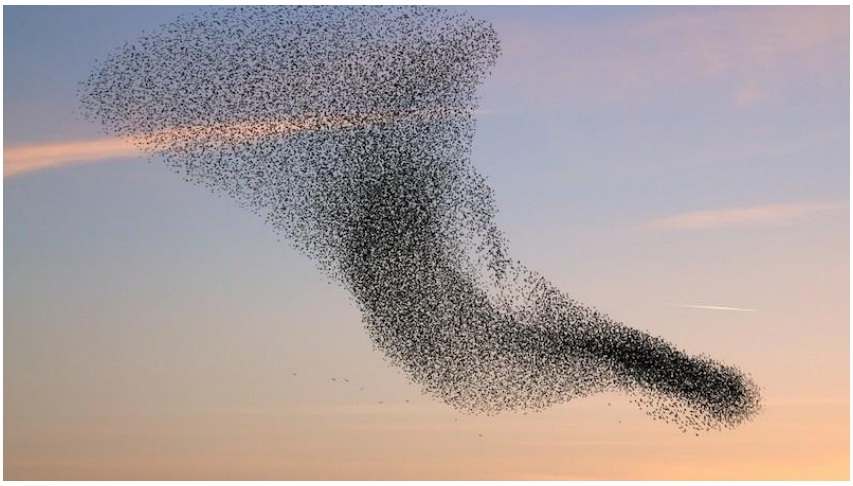

Figura 2-9 Bandada de pájaros (fuente: [46]).

Este concepto se ha desarrollado y actualmente se hace uso de él para referirse a un concepto de estrategia o de técnica empleada en diversas disciplinas, entre ellas en el ámbito militar.

En el caso concreto que nos atañe, NuPIC y los algoritmos HTM necesitan unos parámetros para poder crear la red neuronal y obtener resultados óptimos. Dependiendo del problema que queramos analizar y de cómo sean los datos que tenemos para ello, este ajuste variará. En NuPIC han creado una técnica llamada *swarming* que ajusta todos los parámetros de manera "automática". La misma se basa en crear y probar distintos tipos de modelos sobre los datos de entrada y compara los resultados obtenidos para así conseguir los parámetros óptimos para ese conjunto de datos.

2.3 Sistemas para la obtención de datos.

A continuación se explicarán los dos sistemas empleados para obtener y analizar los patrones de comportamiento. Primero en relación con los buques y posteriormente con las aeronaves.

2.3.1 *AIS (Automatic Identification System)*

Este sistema está diseñado para proveer de información sobre el buque propio a otros barcos o a estaciones en tierra [47]. Su uso es obligatorio desde el 31 de diciembre de 2004 para todos los buques sometidos al convenio SOLAS (*Safety Of Life At Sea*), que es un tratado internacional para la seguridad en el mar.

Mediante este sistema el buque proporciona datos muy variados, empezando por su posición GPS, rumbo, velocidad y número de identificación (MMSI) y también información más compleja como la función del buque, mercancía que transporta, puerto de salida y de destino entre otras.

Los sistemas AIS transmiten en la banda marítima de VHF y utilizan accesos del tipo *Self-Organizing Time Division Multiple Access* (SOTDMA) o *Carrier-Sense Time Division Multiple Access* (CSTDMA) [48].

2.3.2 *ADS-B (Automatic Dependet Survillance-Broadcast)*

El ADS-B, llamado ADS radiodifundido es un sistema de vigilancia mediante el cual se transmiten por radiodifusión datos de las aeronaves a intervalos de tiempo específicos. Este nuevo sistema de vigilancia aérea está sustituyendo a los antiguos modos de vigilancia y mejorando el posicionamiento de las aeronaves.

Actualmente solo está implantado en algunos países como Japón y Australia, pero está previsto que la Unión Europea lo haga durante este año también. A pesar de esto, la práctica totalidad de los aviones comerciales de pasajeros tienen este sistema, ya que además de enviar información, permite recibir los datos de los demás aviones al alcance [49].

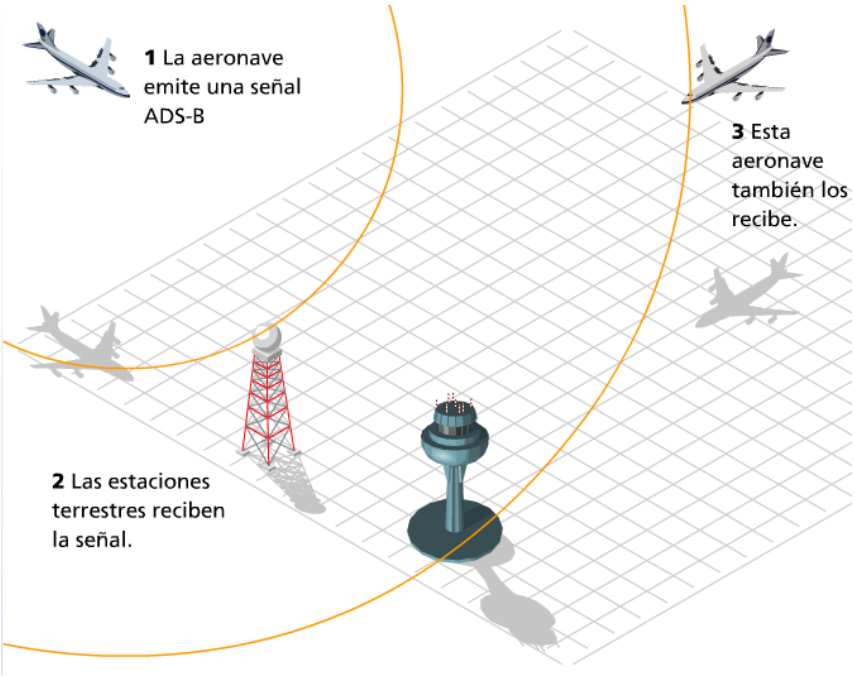

Figura 2-10 Sistema ADS-B (fuente: [50]).

Mediante este sistema la aeronave emite cada ciertos intervalos información de interés sobre el avión como la posición GPS, la altitud, la velocidad, el rumbo y diferentes códigos de identificación como el número ICAO (*International Civil Aviation Organization*), que es único de cada aeronave y el número de vuelo (como por ejemplo RYR6375, vuelo de Ryanair).

Este sistema cuenta con distintos modos para realizar el enlace de datos. El que se va a utilizar en este trabajo y el más extendido se denomina "Modo S extended Squitter". Como su nombre indica, es una extensión del transpondedor en Modo S que transmite en 1090 MHz.

2.4 Aplicaciones previas realizadas en un entorno marítimo

2.4.1 *Neptune*

Neptune es una IA creada por el mayor grupo de cruceros del mundo Carnival Corporation. El motor del sistema es Cortana, una inteligencia artificial desarrollada por Microsoft.

Desde el centro de control de la flota que se encuentra en Hamburgo se coordina el movimiento de más de un centenar de cruceros. La IA recibe aproximadamente dos millones de datos diarios de cada nave, que almacena y utiliza para aprender. Según se dice en [51], en caso de tormenta o cualquier imprevisto, Neptune calcula una nueva ruta óptima informando tanto al centro de control como al capitán del buque.

2.4.2 *Sistema inteligente para vigilancia marítima (Universidad Carlos III)*

Este proyecto, desarrollado por el Grupo de Inteligencia Artificial Aplicada (GIAA) de la Universidad Carlos III de Madrid, permite integrar información recibida a través del AIS y de radares para realizar una vigilancia marítima [52].

2.5 Aplicaciones y estudios realizados para vigilancia y control del tráfico aéreo

Al igual que sucede en el entorno marítimo, se han efectuado múltiples estudios y aproximaciones sobre el control y vigilancia de aeronaves para intentar introducir una inteligencia artificial al mismo, pero pocas son las aplicaciones que se han creado y se han puesto en práctica.

2.5.1 *Next Generation (NextGen)*

Next Generation es una de las propuestas más importantes relacionadas con el control del tráfico aéreo. Este sistema se comenzó a implantar en 2012 por la FAA (por sus siglas en inglés *Federal Aviation Administration*) del gobierno de los Estados Unidos y pretende substituir al actual, que es un sistema basado en radares, por uno basado en posicionamiento por GPS. Las comunicaciones por radio se reducirán para ser reemplazadas por una mayor cantidad de información digital y se pretende aplicar medidas de automatización. De esta manera, los controladores y los pilotos tienen más información y más precisa, obteniendo como resultados un aumento de la eficiencia de los vuelos y una reducción del consumo de combustible [53].

2.5.2 *Airport Surface Detection Equipment-Model X (ASDE-X)*

En relación con la implantación del programa NextGen, el sistema ASDE-X permite el control de las aeronaves que están en proceso de despegue, de aterrizaje o que, simplemente se están moviendo por el aeropuerto [54].

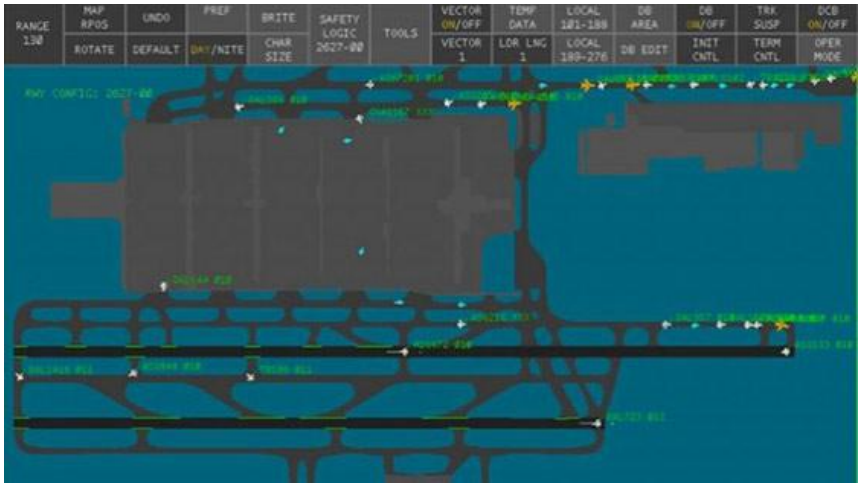

Figura 2-11 Sistema ASDE-X (fuente: [55]).

Este sistema avisa a los controladores de peligros potenciales en las rutas de las aeronaves mediante alarmas visuales en los aviones implicados, siendo de gran utilidad en condiciones de mal tiempo o baja visibilidad.

2.5.3 *ATAD (Air Traffic Anomaly Detector)*

ATAD utiliza información de los vuelos para detectar anomalías relacionadas con la posición, la velocidad y el tiempo [56]. La aplicación está construida sobre HTM-MoClu (Hierarchical Temporal Memory-Models Cluster). La base es similar a la utiliza en el HTM Engine pero MoClu es una plataforma para aplicaciones Java y además permite escalabilidad horizontal, pudiendo usar diversos servidores o dispositivos [57].

Se utilizan dos servidores. Uno de ellos sirve para conectar HTM-MoClu y obtener los datos de los vuelos. El otro se utiliza para entregar los datos al cliente.

En la interfaz de usuario, para la representación gráfica, se utiliza AngularJS una implementación de Google Maps llamada "angular-google-maps and D3 charts".

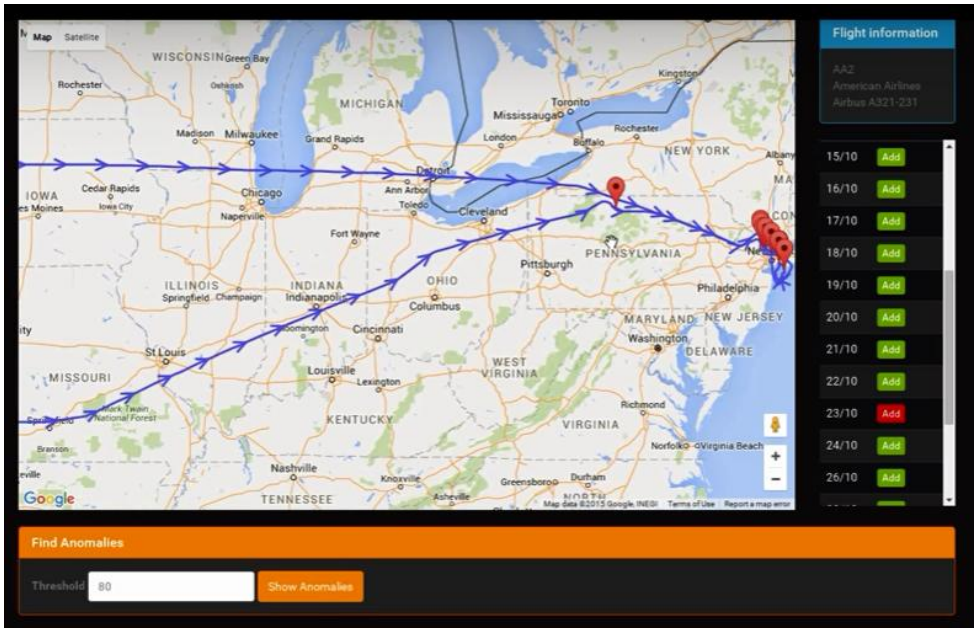

Figura 2-12 Interfaz gráfica ATAD (fuente: [56]).

3 DESARROLLO DEL TFG

3.1 Estudio y selección de los sistemas a utilizar

3.1.1 *Sistema de inteligencia artificial*

Anteriormente ya se enunciaron las características y puntos fuertes de HTM en 2.2.1 *HTM (Hierarchical Temporal Memory)*, a continuación se citan las más importantes subrayando su concordancia e idoneidad teniendo en cuenta los datos con los que vamos a trabajar.

-Transmisión continúa de datos en lugar de datos por lotes. Los datos que pretendemos analizar se transmiten de forma continua a medida que los buques y aviones van modificando su posición, rumbo, velocidad, etc. Además también pueden modificar sus patrones de comportamiento en el tiempo, es decir, no siempre se van a desplazar de la misma manera aunque realicen el mismo tránsito, por ejemplo.

- Datos con patrones basados en el tiempo. Es muy probable que los patrones de determinadas zonas dependan del tiempo. Por ejemplo, es posible que durante las horas matutinas naveguen más buques con dirección norte por el DST de Finisterre y durante las vespertinas lo hagan más en dirección sur, que la velocidad sea distinta durante el día que la noche o que varíe la altura a la que vuelan los aviones que pasan por el espacio aéreo de Galicia.

-Datos para los cuales técnicas simples como umbrales producen falsos positivos y falsos negativos. Es la base de este sistema de IA encontrar cuál es la relación entre las diferentes variables que recibe para así poder realizar una predicción de cómo espera que sea el siguiente conjunto de datos.

3.1.2 *Sistema operativo*

Tras la elección del sistema de inteligencia artificial se plantea la problemática de en qué medio realizar el trabajo. La tecnología que se va a usar esta ampliamente desarrollada en sistemas basados en UNIX (como Linux o MacOS). No así para Windows, pues aunque sí funciona en el mismo, la

cantidad de material (información, tutoriales, etc.) disponible para consulta y el número de aplicaciones desarrolladas es mucho menor.

Finalmente, debido a esta razón, se decide utilizar Ubuntu, una distribución de GNU/Linux que además es software libre. En un principio se pretende usar Ubuntu mediante una máquina virtual instalada en Windows, pero el uso de una máquina virtual reduce la capacidad del ordenador, y además se necesita una velocidad de procesamiento relativamente alta para poder utilizar de manera eficiente el programa. Debido a esto, se decide instalar Ubuntu, en concreto la última versión disponible (la 16.0) en otro disco duro para hacerlo funcionar conjuntamente con Windows.

3.2 Instalación del sistema de inteligencia artificial

3.2.1 *NuPIC y geospatial*

A continuación se procede a instalar NuPIC tal como explica en su video Matthew Taylor, *Open Source Community Flag-Bearer,* [58] y con la ayuda de la explicación que se realiza en la página web de NuPIC [59]. Primero se inicia la instalación de todas sus dependencias, en concreto:

- Compilador C++ 11 (Como gcc 4.8+ o clang).
- Python 2.7 & development headers
- *pip* versión mayor o igual a la 8.1.2
- *setuptools*
- *wheel*
- *numpy*

Lo primero que deberemos hacer es actualizar todos los repositorios de Ubuntu e instalar todas las actualizaciones. Para llevarlo a cabo abrimos el terminal y ejecutamos:

```
sudo apt-get update
sudo apt-get upgrade
```

Tras esto, la primera dependencia es un compilador de C++ 11. Ubuntu 16.0 ya incluye uno, gcc. Y lo más común es que también se cuente con Python 2.7, en todo caso se comprueba la versión instalada, e instalamos los *development headers* introduciendo en el terminal:

```
python --version
sudo apt-get install python-dev
```

El siguiente paso es la instalación *pip*. La forma más segura de hacerlo es a través del siguiente comando:

```
sudo apt-get install python-pip
```

Pero, como señalan los creadores de NuPIC, de esta forma se obtiene una versión muy antigua con problemas de actualización. Por ello, se instala la versión más actualizada de esta otra forma:

```
curl https://bootstrap.pypa.io/get-pip.py | sudo python
```

De esta última forma, se instalan también otras dos dependencias (*wheel* y *setuptools*), independientemente del modo de instalación, es necesario tener la versión 8.1.2 o mayor de *pip* para comprobar la versión que se tiene ejecutamos:

```
pip --version
```

La última dependencia, *numpy*, se instalará al instalar NuPIC.

Primero hay que instalar los NuPIC *bindings*, lo cual realizaremos mediante *pip*. Para ello introducimos el siguiente comando, con el cual también se instalará *numpy*:

```
Pip install https://s3-us-west-2.amazonaws.com/artifacts.numenta.org/numenta/nupic.core/releases/nupic.bindings/nupic.bindings-0.4.0-cp27-none-linux_x86_64.whl --user
```

Al finalizar comprobamos la versión de NuPIC *bindings* para verificar que todo ha ido correctamente con:

```
pip freeze | grep nupic
```

y procedemos a instalar NuPIC ejecutando:

```
pip install nupic --user
```

Una vez instalado NuPIC, para comprobar que todo se ha hecho correctamente se ejecutarán unos tests. Instalamos git y clonamos el directorio de *nupic*:

```
sudo apt-get install git
git clone https://github.com/numenta/nupic.git
```

Ahora entramos en el directorio de *nupic*, fijamos las variables de entorno y ejecutamos los test. Hay diversos test en función de la característica que se quiere probar de la aplicación. En este caso ejecutamos los *unit tests* para probar únicamente si la aplicación se ha instalado y funciona correctamente ejecutando:

```
cd nupic
export NUPIC=$HOME/nupic
export PATH=~/.local/bin:$PATH
./scripts/run_nupic_tests.py –u
```

Ahora vamos a proceder a instalar la aplicación *geospatial* desarrollada por NuPIC que implementa el codificador geoespacial. Copiamos el directorio de github donde se encuentra la aplicación e instalamos los requerimientos [60] :

```
git clone https://github.com/numenta/nupic.geospatial.git
pip install -r requirements.txt
```

A continuación se prueba el funcionamiento de la aplicación y las distintas utilidades que tiene como se explica en [61].

3.3 Obtención de datos

3.3.1 *Datos marítimos*

La obtención de datos AIS se ha realizado a través de la API de MarineTraffic *for AIS Data*. Como se reseña en [13] este servicio se ha proporcionado de manera gratuita. Esto se debe a que este trabajo se encuentra englobado dentro de la *MarineTraffic Research Network*. MarineTraffic presta este servicio como apoyo a los estudios académicos relacionados con el sector marítimo.

Esta API proporciona dos tipos de datos: los de respuesta simple y los de respuesta extensa. Como se explica en [62], la respuesta simple nos permite obtener cada dos minutos los siguientes datos del buque: Número de Identificación del Servicio Móvil Marítimo (MMSI, *Maritime Mobile Service Identity*), latitud, longitud, velocidad, rumbo, estado y tiempo. Por su parte, la respuesta extensa se envía cada hora con los siguientes datos, además de los enviados en el mensaje simple: nombre del buque, tipo de buque, número IMO, *callsign*, bandera, puerto actual, último puerto, momento de salida del último puerto, destino, tiempo estimado para llegar a su destino, eslora, manga, calado, arqueo de registro bruto, tonelaje de peso muerto y año de construcción.

La obtención de datos se ha realizado mediante la API mencionada como se explica en [13] con el siguiente comando:

```
wget
http://services.marinetraffic.com/api/exportvessels/106833382f96e788ba3e6d9c1b1de719a31fa215/timestamp:15/protocol:csv -O
```

De esta forma se obtienen archivos en formato CSV[3] con los datos del mensaje AIS sencillo. Posteriormente, es necesario ordenar los datos primero por el barco en concreto (por MMSI) y después cronológicamente para dejarlos listos para procesarlos con NuPIC.

[3] Archivos CSV: archivos de texto planos donde cada dato se encuentra separado por una coma.

3.3.2 *Datos aéreos*

Los datos de aviones se han obtenido gracias al ya mencionado sistema ADS-B. Para su obtención se estudiaron diferentes posibilidades.

La primera de ellas podría ser a través de un servidor similar a MarineTraffic. Este servidor recopila la información de diversos sistemas entre los cuales está el ADS-B y la proporciona de manera visual en su página web [63].

Otra de las formas de obtención de datos es directamente de los propios aviones. Como ya se explicó en el apartado 2.3.2 ADS-B (*Automatic Dependet Survillance-Broadcast)*, este sistema realiza la transmisión de datos en la banda de UHF, en concreto en 1090Mhz. Por lo tanto, para recibir la señal se debe estar en línea visual con el transmisor. En el caso de los buques esto queda descartado completamente ya que difícilmente se podría abarcar una zona amplia y así conseguir suficientes datos para su análisis. Pero los aviones comerciales comúnmente vuelan a unas alturas que oscilan entre los 20.000 y los 40.000 pies, lo que nos permite tener visual de un área relativamente grande, llegando a recibir transmisiones de aviones incluso a 200 km de distancia.

Finalmente se decide optar por esta vía principalmente por dos razones. La primera porque es una manera totalmente autónoma de recepción de datos, no se depende de internet para conseguirlos. La segunda debido a que la recepción de los datos se podría asemejar a la que recibiría un radar de un buque. De este modo, se puede comprobar la posibilidad de aplicar sistemas de inteligencia artificial a buques de vigilancia marítima o a estaciones en tierra.

Dentro de las formas de recibir estos datos se encuentra la adquisición de un receptor específicamente diseñando para recibirlos. En el mercado encontramos diversos productos como el receptor Kinetic Avionics SBS-1 o el AirNav-systems. Estos receptores, cuyo precio oscila alrededor de los 700€, proporcionan un sistema muy estable y fiable, además son las herramientas utilizadas en estudios y trabajos de aeronáutica.

Otra de las posibilidades es el uso de un receptor SDR (*Software Defined Radio*, radio definida por software). Su uso está muy extendido como receptor de TDT (Televisión Digital Terrestre) y DAB (*Digital Audio Broadcasting*, en castellano transmisión digital de audio), pero las especificaciones de los mismos tras unas pequeñas modificaciones en el *software* de serie abarcan un campo mucho más amplio. Además, este dispositivo es barato y fácil de adquirir.

Finalmente se decide utilizar este tipo de receptor para la adquisición de datos. En concreto se ha utilizado el receptor RTL2832U (véase Figura 3-1). Su valor son unos 8 € y, como se ha mencionado, está pensado para recibir TDT y DAB. Por eso se suministra con un CD-ROM con drivers y programas para ese uso, con un mando a distancia por infrarrojos y con una antena de unos once centímetros para recibir en la banda de la TDT (centrada aproximadamente en 650Mhz).

Figura 3-1 USB Software Radio RTL2832U (Fuente: [64]).

La antena genérica con la que se recibe el receptor no está pensada para recibir en la frecuencia del ADS-B. Debido a esto, con la frecuencia de recepción del ADS-B en 1090Mhz. se fabrica una antena específica para la misma de aproximadamente 69 mm. de longitud y se reduce la longitud del cable coaxial para reducir pérdidas (para más detalles véase el Anexo II) y aumentar las prestaciones de recepción con el objetivo de recibir aeronaves a una mayor distancia.

Figura 3-2 Receptor SDR para 1090Mhz (fuente: propia).

A continuación se procede a realizar la instalación del software necesario para recibir y decodificar la información que se recibe del ADS-B. Para ello se ha seguido el tutorial en [65]. Primero se deben instalar algunas dependencias y permitir el acceso al dispositivo:

```
sudo apt-get install git
sudo apt-get install cmake
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install build-essential
```

Se continúa descargando los drivers y controladores del receptor, compilando e instalando los mismos:

```
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

Ahora hay que permitir al sistema el acceso al dispositivo y posteriormente hay que reiniciar:

```
cd
Sudo cp ./rtl-sdr/rtl-sdr.rules /etc/udev/rules.d/
Sudo reboot
```

Después de reiniciar, comprobamos que todo ha ido correctamente introduciendo en el terminal:

```
rtl_test -t
```

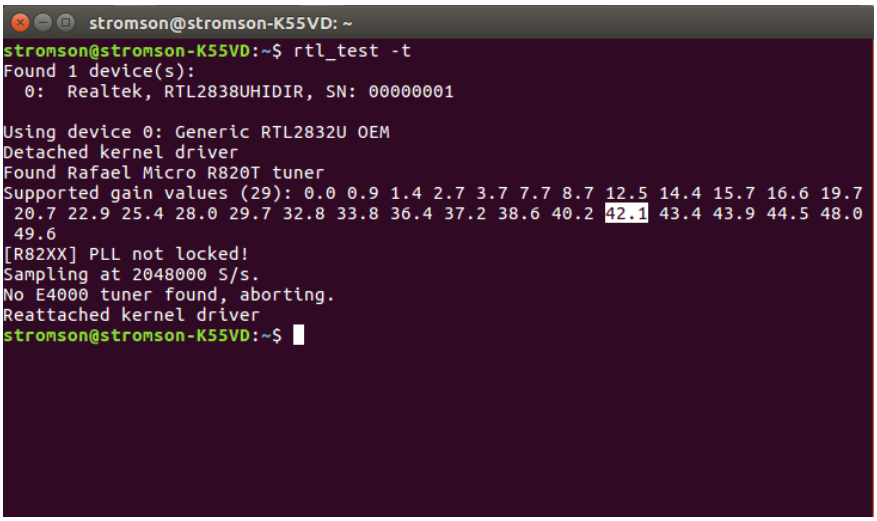

Figura 3-3 Prueba de funcionamiento receptor SDR (fuente: propia).

Posteriormente se procede a instalar DUMP1090 el programa que decodifica las señales ADS-B y nos permite obtener los datos en claro. Abrimos el terminal y ejecutamos:

```
git clone git://github.com/MalcolmRobb/dump1090.git
cd dump1090
make
```

Así se ha descargado e instalado el programa. Ahora lo ejecutamos mediante el comando:

```
./dump1090 --interactive --net
```

Este programa tiene multitud de opciones tanto para configurar el receptor SDR, modificando la ganancia entre otras, como para configurar la salida de la aplicación, ya que su uso está pensado para nutrir de datos a distintas páginas web que presentan la información de los vuelos.

Las opciones que se han usado sirven para mostrar por el terminal los datos (*--interactive)* y para enviarlos por internet*(--net*) a través del puerto IP designado o del predeterminado (127.0.0.0:8080). Tras ejecutar el comando deberemos ver por el terminal los datos decodificados que estamos recibiendo (véase Figura 3-4). Además el programa tiene una interfaz gráfica para visualizar los aviones que se están recibiendo en cartografía de GoogleMaps. Para ello introducimos en el navegador web la siguiente dirección 127.0.0.0:8080 (véase Figura 3-5). Finalmente se realizan diversas pruebas comprobando con qué parámetros se obtiene mayor alcance.

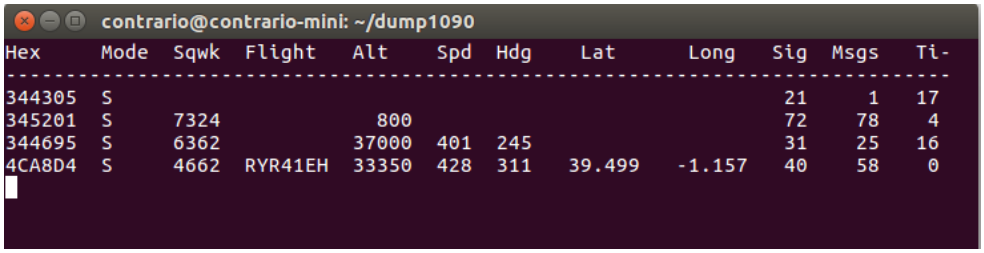

Figura 3-4 Recepción datos ADS-B en el terminal de Ubuntu.

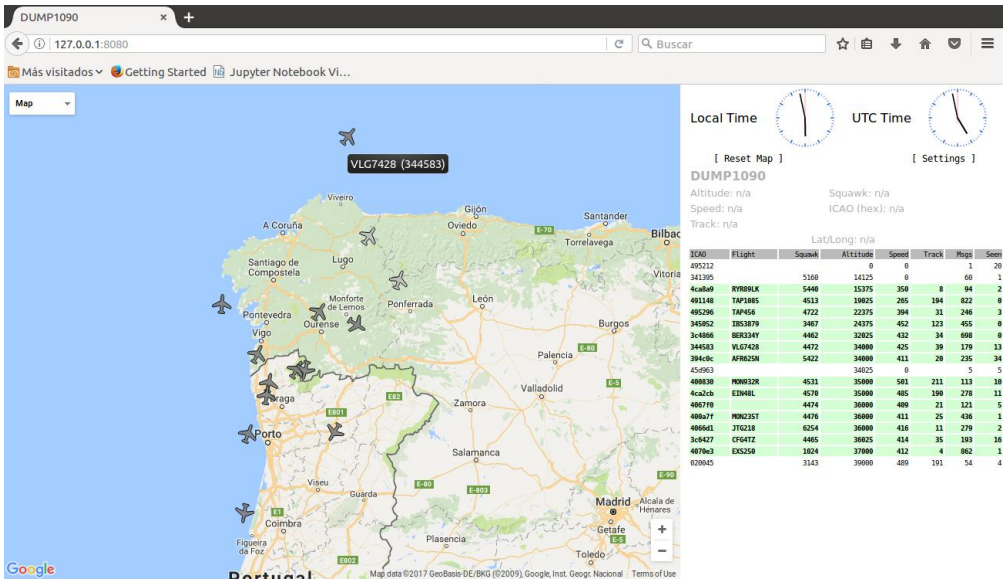

Figura 3-5 Recepción de datos ADS-B en 127.0.0.0:8080.

Ahora se necesita capturar estos datos en un fichero de texto y más concretamente en un fichero CSV y añadirles la fecha y hora a la que fueron capturados para procesarlos por NuPIC.

Para conseguirlo se ha creado un *script* en *bash* que lanza la aplicación Dump1090 para comenzar a capturar datos, y a intervalos regulares de tiempo realiza diferentes operaciones con los datos hasta dejarlos en el formato que se necesita. Los datos se reciben de la aplicación Dump1090 como figura en el Anexo IX.

Se efectúan las siguientes operaciones: adición del tiempo en el momento de captura del dato, eliminación de las cabeceras, adición de comas entre cada uno de los diferentes datos, eliminación de los espacios en blanco y de las filas parcialmente incompletas. Por último se copian estos datos a un fichero de texto o CSV.

```
cd /home/stromson/dump1090

./dump1090 --interactive --net --gain 43.4 --enable-agc |gawk '{print $0 , strftime("%Y-%m-%dT%H:%M:%S"); fflush() }'| sed '/--/d' | grep -v "Mode" | sed 's/./&,/8;s/./&,/13;s/./&,/22;s/./&,/32;s/./&,/40;s/./&,/46;s/./&,/52;s/./&,/62;s/./&,/72;s/./&,/80;s/./&,/87;s/./&,/90' | sed 's/ //g'|sed '/,,,/d' >> /home/stromson/dump1090/DATOS_AVIONES/salida.txt
```

Tras esto, como ocurría con los datos de la API de MarineTraffic, los ordenamos para dejarlos listos para procesar por NuPIC. Se ordenan los datos por número ICAO (International Civil Oraganization)

```
awk -F, '{print $1","$13","$3","$4","$5","$6","$7","$8","$9","$10","$11","$12","$2}' /home/stromson/dump1090/DATOS_AVIONES/salida.txt | sort -n | awk -F, '{print $1","$13","$3","$4","$5","$6","$7","$8","$9","$10","$11","$12","$2}'| sed '/,,,/d' > /home/stromson/lanzar/ordenados.csv
```

3.4 Ejecución del programa

A continuación se explica el funcionamiento de las dos aplicaciones, señalando los archivos que forman parte de las mismas y destacando las funciones más relevantes. La aplicación se ha desarrollado en Python, así como lo está parte de NuPIC. La explicación se realiza de la aplicación marítima ya que ambas funcionan de manera similar.

La primera parte del proceso es la obtención de los datos en formato CSV compatible con la aplicación, como se explica en el apartado anterior. Después se hace la llamada al programa indicándose las diferentes opciones con las que queremos que se ejecuten los datos (véase Tabla 3-1).

En el caso de los buques hacemos la llamada a la aplicación con:

```
./maritme.py <ruta/al/archivo/de/entrada> [opciones]
```

Y para los aviones:

```
./air.py <ruta/al/archivo/de/entrada> [opciones]
```

Opción	Comando
Help	-h, --help
Manual sequence	-m, --manual-sequence
Time encoders	-t, --time-encoders
Verbose	-v, --verbose
Output-dir	-o outputdir, --output-dir=*outputdir*
Scale	-s *scale*, --scale=*scale*

Tabla 3-1 Opciones de comandos para la aplicación (fuente: propia).

Este comando llama al módulo *maritime.py* (para visualizar el código de este módulo y de los siguientes véase del Anexo III al VIII), que contiene la función *runMaritime*, y le pasa los argumentos de entrada que hayamos indicado. Esta función coordina todo el proceso de análisis, es decir transporta la información desde donde se genera a donde se necesita.

Coge el fichero de entrada indicado y se lo envía al siguiente módulo denominado *preprocess_data.py*, que contiene una función homónima. Aquí se realiza el preprocesado de la información eliminándose aquellas posiciones repetidas por fecha y hora o por latitud y longitud y también aquellas muy próximas en el tiempo o en el espacio. Por último, genera un archivo CSV llamado preprocessed_data.csv y se lo pasa a *runMaritime*.

El siguiente archivo que recibe los datos es *geospatial_anomaly.py*. Este módulo es el centro de la aplicación y el encargado de obtener las anomalías utilizando las diferentes herramientas de que consta NuPIC. En primer lugar se definen los codificadores que se van a utilizar y los parámetros de los mismos. Estos los recibe a través del archivo *model_params.py* y también por los indicados en la llamada a la función (-s -t -m). Posteriormente crea el modelo en base a estos codificadores para analizar los datos. Toma el fichero *preprocessed_data.csv* y analiza cada fila de datos obteniendo para cada uno el valor de la anomalía y reescribe los datos en el fichero de salida denominado *anomaly_score.csv*.

Estos datos se van partiendo en secuencias; cuando realiza la partición de cada secuencia reinicia el modelo al estado inicial. Una secuencia es un conjunto de datos que la aplicación toma separadamente formando una ruta completa. La división en secuencias dependerá de las opciones de entrada. Si se indica la opción "-m" en la llamada al programa los datos se partirán cuando el "nombre" (en este caso MMSI) de uno de ellos sea distinto al siguiente o cuando entre dos de ellos exista un intervalo mayor de 34 minutos. Si no se indica nada se parten los datos solo cuando entre dos de ellos exista el mencionado intervalo.

La ruta de este fichero se pasa a la función *runMaritime*, que la entrega a los módulos *anomaly_to_kml.py* y a *postprocess_data.py*. El primero de ellos genera un archivo KML[4] que contiene la representación de los datos analizados en función de su anomalía, como se explica en el apartado 3.6 Representación gráfica. El segundo toma los datos y los prepara para posteriormente realizar el análisis de optimización. A partir del dato que le hallamos indicado, comienza a eliminar los falsos positivos que siempre entrega el módulo *geospatial_anomaly.py*, correspondientes al análisis de las primeras posiciones una vez se ha reseteado la secuencia. Además, marca los datos sobre los que posteriormente se hará el análisis para poder identificar con facilidad aquellos que son de interés. La salida de este módulo es el archivo *data_to_analysis.csv*.

[4] KML: Por sus siglas en inglés *Keyhole Markup Language*. Es un lenguaje basado en XML para representar datos geográficos en 3 dimensiones.

A continuación se presenta un diagrama del funcionamiento, con los diferentes módulos, archivos y funciones que intervienen en el proceso y se representa en él el flujo de información.

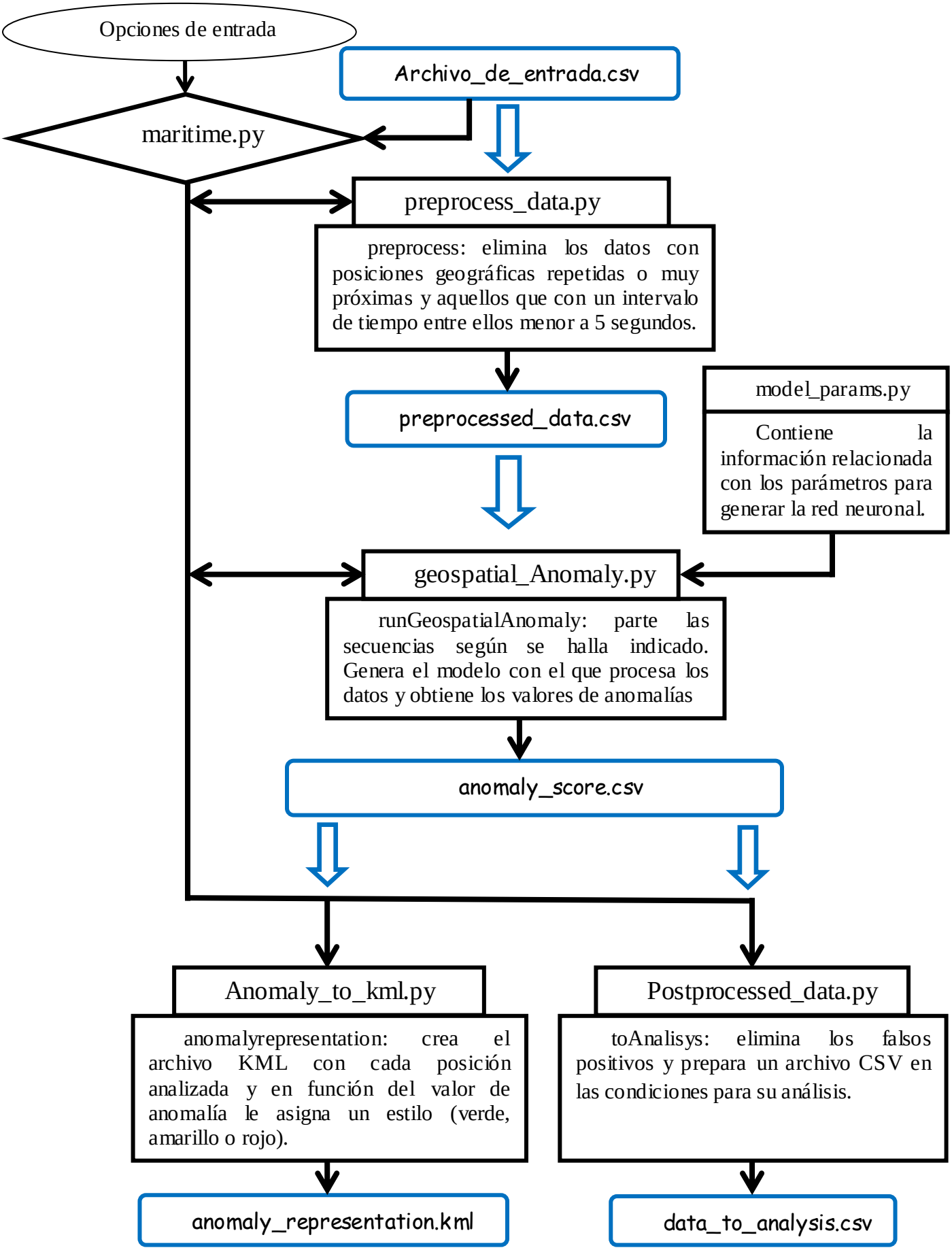

Figura 3-6 Diagrama de funcionamiento de la aplicación.

3.5 Optimización de la aplicación y sus parámetros

3.5.1 *Swarming*

Tras la instalación de NuPIC se procede a instalar el *swarming* para poder optimizar los diferentes parámetros con los que trabaja la aplicación de manera automática en función de los datos de entrada. Primero deberemos instalar MySQL, que es un sistema de gestión de bases de datos relacionales. Este sistema utiliza un lenguaje estructurado de consulta (SQL, por sus siglas en inglés *Structured Query Language*) [59]. Procedemos instalando MySQL con el siguiente comando:

```
sudo apt-get install mysql-server
```

Durante la instalación nos pedirá que pongamos una contraseña que deberemos recordar para posteriormente introducirla en la configuración. En el tutorial nos instan a dejarla vacía, pero debido a actualizaciones en MySQL si dejamos la contraseña vacía al finalizar no funcionará. Tras la instalación ejecutamos los test para ver que todo ha ido bien:

```
python $NUPIC/examples/swarm/test_db.py
```

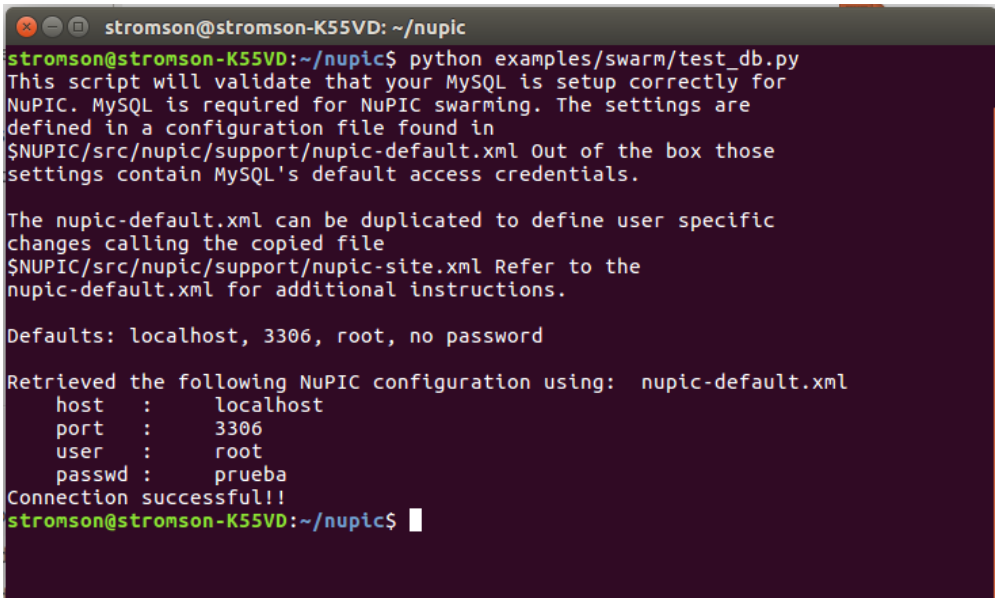

Figura 3-7 Test de *swarming* correcto (fuente: propia).

Si ha funcionado, tenemos todo correcto y ya podemos ejecutar *swarming*. Para ello probamos el ejemplo en [66]. Así comprobaremos cómo funciona mediante :

```
$NUPIC/scripts/run_swarm.py $NUPIC/examples/swarm/simple/search_def.json --maxWorkers=4
```

Veremos cómo va creando diferentes modelos a los cuales va variando los datos y posteriormente realiza el análisis con cada uno de ellos.

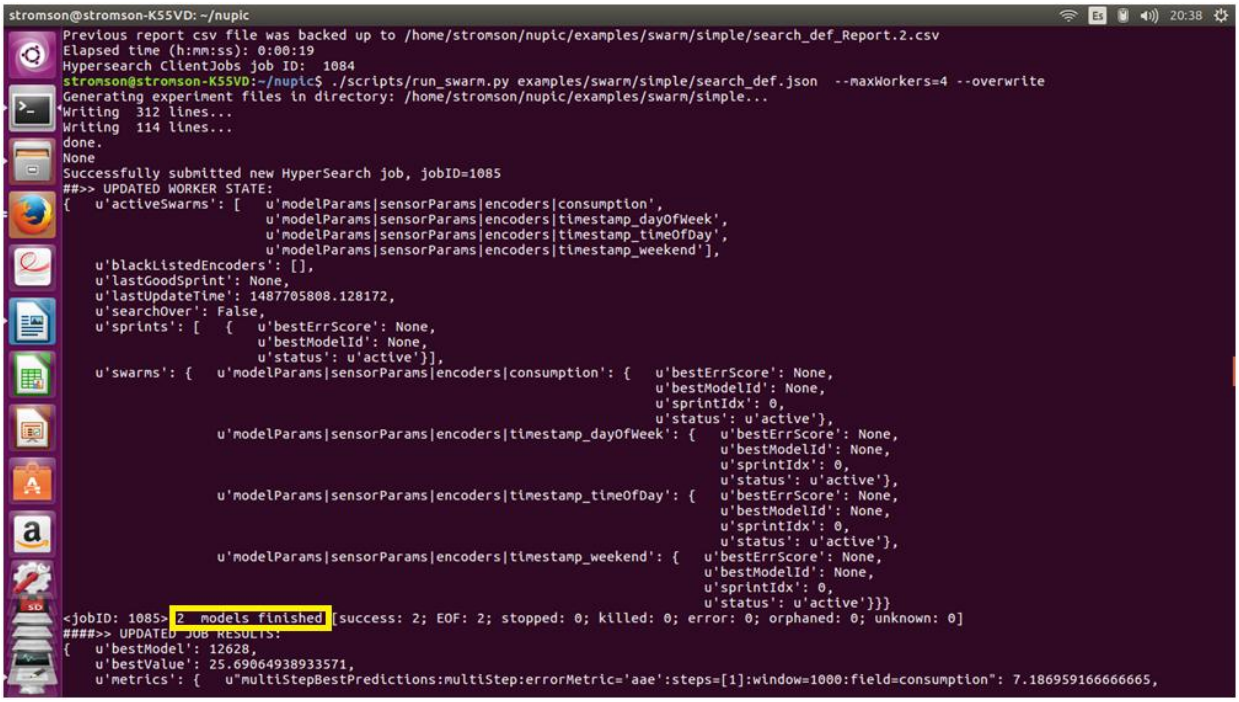

Figura 3-8 Swarming. Creación de los modelos.

Cuando ha finalizado el proceso, vemos el resultado con el total de modelos creados y el óptimo de ellos (véase Figura 3-9). Además, se crea un archivo con todos los parámetros de este último modelo que es el que deberemos utilizar al ejecutar posteriormente la aplicación de NuPIC.

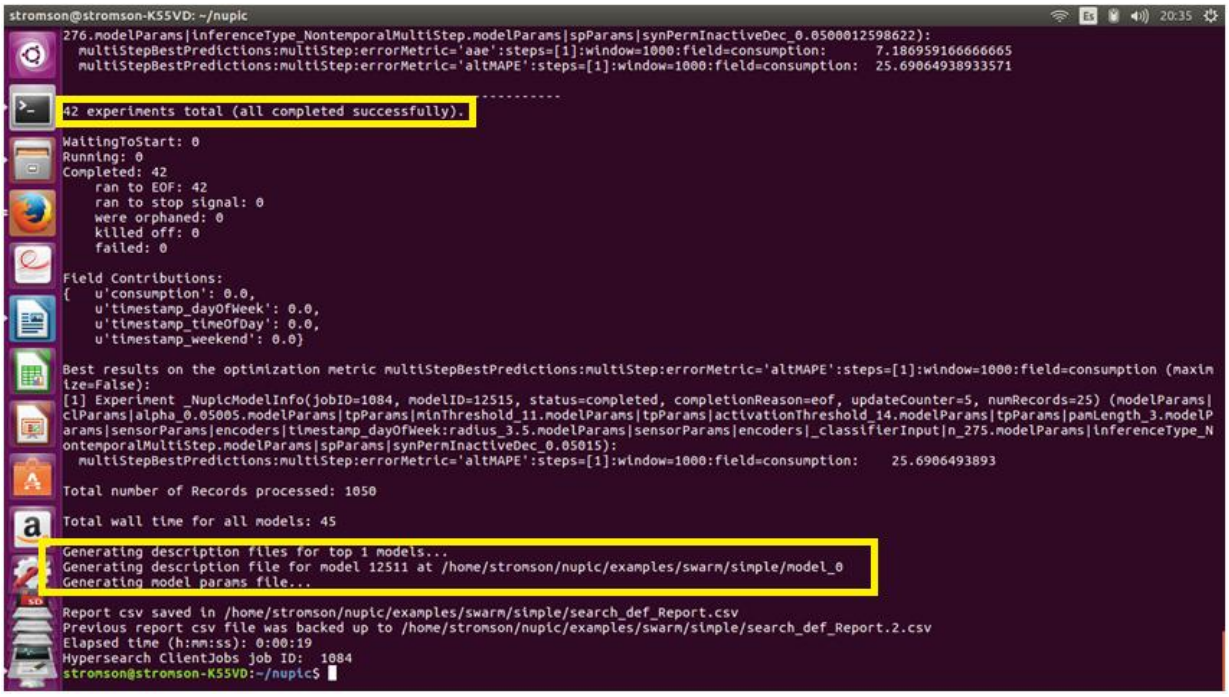

Figura 3-9 Swarming. Finalización de la prueba.

Para familiarizarse con el *swarming* se prueban diversos ejemplos de aplicaciones realizadas por el equipo de NuPIC. Como esta [67] donde se ajusta el modelo para analizar y predecir el consumo de energía de un gimnasio en Australia, entre otras, y se realizan diversos intentos infructuosos de aplicar esta técnica a los datos de estudio.

Posteriormente se observa que todas las aplicaciones desarrolladas en NuPIC que utilizan *swarming* analizan datos escalares y ninguna lo hace sobre posiciones o datos GPS. Por ello se procede a ponerse en contacto con el *Open Source Community Flag-Bearer*, Matthew Taylor. Muy amablemente nos contesta que no está implementado el *swarming* para datos espaciales, pero que no es imposible realizarlo si se tienen amplios conocimientos en Python y en HTM.

Ante esta situación, se decide proceder a estudiar el problema de la optimización desde otra perspectiva.

3.5.2 *Optimización utilizando F-Score*

Debido a la imposibilidad de aplicar swarming en este tipo de datos y a la necesidad de optimizar los diferentes parámetros del sistema es necesario aplicar otro método. Tras barajar distintas posibilidades se decide ejecutar manualmente varios modelos con diversos parámetros y evaluarlos empleando un método que tiene como resultado un único valor denominado F-Score. Este método se emplea para valorar programas de recuperación de documentos, sistemas de diagnóstico de enfermedades y aplicaciones de reconocimiento de patrones.

El F-Score es una de los métodos utilizados para evaluar sistemas de inteligencia artificial y en concreto aquellos de *machine learning* como es nuestro caso [68]. Para obtener el valor se clasifica los resultados obtenidos en 4 grupos como se puede ver en la Figura 3-10.

- Verdaderos positivos (a partir de ahora Vp): en nuestro caso son aquellos valores que se han obtenido como anómalos por la aplicación y que realmente lo son.

-Falsos positivos (a partir de ahora Fp): aquellos que la aplicación señala como anómalos pero que realmente no lo son.

-Verdaderos negativos (Vn): la aplicación los ha señalado como no anómalos y así lo son.

-Falsos negativos (Fn): se han señalado como no anómalos pero en realidad, sí son anómalos.

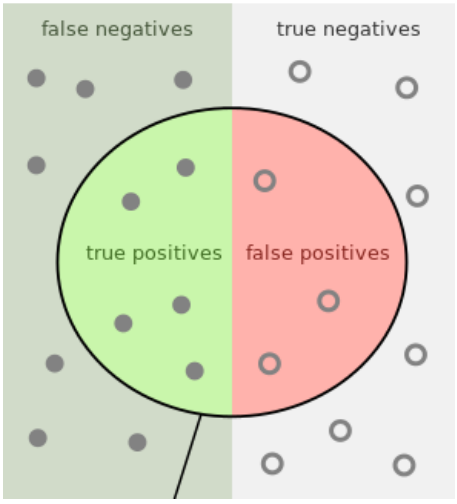

Figura 3-10 Sensibilidad y precisión (fuente: [69]).

Estos cuatro grupos se utilizan para obtener la sensibilidad y la precisión mediante las siguientes fórmulas:

$$\text{Precisión} = \frac{Vp}{Vp+Fp}$$

$$\text{Sensibilidad} = \frac{Vp}{Vp+Fn}$$

Una vez obtenidos estos valores se calcula el F-Score como:

$$\text{F-Score} = \frac{2 \times Precison \times Sensibilidad}{(Precision+Sensibilidad)}$$

De esta manera se obtiene un valor para poder comparar la bondad y lo certero que es un determinado modelo. El F-Score tiene como máximo el valor 1 que indicaría un modelo perfecto, es decir, que realiza todas las predicciones de manera acertada, y el 0 como valor mínimo.

3.5.3 *Creación de las anomalías*

Por la propia naturaleza de los datos, estos están obtenidos de fuentes reales y por lo tanto no es posible a priori diferenciar aquellos no anómalos o normales de aquellos que no lo son. Además, cuanto mayor sea el conjunto de datos analizados es de suponer que mayor será su normalidad, en tanto que no se producen alteraciones del tráfico marítimo o aéreo continuamente. Por ello se supone que todos los datos obtenidos a través del AIS y del ADS-B son normales y, por lo tanto, no anómalos.

Debido a no contar con anomalías reales, se generan diferentes anomalías artificialmente que se incluirán posteriormente en el conjunto de datos normales para su análisis. Dado que la zona que estamos estudiando es un DST, el número de comportamientos que pueden ser considerados como

anómalos aumenta, ya que esta zona es una vía de circulación con sentidos en cada vía. Por esto, no es normal que ningún buque se pare, se salga de ella o varíe el rumbo en exceso.

Para realizar el estudio se han filtrado los datos de todo el DST para utilizar únicamente los de la vía de circulación más al este del dispositivo, que por otra parte es la que tiene mayor densidad de tráfico.

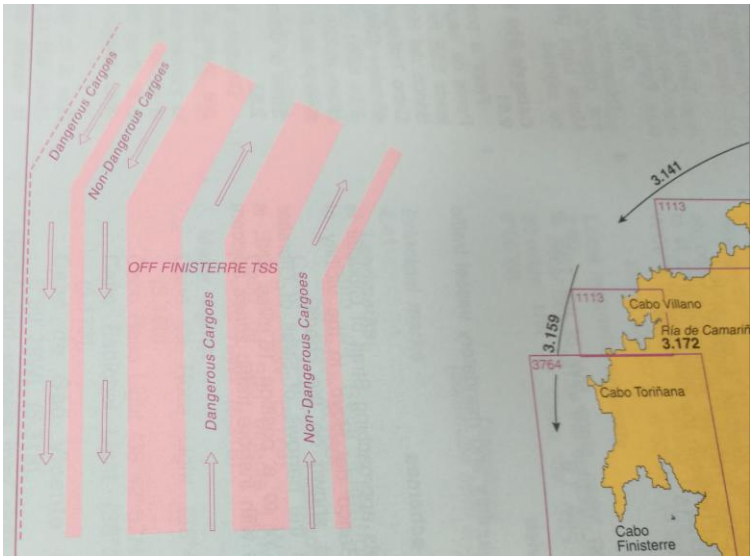

Figura 3-11 DST de Finisterre (fuente: [15]).

Los tipos de anomalías que se han generado incluyen comportamientos de buques con distintos grados de anomalía. Para generarlas, se ha utilizado un editor de rutas GPX y, posteriormente, se han convertido los archivos a formato CSV.

En total se han generado alrededor de 700 datos anómalos con comportamientos tales como encontrarse parado, navegar a una velocidad reducida en comparación con el tráfico de la zona, salir del DST, navegar en dirección contraria al conjunto de buques, hacer un cambio de rumbo no concordante con el del tráfico en esa zona, realizar cambios bruscos y aleatorios de rumbos, navegar a velocidad excesiva, etc. Estos datos se introducen y mezclan con los datos reales y son procesados por la aplicación, una vez ha sido entrenada.

Por otra parte se han ejecutado los módulos con diferentes características para evaluar los distintos resultados y seleccionar la configuración que optimiza el funcionamiento. Al procesar todos los datos (los reales y los creados) con el núcleo de la aplicación se obtiene el archivo *anomaly_scores.csv*. Después, se procesan con el módulo *postprocess_data.py* para eliminar los falsos positivos obtenidos como se explica en el apartado 3.4, y así poder analizar los datos de manera más precisa y conseguir unos resultados más reales, obteniendo finalmente el archivo *data_to_analysis.csv*. Estos datos son los que posteriormente se analizan para calcular el F-Score de cada uno de los modelos en las pruebas realizadas.

Tras familiarizarse con la aplicación y analizar el efecto de cada una de las funciones, características y parámetros que intervienen al generar las anomalías, se crean diferentes pruebas para evaluar el modelo. Para cada una de las pruebas se calcula el F-Score (con los datos aproximadamente a partir del once mil, una vez ya se ha entrenado a la aplicación) y, en algunos casos, la velocidad de convergencia. Para calcularla se utiliza el promedio de anomalías cada 400 muestras y se obtiene la gráfica de convergencia del modelo en cada uno de los supuestos.

Cabe recalcar que este es un proceso dinámico, por lo que conforme se obtienen resultados positivos o negativos se modifican los parámetros y funciones por los que se rige el funcionamiento del módulo que obtiene el valor de anomalía.

3.6 Representación gráfica

Con el objetivo de hacer más visuales los resultados entregados por la aplicación, se decide representar gráficamente los datos de buques y aviones con el programa Google Earth y archivos en formato KML. Se utiliza el módulo de Python, llamado *anomaly_to_kml.py* similar al propuesto en [13] y en [70]. En el caso de los aviones, al contener además la altura es necesario representarlos en las 3 dimensiones y también en función de si se encuentran sobre el mar o sobre tierra. Para ello se añaden las cabeceras necesarias y se modifica como se representan los puntos en el archivo KML [71].

Este módulo genera un archivo que hemos denominado *anomaly_representation.kml* a partir de los datos de salida de NuPIC contenidos en el fichero *anomaly_scores.csv*.

Primero crea las cabeceras y un documento con tres estilos de icono green, yellow y red. Tras esto comienza a tomar la ruta de cada buque o aeronave identificándolo por su MMSI o su número ICAO, respectivamente, y crea un fichero para cada nuevo buque o avión. A continuación, va creando cada uno de los puntos que han sido analizados por NuPIC en la posición correspondiente (longitud, latitud y altura, en caso de los aviones) y los representa en función del valor de anomalía que han recibido (en color rojo si él valor es mayor de 0.7, en amarillo si es mayor de 0.35 y en verde el resto). Además añade a cada posición los datos que contiene el fichero de entrada. En el caso de los buques: *MMSI, longitude, latitude, speed, course, status, timestamp, anomaly_score, new_sequence y web* y para los aviones: *ICAO, name flight, altitude, heading, latitude, longitude, timestamp, anomaly_score, new_sequence, web*.

Finalmente en el último campo de cada una de las posiciones se añade un enlace a internet. Los enlaces se generan a partir de la dirección de dos páginas web añadiendo en cada caso el identificador del buque o avión al que pertenezca el dato analizado. Para los aviones es un enlace a la página web flightaware.com, donde se nos muestra toda la información de esa línea regular de vuelos, y en el caso de los buques es un enlace a la página web MarineTraffic, donde podemos ver toda la información referente al buque en cuestión. Además para los buques se añade a la posición analizada el estado que transmiten a través de AIS.

4 RESULTADOS Y DESARROLLO DE LAS PRUEBAS

4.1 Resultados de la optimización del modelo marítimo

En este proceso se ha partido de los parámetros y funciones por defecto que presenta la aplicación *geoespatial* y se han ido adaptando, en las siguientes pruebas, aquellos cambios que beneficiaban al modelo.

Los parámetros y características que se van a modificar se explican a continuación. La escala y la forma en la que se parten las secuencias de datos (*s* y *m*, respectivamente), mencionadas con anterioridad en 3.4 Ejecución del programa. El modo de análisis del espacio, por defecto el codificador geoespacial toma una altura (igual a 0) para todos los datos (analizando un espacio en 3D), lo que a priori no es correcto, igualmente se comprueba el resultado con los indicadores al realizarlo de esta forma. La variación de los parámetros que constituyen la red neuronal contenidos en el archivo *model_params.py*, en concreto de *n*, que es el número de bits que forman el vector resultado de la codificación, y *w*, que es el número que hay activos en el vector. Por último se añaden dos codificadores escalares para la codificación del rumbo y la velocidad, así se tendrán en cuenta estos valores, ya que el codificador geoespacial toma la velocidad pero solo para ajustar el tamaño del "recuadro grande" mencionado en 2.2.3 *Geospatial Coordinate Encoder*, no propiamente como un dato de análisis.

A continuación se presentan los resultados de las pruebas más significativas.

4.1.1 *Prueba de escala*

Se procede realizando la prueba sobre los datos generados como se describe en el apartado 3.5.3 Creación de las anomalías, fijando el umbral para considerar un dato como anómalo en 0,5.

En la primera prueba que se realiza se varía el parámetro de la escala, ya que a priori es el que más parece influir en la convergencia del modelo y en cómo detecta las anomalías. Se realiza para valores de escala comprendidos entre 5 y 600 metros.

Se ha agrupado las pruebas de manera que sean más visuales y para cada una se muestra un gráfico de la velocidad de convergencia, que nos indica la rapidez con la que NuPIC se adapta a los datos y por lo tanto a los patrones, el F-Score y los valores para calcularlo: precisión y sensibilidad (explicados en 3.5.2 Optimización utilizando F-Score). En esta primera prueba se omite el gráfico de las escalas 5 y 20 ya que no se apreciaba convergencia.

Como se puede ver en la Figura 4-1 la velocidad de convergencia aumenta al ir aumentando la escala, así como muestra el F-Score (Tabla 4-1) obtenido para cada una de las pruebas, que también aumenta al ir aumentando la escala.

Escala	Precisión	Sensibilidad	F-Score (a>0,5)
5	0,074	0,960	0,138
20	0,074	0,957	0,138
40	0,073	0,937	0,136
80	0,080	0,885	0,147
120	0,103	0,741	0,180

Tabla 4-1 F-Score, precisión y sensibilidad según escala (5-120).

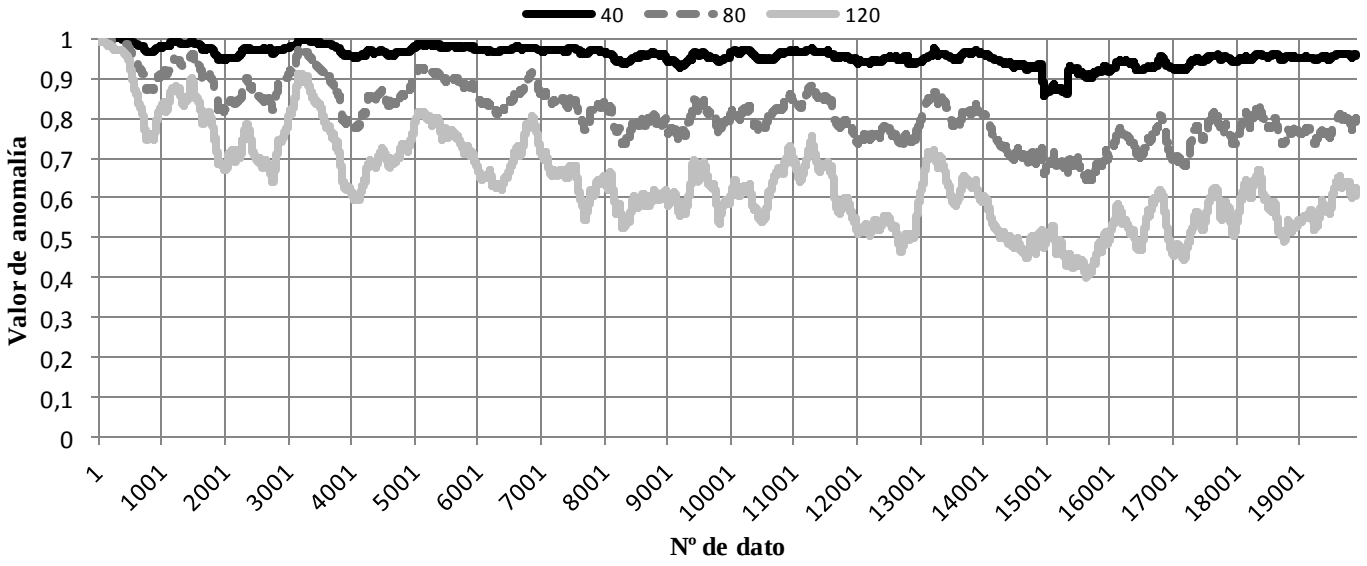

Figura 4-1 Promedio de anomalías para escalas 40, 80 y 120.

En las siguientes pruebas de escala el modelo sigue mejorando tanto su velocidad de convergencia como el F-Score aunque se precia una disminución en la velocidad de mejora de ambos indicadores. Como se puede comprobar en la Tabla 4-2 y en la Figura 4-2.

Escala	Precisión	Sensibilidad	F-Score (a>0,5)
120	0,103	0,741	0,180
160	0,120	0,544	0,197
200	0,151	0,448	0,226

Tabla 4-2 F-Score, precisión y sensibilidad según escala (5-120).

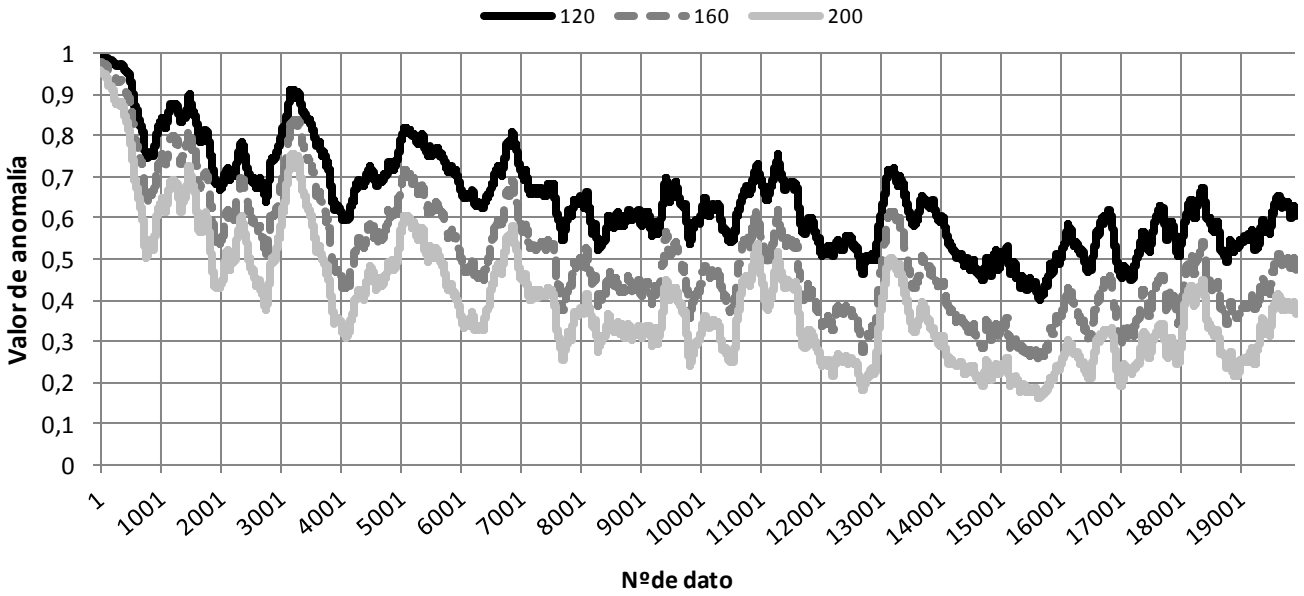

Figura 4-2 Promedio de anomalías para escalas 120, 160 y 200.

En la Figura 4-3 se observa que en la representación del promedio de anomalías entre las escalas 240 y 280 comienza a haber una menor separación y en algunos casos se empiezan a solapar ambas. Así mismo en la Tabla 4-3 vemos como el F-Score toma un valor máximo para la escala 240 y en la siguiente escala (280) ya desciende ligeramente. Por lo tanto se toma el valor de 240 como óptimo respecto al resto de escalas probadas.

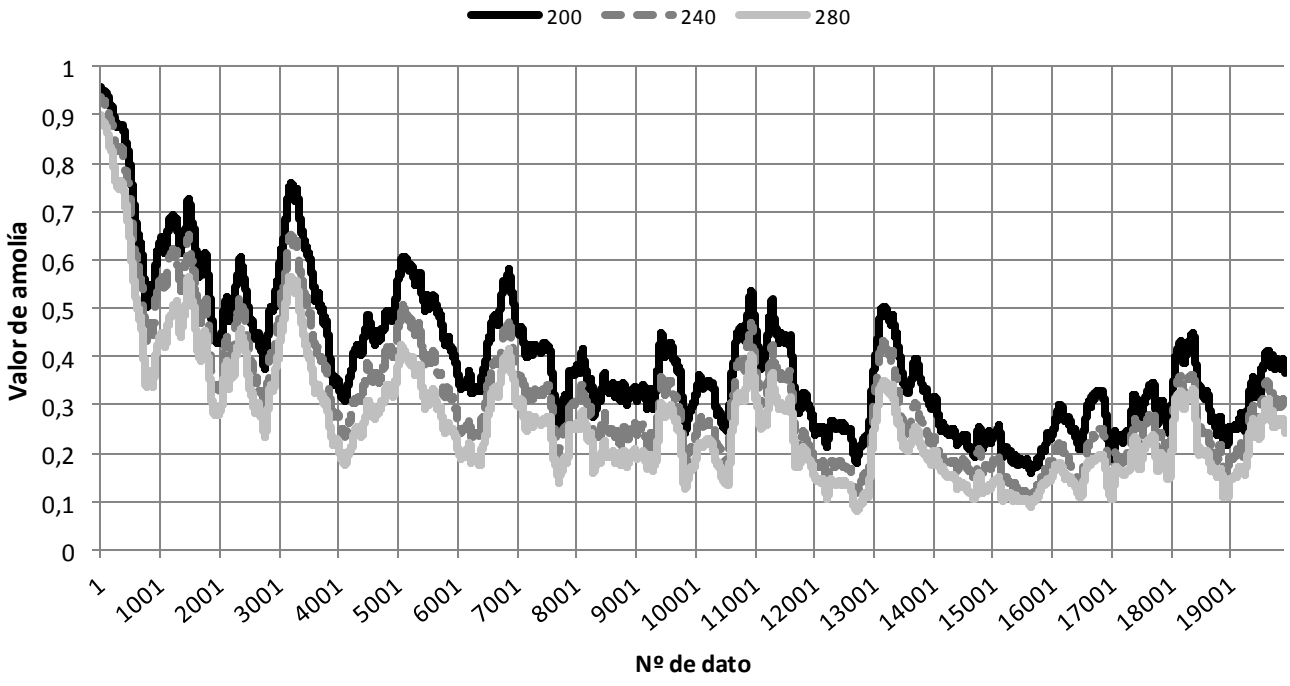

Figura 4-3 Promedio de anomalías para escalas 200, 240 y 280.

Escala	Precisión	Sensibilidad	F-Score (a>0,5)
200	0,151	0,448	0,226
240	0,181	0,389	0,247
280	0,184	0,330	0,236

Tabla 4-3 F-Score, precisión y sensibilidad según escala (200-280).

Se continúa realizando pruebas hasta el valor de escala 600. Como se puede comprobar en la Tabla 4-4 el F-Score aumenta ligeramente respecto a la prueba de 280, pero en las siguientes pruebas la tendencia es a seguir disminuyendo.

Escala	Precisión	Sensibilidad	F-Score (a>0,5)
280	0,184	0,330	0,2363
320	0,203	0,313	0,2464
360	0,209	0,282	0,2400
400	0,218	0,248	0,2247
440	0,221	0,228	0,2247
480	0,241	0,221	0,2306
520	0,270	0,207	0,2346
600	0,292	0,182	0,2245

Tabla 4-4 F-Score, precisión y sensibilidad según escala (280-600).

Por lo tanto se toma como escala óptima el valor de 240, mejorando la precisión del modelo notablemente así como el F-Score respecto a la escala utilizada por defecto (Tabla 4-5).

Escala	Precisión	Sensibilidad	F-Score (a>0,5)
5	0,074	0,960	0,138
240	0,181	0,389	0,247

Tabla 4-5 Resumen prueba de escala.

4.1.2 *Modelo en 3 o 2 dimensiones*

Entre las características del codificador geoespacial se encuentra codificar posiciones tridimensionales que contengan altura. Por defecto, al introducir una posición bidimensional, la aplicación utiliza el codificador tridimensional asignando a cada valor altura 0. Esto evidentemente, no es correcto a priori. En todo caso se realiza una prueba para analizar el efecto de utilizar un codificador bidimensional respecto a uno tridimensional, indicándose además los verdaderos positivos (Vp), los verdaderos negativos (Vn), los falsos positivos (Fp) y los falsos negativos (Fn). Se puede comprobar en la Tabla 4-6 que el F-Score es mayor al generar el modelo en 2 dimensiones. Además se observa una mejora muy importante en la velocidad de procesado del conjunto de todas las posiciones pasando de aproximadamente 20 minutos a apenas 6 minutos, lo que por otra parte es lógico. Por lo tanto, se toma como la característica del modelo.

Modelo	Vp	Vn	Fp	Fn	Total datos	Precisión	Sensibilidad	F-Score (a>0,5)
3D	227	7624	966	492	9309	0,190	0,316	0,237
2D	280	7322	1268	439	9309	0,181	0,389	0,247

Tabla 4-6 Resultados prueba 2 dimensiones o 3 dimensiones.

4.1.3 *Uso de codificador de tiempo*

Se señala en el apartado 3.4 Ejecución del programa que una de las opciones con las que cuenta la aplicación es tomar el dato de la fecha como una característica más para obtener la anomalía. Hasta ahora no se había hecho uso de esta opción. Se realiza la prueba y se compara con el último modelo obtenido. Se puede comprobar en la Tabla 4-7 que el F-Score empeora notablemente, lo que es lógico, ya que la secuencia de tránsitos en el DST no sigue ningún patrón determinado. Por lo tanto se descarta el uso de esta característica en el modelo.

Codificador de tiempo	Vp	Vn	Fp	Fn	Total datos	Precisión	Sensibilidad	F-Score (a>0,5)
Con	3	8567	25	716	9311	0,107	0,004	0,008
Sin	280	7322	1270	439	9311	0,181	0,389	0,247

Tabla 4-7 Resultados prueba codificador de tiempo.

4.1.4 *Partición de las secuencias de datos*

Una de las partes básicas del pre-análisis consiste en dividir los datos en secuencias. Por defecto se parten las secuencias por tiempo únicamente, como se explica en el apartado 3.4 Ejecución del programa. Además el umbral de división predeterminado era 5 segundos, posteriormente al estudio de los datos se fija en 34 minutos. Por otra parte, existe la opción de partirlas por nombre de ruta (en este caso por MMSI). Se suma la opción de partirlas por tiempo a que se divida cada ruta por MMSI obteniendo 3 formas de dividir los datos en secuencias. Como se puede comprobar en la Tabla 4-8 el mayor F-Score corresponde a la partición por MMSI y además por tiempo.

Partición	Vp	Vn	Fp	Fn	Total datos	Precisión	Sensibilidad	F-Score (a>0,5)
Sin partición	306	7454	1545	457	9762	0,1653	0,4010	0,2341
Por tiempo	281	7328	1402	442	9453	0,1670	0,3887	0,2336
Por tiempo y por MMSI	280	7322	1268	439	9309	0,1809	0,3894	0,2470

Tabla 4-8 Resultados prueba de partición de secuencias.

4.1.5 *Adición de codificadores*

Tras estudiar en profundidad el modelo, se observan las mismas carencias que en [13] a la hora de detectar algunos tipos de comportamientos, que están relacionados con la incapacidad de detectar anomalías relativas al rumbo: cómo circular por la vía contraria del DST o cambios bruscos del mismo. Estas carencias tienen su causa en que parte de los datos que se disponen no se utilizan en el análisis, como por ejemplo el rumbo.

Por ello se le añade un codificador escalar que analice este dato y lo incorpore al modelo. La aplicación aprende de esta forma los rumbos que son más usuales en cada uno de los espacios que se analizan. Se realiza la prueba y se compara con la anterior.

Como se comprueba en la Tabla 4-9 el F-Score aumenta significativamente con esta mejora y así lo hacen también los verdaderos positivos (número de anomalías detectadas) y los verdaderos negativos (datos sin anomalías), mientras que los falsos positivos y los falsos negativos disminuyen, mejorando el modelo en todas las detecciones.

Codificador	Vp	Vn	Fp	Fn	Total datos	Precisión	Sensibilidad	F-Score (a>0,5)
Sin	280	7322	1268	439	9309	0,181	0,389	0,247
Con	334	7407	1183	385	9309	0,220	0,465	0,299

Tabla 4-9 Resultados prueba codificador de rumbo.

Se toma este modelo como el óptimo y debido al buen resultado obtenido se plantea proceder de la misma manera con la velocidad. El codificador geoespacial sí toma la velocidad como dato, pero únicamente para codificar la posición en los SDR no en sí como un dato de análisis. Por esto, se implementa en el modelo y se realizan las pruebas.

Se comprueba que el modelo vuelve a mejorar notablemente, como se puede ver en el valor del F-Score (Tabla 4-10) debido sobre todo a la reduciendo de los falsos positivos a menos de la mitad, pero por otra parte, el número de verdaderos positivos baja de manera importante.

Codificador	Vp	Vn	Fp	Fn	Total datos	Precisión	Sensibilidad	F-Score (a>0,5)
Sin	334	7408	1184	385	9311	0,220	0,465	0,299
Con	236	8076	516	483	9311	0,314	0,328	0,321

Tabla 4-10 Resultados prueba codificador de velocidad

Debido a este resultado con los verdaderos positivos, se decide hacer otra prueba para evaluar las características del modelo y tener otro indicador más completo para comprobar el efecto que tiene implementar este codificador. La prueba que se realiza, denominada curva ROC[5], nos permite representar el modelo para los diferentes umbrales de anomalía (de 0 a 1). Como se puede observar en la Figura 4-4 el modelo con el codificador de velocidad se encuentra por encima del que no lo tiene en todo caso, por lo que se concluye que el codificador de velocidad mejora el modelo y se adapta el mismo.

[5] Curva ROC (por sus siglas en inglés *Receiver Operating Characteristic*): es la representación gráfica de la sensibilidad frente a 1- especificidad para un sistema clasificador binario, según se varía el umbral de discriminación [77].

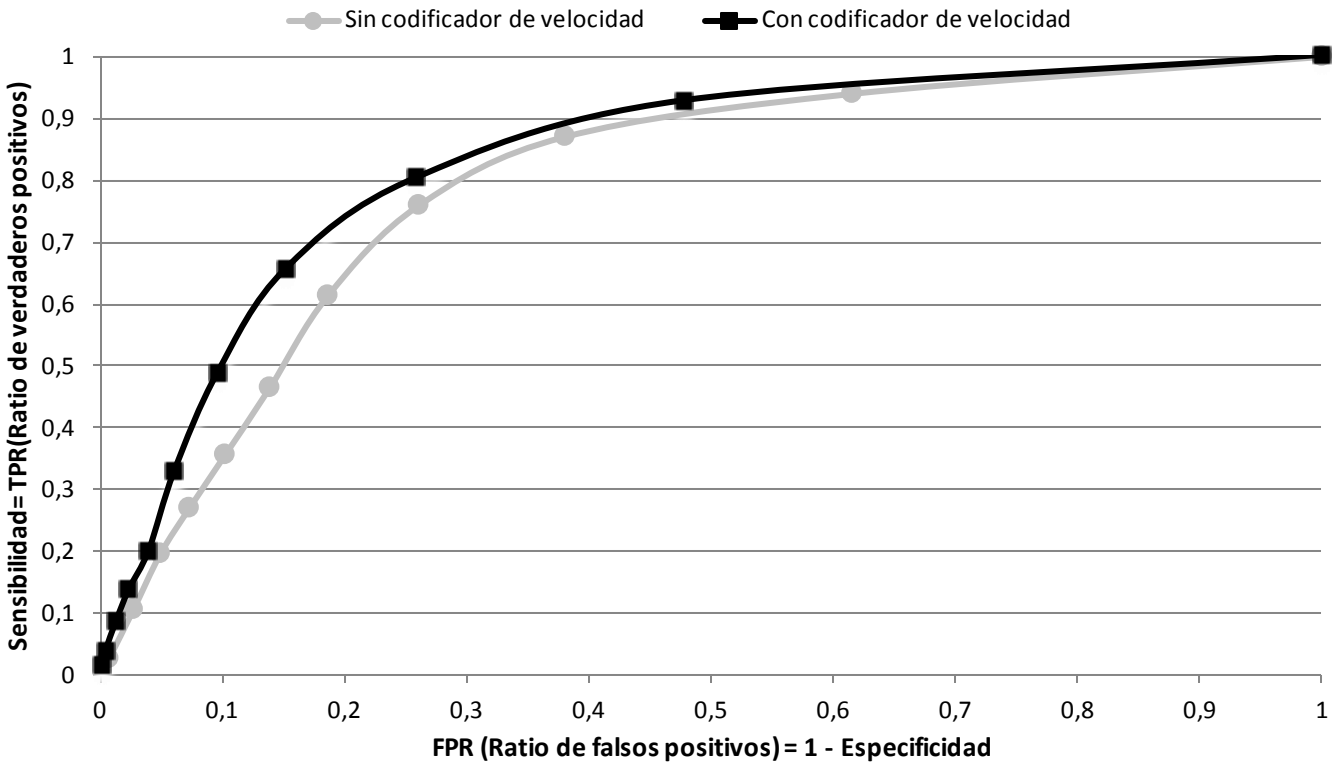

Figura 4-4 Curva ROC codificadores de rumbo y velocidad.

Con esta prueba, además, se puede obtener cuál es el umbral de anomalía para el que se obtiene un mayor F-Score y por lo tanto un modelo mejor, concluyendo que este valor es 0,35. Se comprueba que con este valor el modelo mejora notablemente pasando de un F-Score de 0,32 a 0,379.

Vp	Vn	Fp	Fn	Total datos	Precisión	Sensibilidad	F-Score
497	6706	1886	222	9311	0,209	0,691	0,320 (a>0,5)
409	7559	1033	310	9311	0,284	0,569	0,379 (a>0,35)

Tabla 4-11 Resultados prueba distintos valores anomalía.

4.1.6 *Optimización de los parámetros de los codificadores*

A continuación se realizan diferentes pruebas para ajustar los parámetros de cada uno de los codificadores añadidos. Nos vamos a centrar en los dos parámetros principales que definen cómo se codifica la información de entrada. Estos parámetros son *n* y *w*. Se realizan diferentes pruebas para comprobar cómo afecta la variación del valor de los mismos al resultado de las anomalías y cómo se interrelacionan los diferentes codificadores al variarlos.

El codificador geoespacial tiene como valores por defecto n = 2048 y w = 101, y al añadir los codificadores para la velocidad y el rumbo se introduce n=200 y w=21 en ambos casos. Esta es la condición inicial de la que se parte.

Se realizan dos conjuntos de pruebas variando de igual forma los parámetros de los codificadores de rumbo y velocidad. En el primer conjunto de pruebas solo se varían aquellos de los codificadores añadidos. En el segundo conjunto se modifican además los parámetros del codificador geoespacial, entre otros, contenidos en el archivo *model_params.py* tomando aquellos obtenidos como óptimos en [13].

Tras realizar las pruebas se observa que el mejor modelo sigue siendo el que se tenía hasta la fecha, al que se denomina condición inicial (C1).

model_params.py predeterminado					**model_params.py modificado**				
Codificadores rumbo y velocidad					**Codificadores rumbo y velocidad**				
n	**w**	**Precisión**	**Sensibilidad**	**F-Score (a>0,35)**	**n**	**w**	**Precisión**	**Sensibilidad**	**F-Score (a>0,35)**
200	21	0,244	0,69	0,360	200	21	0,208	0,207	0,208
512	21	0,156	0,794	0,261	512	21	0,238	0,274	0,255
1024	21	0,11	0,875	0,195	1024	21	0,250	0,350	0,292
2048	21	0,089	0,911	0,162	2048	21	0,208	0,207	0,208

Tabla 4-12 Resultados prueba modificación model_params.py.

En base a que los resultados obtenidos al modificar el archivo *model_params.py* no son significativos se procede modificando únicamente los codificadores. Se siguen realizando pruebas con estos parámetros y se obtiene una mejora en el caso de la prueba C2 (véase Tabla 4-13).

model_params.py predeterminado

Prueba	**Codificadores rumbo y velocidad**									
	n	**w**	**Vp**	**Vn**	**Fp**	**Fn**	**Total**	**Precisión**	**Sensibilidad**	**F-Score (a>0,35)**
C1	200	21	496	7053	1539	223	9311	0,244	0,69	0,36
	200	51	104	8589	3	615	9311	0,972	0,145	0,252
	512	21	571	5510	3082	148	9311	0,156	0,794	0,261
C2	512	51	222	8542	50	497	9311	0,816	0,309	0,448
	512	101	52	8579	13	667	9311	0,8	0,072	0,133

Tabla 4-13 Resultados prueba parámetros codificadores.

Debido al resultado obtenido al añadir el codificador de la velocidad, donde el modelo mejora globalmente pero el número de anomalías detectadas disminuye de manera importante, se pretende dar una menor importancia al codificador de la velocidad al calcular la anomalía.

En 2.2.2 NuPIC (*Numenta Platform for Intelligent Computing*) se explica cómo se calcula el valor de la anomalía y en [72] se puede comprobar que la forma en que se pondera el efecto de cada codificador es mediante la longitud de su vector, con lo cual variando la longitud de la codificación se varía el peso de cada una de las contribuciones. Considerando esto, las siguientes pruebas pretenden dar menos peso al codificador de velocidad en el conjunto del modelo. Se toma el mejor modelo hasta el momento, es decir con la condición inicial (C1) mencionada anteriormente y se realizan las pruebas.

Como se puede comprobar en la Tabla 4-14 el F-Score mejora nuevamente para dos de las pruebas, correspondientes ambas a disminuir la longitud del vector (*n*).

Prueba	Condiciones en los codificadores						Resultados		
	Geoespacial		Rumbo		Velocidad				
	n	w	n	w	n	w	Precisión	Sensibilidad	F-Score (a>0,35)
C1	2048	51	200	21	200	21	0,276	0,569	0,371
C3	2048	51	200	21	100	21	0,326	0,604	0,423
C4	2048	51	200	21	50	21	0,378	0,624	0,471
C2'	2048	51	512	21	50	21	0,425	0,345	0,381
	4096	101	200	21	50	21	0,207	0,473	0,288
	4096	101	512	51	50	21	0,346	0,488	0,405
	8128	201	400	41	50	21	0,206	0,317	0,250
	8128	201	600	51	50	21	0,237	0,337	0,278

Tabla 4-14 Resultados pruebas finales.

Por último, se comparan todos los indicadores de las pruebas que presentan un mayor F-score con respecto a la condición inicial (C1). Se comprueba que para la prueba C4 además de mejorar el F-Score el número de verdaderos positivos y verdaderos negativos aumenta significativamente, que es lo que se estaba buscando, y que el número de falsos positivos y falsos negativos disminuye por lo que el modelo mejora en las dos características (sensibilidad y precisión) y en todas las detecciones. Se concluye que este modelo es el que mejor resultados obtenido y se toma como óptimo.

Prueba	Vp	Vn	Fp	Fn	Total	Precisión	Sensibilidad	F-Score (a>0,35)
C1	409	7562	1074	310	9355	0,276	0,569	0,371
C2	222	8542	50	497	9311	0,816	0,309	0,448
C3	434	7737	899	285	9355	0,326	0,604	0,423
C4	449	7898	738	270	9355	0,378	0,624	0,471

Tabla 4-15 Comparación pruebas mayor F-Score.

A continuación se comparan los resultados finales con respecto a la primera prueba de análisis realizada y con respecto a los obtenidos con los parámetros de [13] pero habiendo añadido ya los codificadores de rumbo y velocidad. Además, en la primera prueba ya se suponían ciertas características como que el espacio se debía analizar en 2D y no en 3D (aunque estuviera así por defecto). Como se observa en la Tabla 4-16 el modelo se ha mejorado notablemente pasando el F-Score de 0,14 a 0,47. La precisión con respecto a la primera prueba se incrementa en más del 30%. En cuanto a la sensibilidad (la prueba inicial es más sensible ya que detecta todo como anómalo), se encuentra en torno al 60 % con una mejora de más del 40 % frente a la prueba realizada con los parámetros de [13].

Prueba	Precisión	Sensibilidad	F-Score (a>0,35)
Prueba inicial	0,074	0,960	0,138
Parámetros de [13]	0,208	0,207	0,208
C4	0,378	0,624	0,471

Tabla 4-16 Comparación de resultados.

4.1.7 *Resultados cualitativos*

A continuación se presentan gráficamente algunas de las anomalías generadas. En concreto se han seleccionado 5 consideradas como las más explicativas.

En la primera, el buque navega de manera correcta hasta que realiza un cambio de rumbo brusco al oeste en un lugar donde el tráfico continúa con el mismo rumbo. La aplicación detecta claramente el cambio de rumbo inusual en ese punto como se puede ver en la Figura 4-5. Posteriormente, al tomar el buque un rumbo mantenido, el modelo se adapta a este nuevo rumbo y el valor de anomalía comienza a disminuir. Aunque las últimas posiciones se etiquetan como poco anómalas, la inclusión del rumbo como característica de entrada permite detectar el comportamiento como totalmente anómalo al poco de producirse, lo que supone una mejora importante respecto al trabajo en [13].

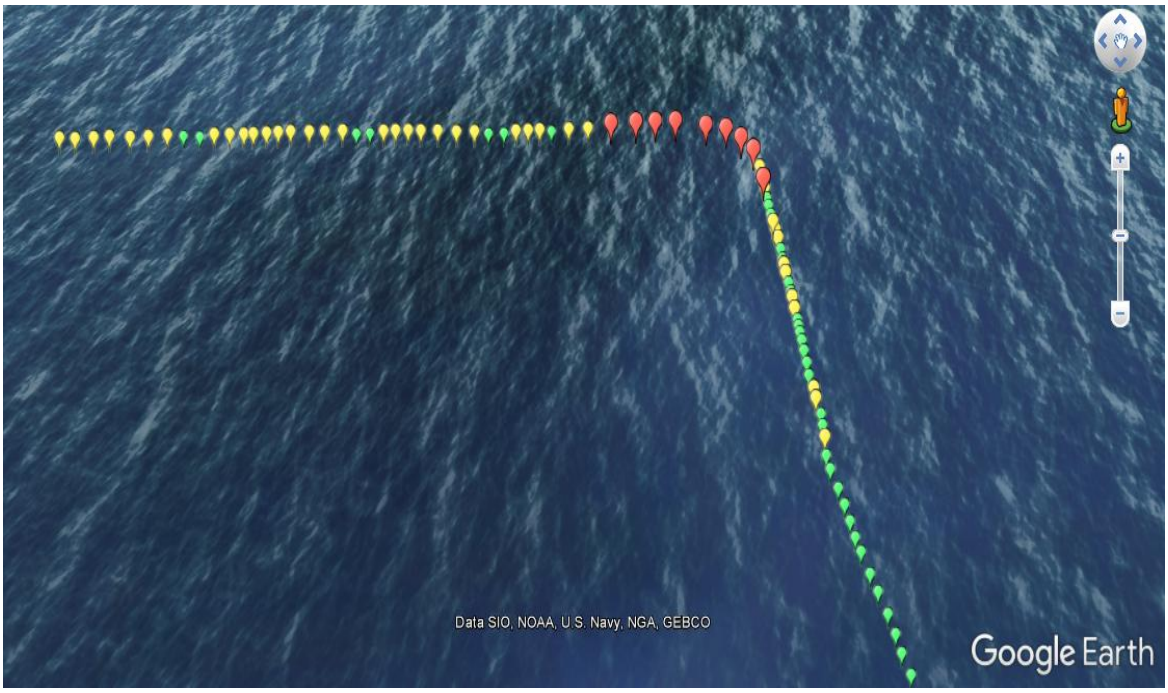

Figura 4-5 Visualización 1º anomalía.

En el segundo caso elegido (Figura 4-6) el buque continua a rumbo constante (rumbo aproximadamente norte) mientras que el tráfico realiza una caída de rumbo (ejemplo de la caída que realiza el tráfico en las dos rutas que se dirigen más hacia estribor). Como se comprueba, esta anomalía nuevamente debida al rumbo se detecta perfectamente, además en este caso el comportamiento del buque es mucho más sutil ya que la diferencia de rumbo es aproximadamente 25º respecto al resto del tráfico.

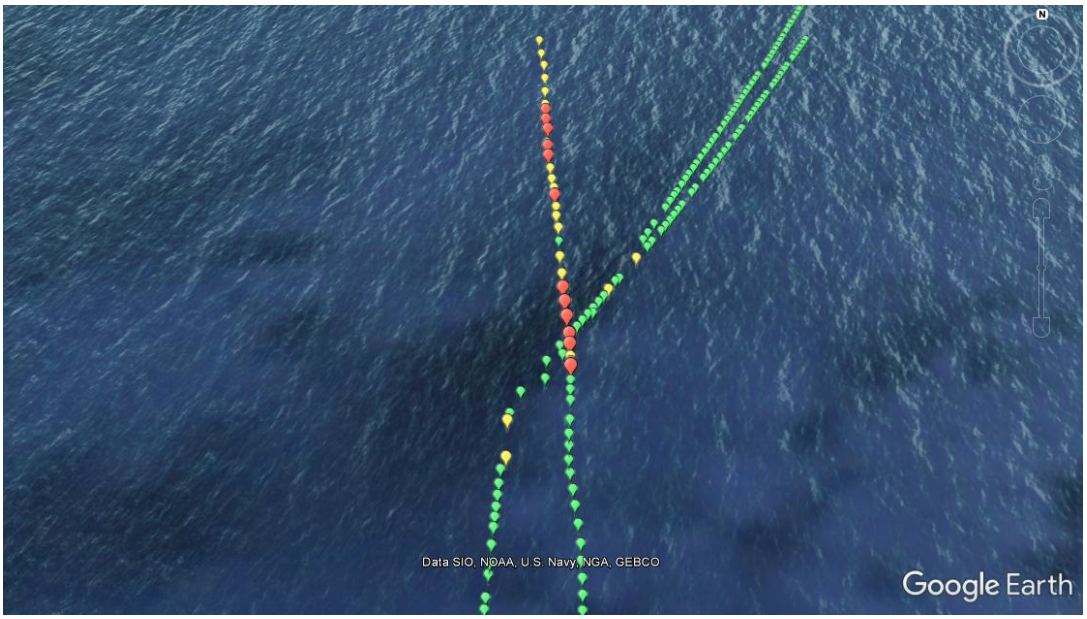

Figura 4-6 Visualización 2º anomalía.

En el tercer caso (Figura 4-7) el buque intenta entrar en el DST a rumbo contrario al que lleva el tráfico (en este caso el tráfico está saliendo por la parte norte del DST). Cuando todavía no ha entrado en el mismo ya se identifica como anómalo y, tras entrar en él, continúa detectándose como anómalo. Con lo que podemos comprobar que nuevamente detecta este rumbo poco usual en la zona, concluyendo que la detección de un rumbo inusual está integrada.

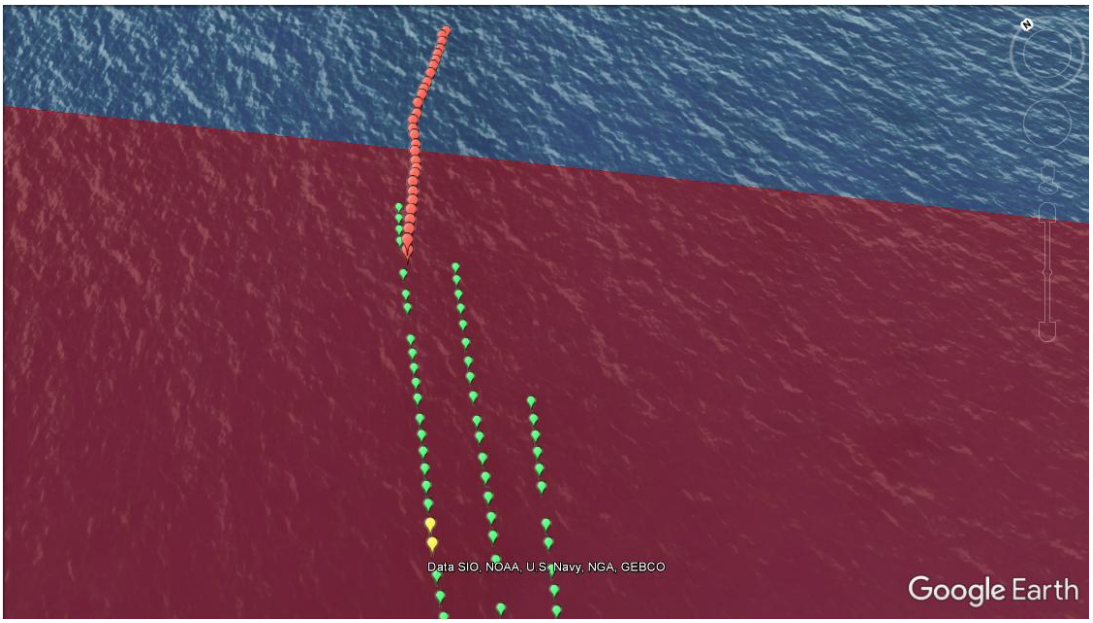

Figura 4-7 Visualización 3º anomalía.

En el cuarto caso, el buque navega a rumbo constante y a una velocidad de acuerdo a como lo hace el tráfico. Posteriormente disminuye la velocidad de 10 a 5 nudos en el punto señalado en la Figura 4-8. Como se puede comprobar la aplicación no detecta ninguna anomalía en ese punto, probablemente porque el tráfico sí navega en ocasiones a 5 nudos dentro del DST.

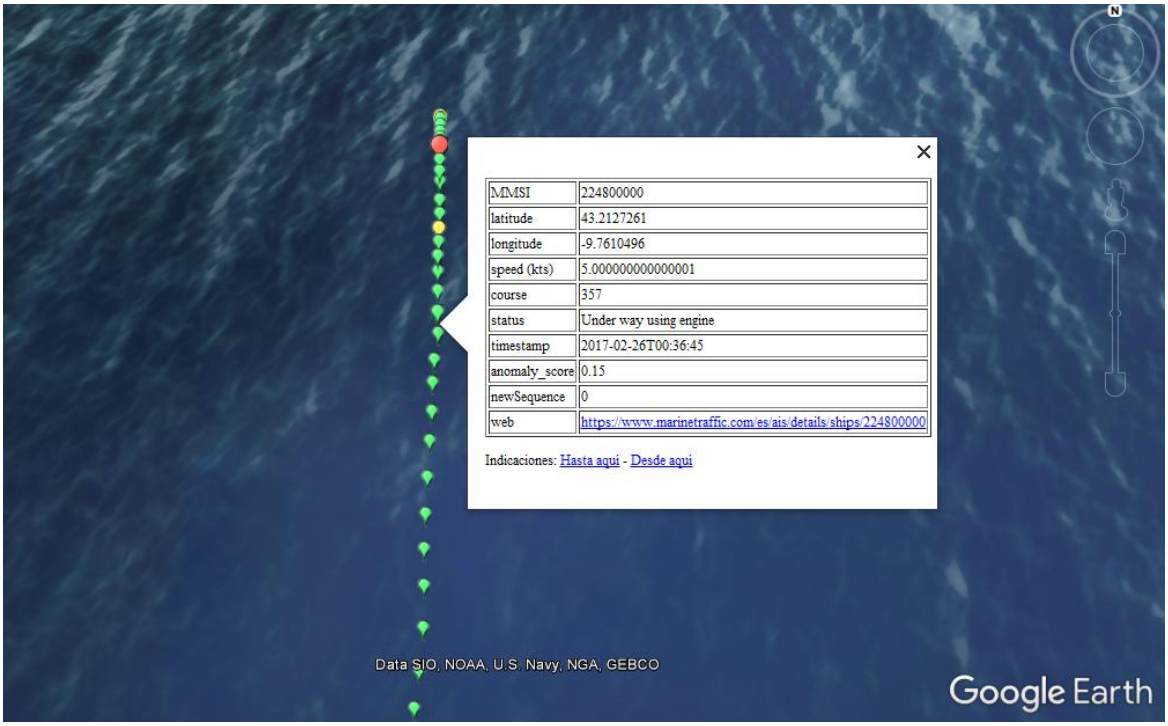

Figura 4-8 Visualización 4º anomalía (1).

Tras esto disminuye la velocidad hasta llegar a 2 nudos (véase Figura 4-9), en este caso sí se detectan como anómalas las dos posiciones siguientes a esta disminución de velocidad pero posteriormente toma como normal este comportamiento. Nuevamente se vuelve a mejorar la capacidad de detección respecto a [13], en este caso de una velocidad excesivamente baja.

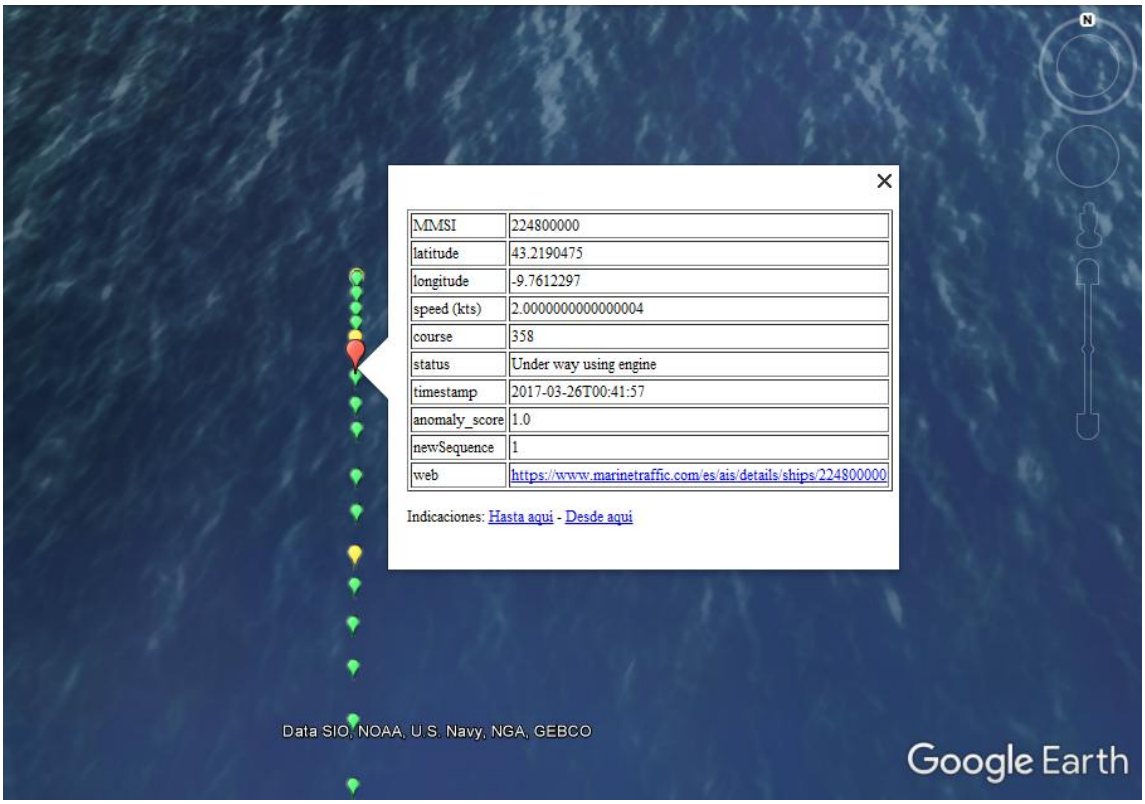

Figura 4-9 Visualización 4º anomalía (2).

Finalmente, el buque reduce velocidad hasta detenerse prácticamente por completo. Como se puede ver en la Figura 4-10 la aplicación detecta la primera posición como anómala pero con una valor de anomalía bajo. Además, posteriormente ya no detecta ninguna de las posiciones como anómalas a pesar de que el buque se encuentre prácticamente parado.

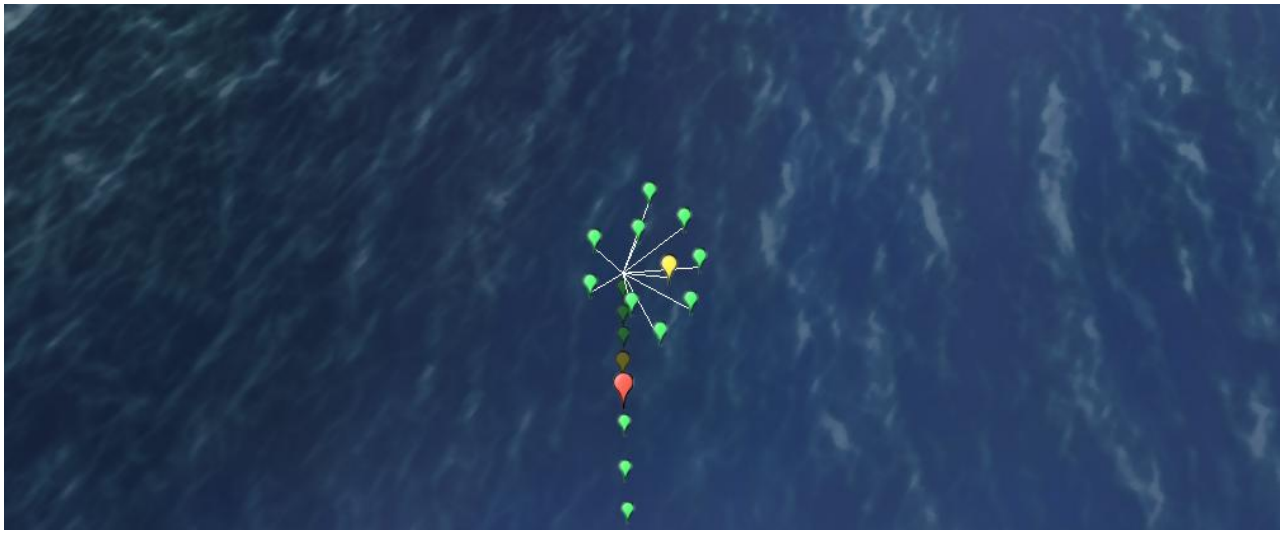

Figura 4-10 Visualización 4º anomalía (3).

En el último caso de estudio el buque navega a velocidad normal y posteriormente aumenta a 15 nudos en el punto indicado en la Figura 4-11. La aplicación no lo detecta aunque en el siguiente punto señala un valor algo más elevado de anomalía.

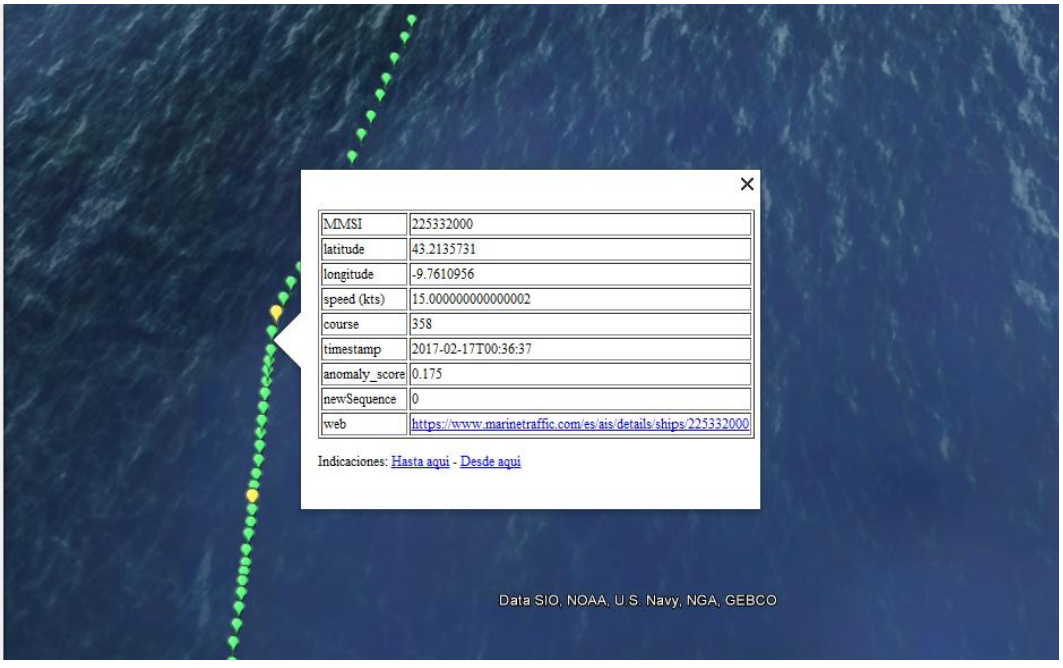

Figura 4-11 Visualización 5º anomalía (1).

El buque continúa aumentando de velocidad hasta 25 nudos, pero no se detecta ninguna anomalía en este caso. Posteriormente, aumenta hasta 30 nudos en el punto señalado en la Figura 4-12. En este

caso sí detecta este aumento de velocidad como una anomalía presentandadolo en rojo y también la siguiente posición aunque con un valor bajo de anomalía. Se debe recalcar que la circulación en los DST es mucho más continúa y en general a mayor velocidad que en costeras. Posteriormente el modelo comienza a adaptar esta nueva conducta y disminuye el valor de la anomalía.

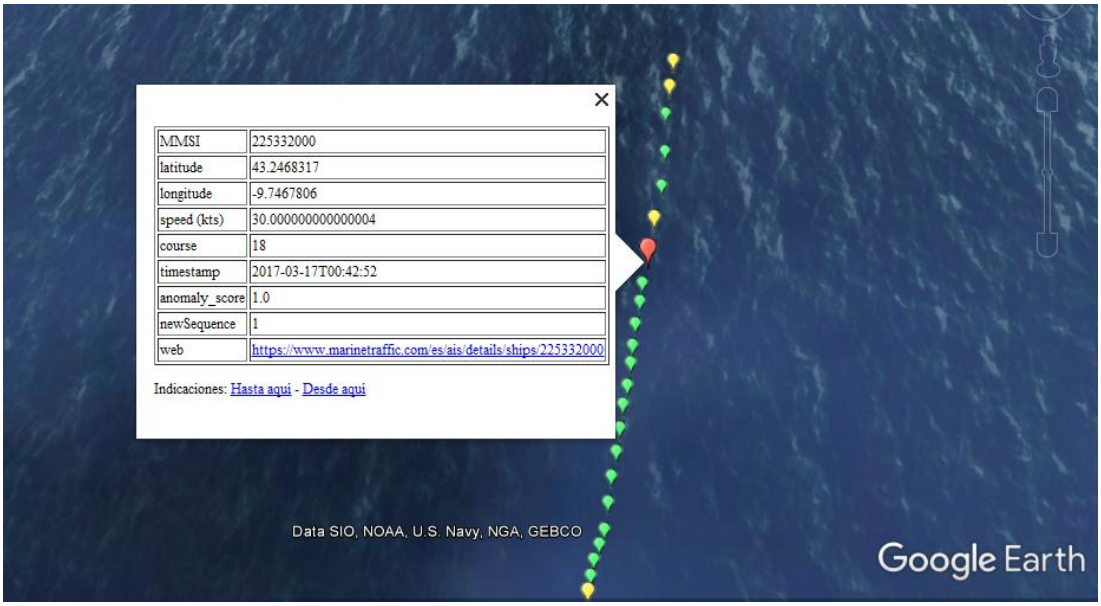

Figura 4-12 Visualización 5º anomalía (2).

En resumen, se han mejorado importantes carencias respecto al modelo de [13]. La detección de anomalías respecto al rumbo está afianzada obteniendo unos resultados muy buenos en diferentes condiciones. En cuanto a la velocidad se observan mejoras importantes aunque únicamente cuando la diferencia con las posiciones anteriores y con la del tráfico es abrupta.

4.2 Modelo aéreo

Se toman las optimizaciones realizas en el modelo marítimo y se aplican a este caso los mismos parámetros a los codificadores. En cuanto a la escala se realizan diferentes pruebas para comprobar cuál es la óptima para este escenario. *A priori* se puede pensar que esta debe ser mayor ya que las aerovías tienen una anchura de unas 10 millas náuticas, mientras que un sentido de circulación en el DST tiene aproximadamente 5 millas. Además en este caso se analiza en 3 dimensiones por lo que el espacio a analizar aún es mayor. A continuación se presentan los resultados del modelo aéreo.

4.2.1 *Pruebas de escala*

En la Tabla 4-17 se puede observar la media de los resultados de anomalías para las diferentes escalas además de la diferencia con el resultado obtenido en la escala anterior. Podemos comprobar que al disminuir la escala así lo hace también la media de anomalías (como es lógico). En el valor 900 vemos un punto de inflexión y partir de este el descenso es poco significativo. Para corroborar este resultado se calcula el promedio de anomalías cada 400 muestras y se obtiene la gráfica de convergencia del modelo en cada uno de los supuestos como se realizo en el modelo marítimo.

Escala	Media anomalías	Diferencia
200	0,592	----------
300	0,580	-0,012
400	0,563	-0,017
500	0,520	-0,043
600	0,469	-0,051
700	0,435	-0,034
800	0,407	-0,028
900	0,343	-0,064
1000	0,325	-0,018
1200	0,287	-0,039
1400	0,245	-0,041
1600	0,245	0,000

Tabla 4-17 Resultados pruebas de escala ámbito aéreo.

Se comprueba en la Figura 4-13 que no hay una convergencia significativa del modelo, posiblemente porque el espacio a analizar es sumamente grande y la aplicación no ha recibido los suficientes datos. Por ello, aunque parece converger, finalmente no lo hace debido a que se le entregan datos de diferentes zonas cada vez. Además, prácticamente no se aprecia diferencia entre las representaciones de cada uno de los valores de la escala.

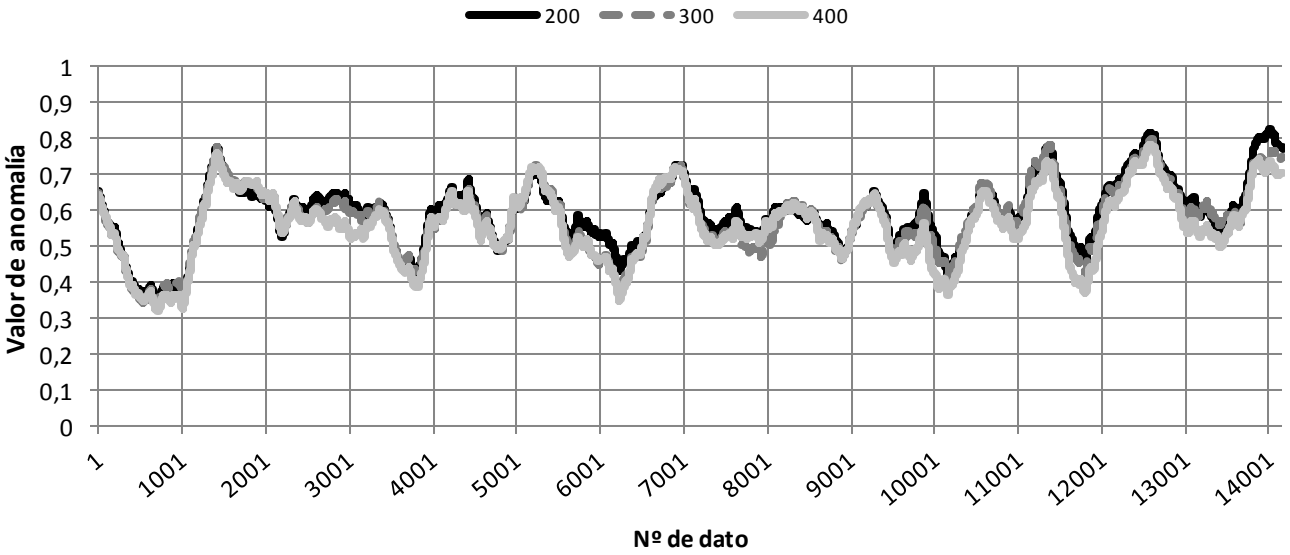

Figura 4-13 Promedio de anomalías para escalas de 200 a 400.

En esta segunda gráfica (Figura 4-14) podemos apreciar una mayor diferencia entre las escalas, especialmente entre la de 500 y 600. Al igual que indicaba la media de anomalías.

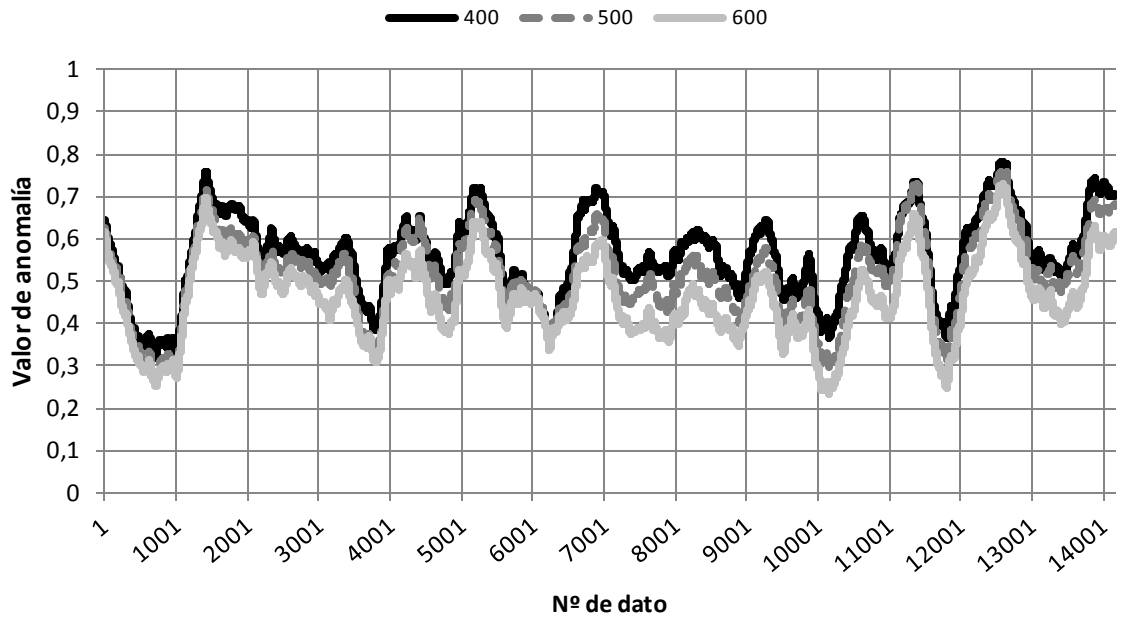

Figura 4-14 Promedio de anomalías para escalas de 400 a 600.

Finalmente en esta última representación se aprecia una diferencia algo más significativa entre la escala de 800 y la de 900, no así con la de 1000 que prácticamente se encuentra solapando a la de 900. Esto concuerda con lo que se obtuvo en el valor de la media de anomalías para esta escala.

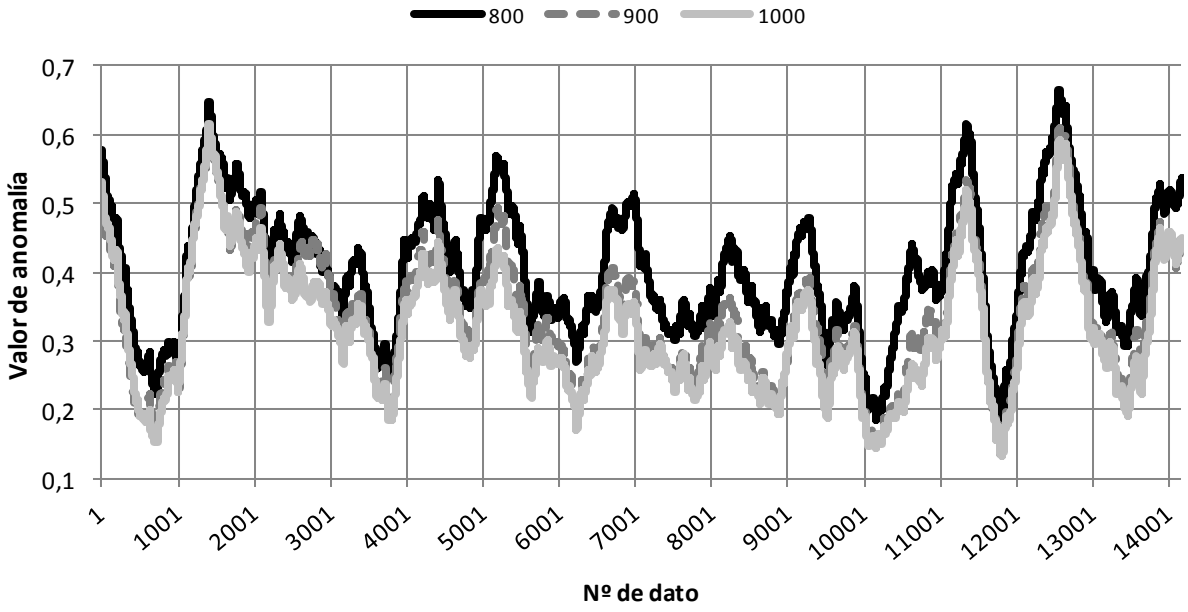

Figura 4-15 Promedio de anomalías para escalas de 800 a 1000.

En base a los resultados, y especialmente por los resultados obtenidos en la Tabla 4-17 se decide coger 900 como escala para analizar las rutas de aeronaves.

4.2.2 *Resultados cualitativos*

A continuación se presenta un ejemplo donde un avión realiza un despegue desde el aeropuerto de Vigo para incorporarse a una de las aerovías que cubren el cielo gallego, destacándose las partes más significativas de este ejemplo.

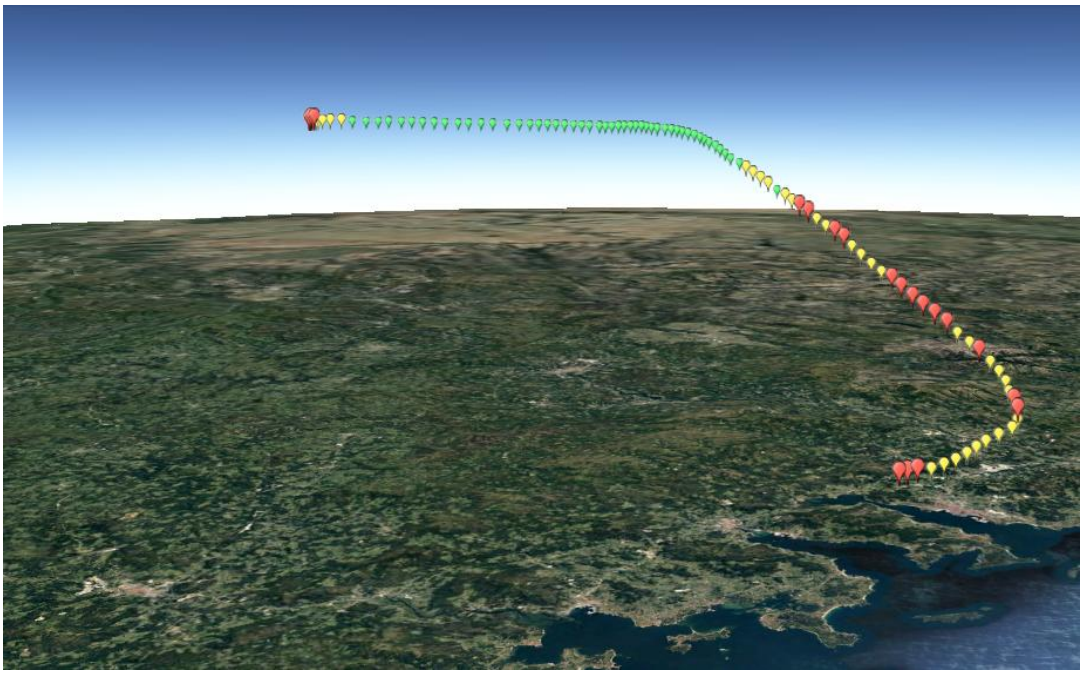

Figura 4-16 Visualización ruta aviones (1).

Como se puede comprobar en la Figura 4-16, el tramo de despegue e incorporación a la aerovía lo detecta prácticamente como totalmente anómalo. Esto es normal ya que la aplicación probablemente no haya visto ninguna otra aeronave realizando esta incorporación y por lo tanto lo toma como inusual.

Posteriormente el avión continúa ascendiendo y se dirige hacia la aerovía. Se observa en la Figura 4-17 que cuando el avión entra en la aerovía, la aplicación comienza a entregar unos resultados no anómalos, como se esperaba. La aeronave continúa ascendiendo hasta situarse a 34000 pies en la aerovía y prosigue su ruta. Durante la misma obtenemos resultados positivos del análisis ya que todas las posiciones dentro de la aerovía son no anómalas.

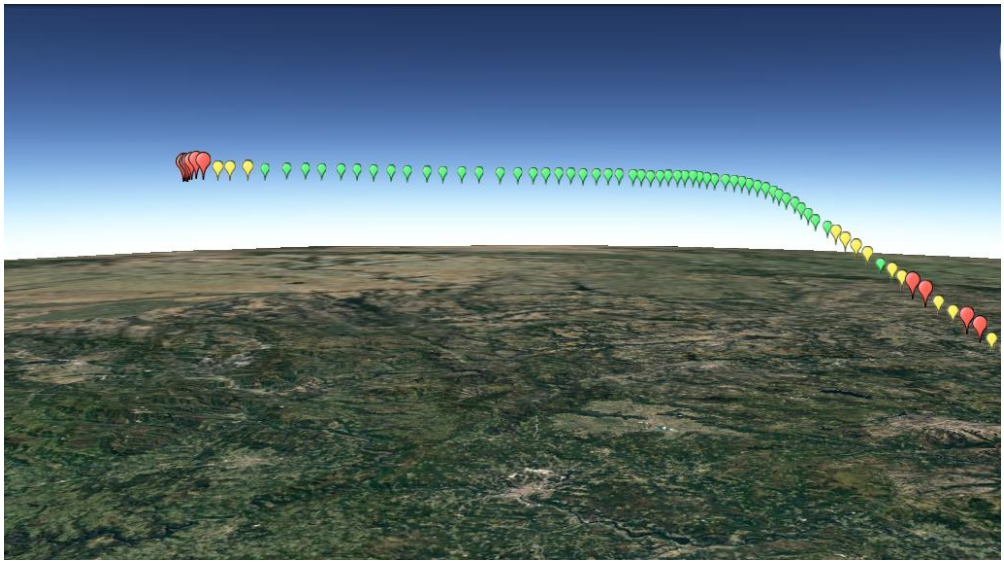

Figura 4-17 Visualización ruta aviones (2).

Posteriormente el avión realiza un giro imprevisto saliéndose de la aerovía. En la Figura 4-18 podemos ver que este giro se detecta como totalmente anómalo. La aeronave continúa a rumbo hasta salirse de la aerovía lo que sigue identificándose como anómalo como es de esperar.

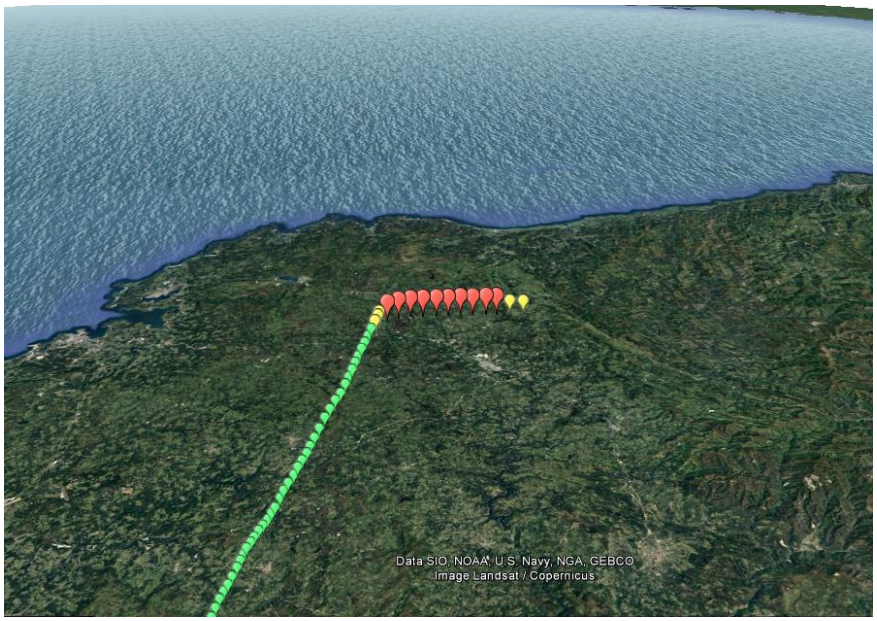

Figura 4-18 Visualización ruta aviones (3).

Los resultados obtenidos del análisis del ámbito aéreo son positivos, detectándose de manera acertada tanto la entrada del avión en la aerovía como su salida de la misma debido a un cambio de rumbo brusco. Concluyendo que se puede aplicar este sistema al análisis y vigilancia del espacio aéreo.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Cumplimiento de los objetivos y conclusiones

A continuación, se repasan los objetivos mencionados en la primera sección, comentando su grado de cumplimiento, y se presentan los resultados objetivos. Después se presentan unas conclusiones generales del trabajo y por último unas personales.

En primer lugar, se han estudiado los datos y determinado el grado de idoneidad de sus características para ser analizadas mediante el motor seleccionado, obteniendo un resultado positivo y por ello aplicando el sistema de inteligencia artificial.

Se ha instalado NuPIC y se ha adaptado la aplicación *geospatial*, para poder procesar la información de los dos espacios que se pretenden analizar. Además, se ha mejorado el preprocesado de los datos para entregarlos a la aplicación, de forma que pueda realizar el análisis correctamente.

Se ha realizado la instalación del *swarming* pero debido a la falta de implementación del mismo en datos geoespaciales se ha procedido mediante una alternativa: utilizar técnicas de optimización como el F-Score y la curva ROC. Para ello se han estudiado los diferentes parámetros y características de la aplicación y se han desarrollado unas pruebas para usar las técnicas mencionadas. Como resultado se ha optimizado la aplicación satisfactoriamente.

La obtención de datos ADS-B se ha realizado favorablemente de forma autónoma e independiente, y se han adaptado los datos para que puedan ser procesados por NuPIC. Por último, se representan los datos para identificar rápidamente los resultados del análisis.

En cuanto a los resultados:

-Se ha disminuido el tiempo de procesamiento notablemente aplicando la codificación en 2D, en el caso de los buques.

-Se detectan anomalías por cambios de rumbo, que no eran detectadas por la aplicación previa.

-Se han conseguido incorporar nuevos codificadores (rumbo y velocidad) y con ello se han mejorado las prestaciones de manera evidente.

Finalmente, con todas las mejoras, se ha aumenta el valor de los indicadores utilizados para el análisis perfeccionando la aplicación.

Como conclusiones, se observa que cuanta más cantidad de datos (no por el número sino por lo que representan) se le entregue a la aplicación para analizar, mayor es su rendimiento. Esto se debe a que con mayor cantidad de información realiza mejor el aprendizaje de los patrones.

La aplicación positiva del receptor SDR abre la posibilidad de llegar a aplicar sistemas de inteligencia artificial a las unidades de la Armada y a los centros de control y vigilancia del tráfico tanto aéreo como marítimo.

En cuanto a mis conclusiones personales, he descubierto una nueva tecnología que aunque, conocía superficialmente, no tenía idea de las capacidades que puede llegar a tener, a pesar de que aún se encuentra en desarrollo.

5.2 Líneas futuras

La inteligencia artificial y en concreto HTM se encuentran en continuo desarrollo, por ello es interesante seguir de cerca la evolución de la misma para adaptar sus avances a los modelos que se quieran crear. Además, como se ha comprobado en su página web y su foro, la comunidad de NuPIC es muy activa, así se insta a que se continúe de cerca su evolución.

La API de MarineTraffic, como se explicó con anterioridad, proporciona otro tipo de mensajes con diferentes datos. Parece interesante tomar estos datos e implementarlos en el modelo para que así la aplicación pueda tener en cuenta el tipo de barco, último puerto de salida o a dónde se dirige. Esto permite crear un modelo específico para cada buque o conjunto de ellos con características y comportamientos similares mejorando las prestaciones.

Otra vía de mejora obligatoria es estudiar en mayor profundidad el funcionamiento del corazón de NuPIC para obtener con certeza cuál es la función de cada uno de los diferentes parámetros que rigen el desarrollo del análisis. En concreto, todos aquellos dentro del archivo de *model_params.py* y dentro de cada uno de los codificadores. Relacionado con esto, se aprecia que la aplicación aprende demasiado rápido las conductas anómalas. Por ello, sería de utilidad disminuir la velocidad de aprendizaje.

Otro forma de continuar el estudio es profundizar en el análisis del entorno aéreo estudiando la cartografía para identificar claramente las aerovías y adaptar el modelo a la situación de las mismas, teniendo en cuenta además la situación geográfica de los distintos aeropuertos de la zona que se quiera analizar.

Por otra parte, parece interesante realizar el estudio de únicamente una aeronave que realice siempre el mismo trayecto, pidiendo datos de la misma a uno de los muchos servidores de internet como flightradar.com para adaptar el modelo en concreto a un caso con un horario y una ruta concreta y así también obtener un mejor conocimiento de los patrones de comportamiento de las aeronaves. Los datos podrían obtenerse de alguno de los sitios Internet que proporcionan este servicio, como flightradar.com.

Por último, se podría proceder de forma similar a como se ha hecho con los buques y generar artificialmente rutas que se salgan del patrón de comportamiento de las aeronaves para observar diferentes indicadores de análisis y cómo influyen en este caso los diferentes parámetros y funciones.

6 BIBLIOGRAFÍA

En esta sección figurarán todas las referencias utilizadas en el trabajo.

[1] Secretario General Kitack Lim, «Organizacion Maritima Internacional,» [En línea]. Available: http://www.imo.org/es/MediaCentre/Multimedia/Video/Paginas/Default.aspx?playerUrl=rPSRr7UaFzM&autoplay=true&title=World Maritime Day Forum - Summary. [Último acceso: 20 Enero 2017].

[2] Documento informativo DÍA MARÍTIMO MUNDIAL 2016, «Organizacion Maritima Internacional,» [En línea]. Available: http://www.imo.org/en/About/Events/WorldMaritimeDay/Documents/World%20Maritime%20Day%202016_Background%20paper%20(SPANISH).pdf. [Último acceso: 18 Enero 2017].

[3] «Naciones Unidas, Conferencia sobre comercio y desarrollo,» [En línea]. Available: http://unctad.org/es/PublicationsLibrary/rmt2015_es.pdf. [Último acceso: 20 Enero 2017].

[4] «El Pais,» [En línea]. Available: http://elpais.com/elpais/2015/03/06/media/1425661983_825046.html. [Último acceso: 21 Enero 2017].

[5] «Ministerio de fomento,» [En línea]. Available: http://observatoriotransporte.fomento.es/NR/rdonlyres/0AE839CF-9E00-46F3-A27C-88B14AC37715/136237/INFORME_OTLE_2015.pdf. [Último acceso: 21 enero 2017].

[6] «Salvamento Marítimo,» [En línea]. Available: http://www.salvamentomaritimo.es/. [Último acceso: 21 Enero 2017].

[7] Puertos del Estado, «Sistema Portuario Español,» [En línea]. Available: http://www.tecniberia.es/jornadas/documentos/FernandoGzlLaxe_PuertosdelEstado.pdf. [Último acceso: 21 Enero 2017].

[8] Departamento de Seguridad Nacional, «Estrategia de Seguridad Marítima Nacional,» 2013.

[9] José Javier Muñoz Castresana (Tcol del Ejército del Aire), «EL CONTROL DEL ESPACIO AÉREO EN SITUACIONES DE PAZ, CRISIS Y GUERRA».

[10] «ENAIRE,» [En línea]. Available: http://www.enaire.es/csee/Satellite/navegacion-aerea/es/Navegacion-Aerea.html. [Último acceso: 12 Febrero 2017].

[11] «Ejercito del Aire,» [En línea]. Available: http://www.ejercitodelaire.mde.es/ea/pag?idDoc=6E72BD61A3CAA925C1257448003B096B. [Último acceso: 12 Febrero 2017].

[12] «Foro de cultura de defensa,» [En línea]. Available: http://forodeculturadedefensa.blogspot.com.es/2013/10/escuadrones-de-vigilancia-aerea-eva-los.html. [Último acceso: 12 Febrero 2017].

[13] C. A. Pérez-Seoane, «Desarrollo de un sistema de inteligencia artificial para la supervisión y detección de anomalías en rutas marítimas,» 2016.

[14] «Sotavento escuela naútica,» [En línea]. Available: http://www.sotaventonline.com/dispositivos-separacion-trafico/. [Último acceso: 17 Febrero 2017].

[15] West coast of Spain and Portugal pilot, Admiralty sailing directions, 2011.

[16] «Google Maps,» [En línea]. Available: https://www.google.es/maps. [Último acceso: 17 Febrero 2017].

[17] Stuart J. Russell and Peter Norvig, Artificial Intelligence. A Modern Approach, New Jersey: Prentice-Hall, 1995.

[18] Departamento de ciencas de la computación, «Universidad de Buenos Aires,» [En línea]. Available: https://www.dc.uba.ar/materias/aa/2011/cuat2. [Último acceso: 25 Enero 2017].

[19] Warren S. McCulloc, Walter H. Pitts, «A logical calculus of the ideas immanent in nervous activity,» de *The bulletin of mathematical biophysics*, Chicago, Kluwer Academic Publishers, 1943, pp. 115–133, Volume 5, Issue 4, .

[20] A. M. Turing, «Computing Machinery and Intelligence,» *Mind*, vol. 59, nº 236, pp. 433-460, 1950.

[21] «Universidad Carlos III de Madrid,» [En línea]. Available: http://ocw.uc3m.es/ingenieria-telematica/inteligencia-en-redes-de-comunicaciones/material-de-clase-1/01-historia-de-la-inteligencia-artificial. [Último acceso: 25 Enero 2017].

[22] John McCarthy, Marvin L. Minsky, Nathaniel Rocheste, and Claude E. Shannon, «A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence,» 1955.

[23] F. Rosenblatt, «THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN,» *Psychological Review*, vol. 65, nº 6, 1958.

[24] Joseph Weizenbaum , «ELIZA—a computer program for the study of natural language communication between man and machine,» *Communications of the ACM* , vol. 9, pp. 36-45, 1966.

[25] [En línea]. Available: http://www.manifestation.com/neurotoys/eliza.php3. [Último acceso: 17 Febrero 2017].

[26] L. Amador Hidalgo, Inteligencia Artificial y Sistemas Expertos, Córdoba: Universidad de Cordoba, 1996.

[27] «IBM,» [En línea]. Available: https://www.research.ibm.com/deepblue/. [Último acceso: 28 Enero 2017].

[28] «Abc,» [En línea]. Available: http://www.abc.es/archivo/20140501/abci-deep-blue-kasparov-201404301111.html. [Último acceso: 24 Enero 2017].

[29] «UNIVERSITY OF WISCONSIN–MADISON- Computer Science,» [En línea]. Available: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html. [Último acceso: 17 Febrero 2017].

[30] «Xakata,» [En línea]. Available: https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial. [Último acceso: 25 Enero 2017].

[31] «Nethunting,» [En línea]. Available: http://www.nethunting.es/wp-content/uploads/2016/12/ia_nethunting_01.jpg. [Último acceso: 10 Febrero 2017].

[32] Instituto de Investigación en Inteligencia Artificial (CSIC), «Fundacion General del CSIC,» [En línea]. Available: http://www.fgcsic.es/lychnos/es_es/articulos/inteligencia_artificial. [Último acceso: 28 Enero 2017].

[33] Fernando Sancho Caparrini , «Dpto. de Ciencias de la Computación e Inteligencia Artificial,» [En línea]. Available: http://www.cs.us.es/~fsancho/?e=77. [Último acceso: 12 Febrero 2017].

[34] Nelson Ubaldo Quispe Mamani , «Monografias: Redes Neuronales Artificiales, Metodología de Desarrollo y Aplicaciones,» [En línea]. Available: http://www.monografias.com/trabajos95/redes-neuronales-artificiales-metodologia-desarrollo-y-aplicaciones/redes-neuronales-artificiales-metodologia-desarrollo-y-aplicaciones.shtml. [Último acceso: 21 Enero 2017].

[35] Jeff Hawkins, Sandra Blakesl, On Intelligence, Nueva York: St. Martin's Griffin, 2004.

[36] Jeff Hawkins, «Numenta,» [En línea]. Available: http://numenta.com. [Último acceso: 12 Febrero 2017].

[37] Numenta , «Hierarchical Temporal Memory including HTM Cortical Learning Algorithms,» 2011.

[38] Matthew Taylor, «GitHub-Numenta,» [En línea]. Available: https://github.com/numenta/nupic/wiki/Hierarchical-Temporal-Memory-Theory. [Último acceso: 15 Febrero 2017].

[39] «NuPic-SDR,» [En línea]. Available: https://github.com/numenta/nupic/wiki/Sparse-Distributed-Representations. [Último acceso: 12 Febrero 2017].

[40] «GitHub-Numenta-CLA,» [En línea]. Available: https://github.com/numenta/nupic/wiki/CLA-for-ML-AI-Researchers. [Último acceso: 15 Febrero 2017].

[41] «Nupic-encoders,» [En línea]. Available: https://github.com/numenta/nupic/wiki/Encoders. [Último acceso: 12 Febrero 2017].

[42] «NuPic-CLA classifier,» [En línea]. Available: https://github.com/numenta/nupic/wiki/CLA-Classifier. [Último acceso: 12 Febrero 2017].

[43] Chetan Surpur, Numenta , «Geospatial Coordinate Encoder».

[44] «Youtube - Geospatial encoder,» [En línea]. Available: https://www.youtube.com/watch?v=KxxHo-FtKRo. [Último acceso: 12 Febrero 2017].

[45] «Inbits,» [En línea]. Available: http://inbits.com/2014/08/implications-of-the-geospatial-encoder/. [Último acceso: 12 Febrero 2017].

[46] «Collective Intelligence Research Institute,» [En línea]. Available: http://cir.institute/swarm-collective-intelligence/. [Último acceso: 12 Febrero 2017].

[47] «IMO (International Martime Organization),» [En línea]. Available: http://www.imo.org/en/OurWork/safety/navigation/pages/ais.aspx. [Último acceso: 11 Febrero 2017].

[48] «Azimutmarine,» [En línea]. Available: http://www.azimutmarine.es/nautica/seguridad-ais. [Último acceso: 12 Febrero 2017].

[49] «AESA (Agencia Estatal de Seguridad Aérea),» [En línea]. Available: http://www.seguridadaerea.gob.es/lang_castellano/navegacion/programas/ads/sistema_ads.aspx. [Último acceso: 10 Febrero 2017].

[50] «ENAIRE-ADS-B,» [En línea]. Available: http://www.enaire.es/csee/Satellite/navegacion-aerea/es/Page/1043829981402//Vigilancia-dependiente-automatica-ADS.html. [Último acceso: 10 Febrero 2017].

[51] «Revista GQ- Neptune,» [En línea]. Available: http://www.revistagq.com/noticias/articulos/inteligencia-artificial-trafico-maritimo-simulador-barcos/24987. [Último acceso: 12 Febrero 2017].

[52] «Universidad Carlos III de Madrid-GIAA,» [En línea]. Available: http://portal.uc3m.es/portal/page/portal/actualidad_cientifica/noticias/maritime_surveillance. [Último acceso: 12 Febrero 2017].

[53] «NextGen-FAA (Federal Aviation Administration),» [En línea]. Available: https://www.faa.gov/nextgen/. [Último acceso: 10 Febrero 2017].

[54] «ASDE-X. FAA (Federal Aviation Administrarion),» [En línea]. Available: https://www.faa.gov/air_traffic/technology/asde-x/. [Último acceso: 12 Febrero 2017].

[55] «Roger-Wilko,» [En línea]. Available: http://www.roger-wilco.net/airport-information-network-ain-what-is-this/. [Último acceso: 12 Febrero 2017].

[56] «ATAD (Air Traffic Anomaly Detector),» [En línea]. Available: https://devpost.com/software/air-traffic-anomaly-detector. [Último acceso: 13 Febrero 2017].

[57] «HTM-MoClu,» [En línea]. Available: https://github.com/antidata/htm-moclu. [Último acceso: 12 Febrero 2017].

[58] Matthew Taylor, «HTM Open SourceNuPic - YouTube,» [En línea]. Available: https://www.youtube.com/watch?annotation_id=annotation_2695866579&feature=iv&src_vid=1fIpgXHXAZA&v=rN-57iBvcT4. [Último acceso: 17 Febrero 2017].

[59] «GitHub-Nupic-Install,» [En línea]. Available: https://github.com/numenta/nupic/wiki/Compiling-NuPIC-on-Ubuntu-14. [Último acceso: 17 Febrero 2017].

[60] «GitHub-geospatial,» [En línea]. Available: https://github.com/numenta/nupic.geospatial. [Último acceso: 15 Febrero 2017].

[61] «youtube-geospatial app,» [En línea]. Available:

https://www.youtube.com/watch?v=M4dD9wCQLkA. [Último acceso: 15 Febrero 2017].

[62] «Ais For Data,» [En línea]. Available: https://www.marinetraffic.com/en/ais-api-services/documentation/_:0504a2156e06e629ea85176e7fe3dc24. [Último acceso: 24 Febrero 2017].

[63] «Flightradar24,» [En línea]. Available: https://www.flightradar24.com. [Último acceso: 24 Febrero 2017].

[64] «AliExpress,» [En línea]. Available: https://www.aliexpress.com/item-img/Usb-2-0-Software-Radio-DVB-T-RTL2832U-R820T2-SDR-receptor-de-la-TV-Digital-producto/32635958974.html?spm=2114.10010408.1000017.2.r8ztZx#. [Último acceso: 24 Febrero 2017].

[65] «N8MDP,» [En línea]. Available: http://www.n8mdp.com/ads_b_setup.php. [Último acceso: 24 Febrero 2017].

[66] «GitHub-Runing-Swarms,» [En línea]. Available: https://github.com/numenta/nupic/wiki/Running-Swarms. [Último acceso: 17 Febrero 2017].

[67] Numenta Open Source, «Youtube- One Hot Gym tutorial,» [En línea]. Available: https://www.youtube.com/watch?v=S-0thrzOHTc. [Último acceso: 20 Febrero 2017].

[68] Cornell University, «Performance Measures for Machine Learning».

[69] «Revolvy,» [En línea]. Available: https://www.revolvy.com/main/index.php?s=Precision%20and%20recall&item_type=topic. [Último acceso: 24 Febrero 2017].

[70] «Developers.Google,» [En línea]. Available: https://developers.google.com/kml/articles/csvtokml. [Último acceso: 14 Febrero 2017].

[71] «DeveloperGoogle AltitudeModes,» [En línea]. Available: https://developers.google.com/kml/documentation/altitudemode?hl=es-419#relativetoseafloor. [Último acceso: 17 Febrero 2017].

[72] «Jupyter Notebook-Encoders,» [En línea]. Available: http://nbviewer.jupyter.org/github/numenta/nupic/blob/master/examples/NuPIC%20Walkthrough.ipynb. [Último acceso: 24 Febrero 2017].

[73] «osmocom,» [En línea]. Available: http://osmocom.org/projects/sdr/wiki/rtl-sdr. [Último acceso: 27 Febrero 2017].

[74] «rtlsdr,» [En línea]. Available: https://rtlsdr.org/. [Último acceso: 27 Febrero 2017].

[75] [En línea]. Available: http://superkuh.com/rtlsdr.html. [Último acceso: 24 Febrero 2017].

[76] «Zargotechnologies,» [En línea]. Available: http://www.zargotechnologies.com/pdf/RG174.pdf . [Último acceso: 14 Febrero 2017].

[77] «Coax Cable Loss / Antenna Gain Calculator,» [En línea]. Available: http://www.qsl.net/co8tw/Coax_Calculator.htm . [Último acceso: 1 Marzo 2017].

[78] «asd-b exchange,» [En línea]. Available: https://www.adsbexchange.com/forums/topic/beginners-2-cantenna-easy-diy-antenna-to-improve-rangeplane-count/. [Último acceso: 20 Febrero 2017].

[79] «4Nec2,» [En línea]. Available: http://www.qsl.net/4nec2/. [Último acceso: 4 Marzo 2017].

[80] Base Aérea de Morón. Ejercito del Aire, «Tutorial para la simunalición de antenas con 4nec2».

[81] «FlightRadar 24,» [En línea]. Available: https://www.flightradar24.com. [Último acceso: 10 Febrero 2017].

[82] «MarineTraffic,» [En línea]. Available: https://www.marinetraffic.com. [Último acceso: 10 Febrero 2017].

[83] «RedesZone,» [En línea]. Available: https://www.redeszone.net/2010/11/09/criptografia-algoritmos-de-autenticacion-hash/. [Último acceso: 20 Febrero 2017].

[84] «Una Aplicación de la Regla de Bayes en Ciencias de la Salud,» XXVI Simposio Internacional de Estadística 2016, 2016.

ANEXO I: SIGLAS Y ACRÓNIMOS

- ADS-B: *Automatic Dependet Survillance-Broadscast* (Vigilancia Dependiente Automática - Radiodifusión).
- AIS: *Automatic Identification System* (Sistema de Identificación Automática).
- ANN: *Artificial Neural Network* (Red Neuronal Artificial).
- API: *Application Programming Interface* (Interfaz de Programación de Aplicaciones).
- CLA: *Cortical Learning Algorithm*.
- COVAM: Centro de Operaciones y Vigilancia de Acción Marítima.
- CSV: *Comma Separated Values* (valores separados por comas).
- DST: Dispositivo de Separación de Tráfico.
- EVA: Escuadrón de Vigilancia Aérea.
- FAA: *Federal Aviation Administration* (Administración Federal de Aviación, de los Estados Unidos).
- HTM: *Hierarchical Temporal Memory*.
- IA : Inteligencia Artificial.
- ICAO: *International Civil Aviation Organization* (Organización de Aviación Civil Internacional).
- KML: *Keyhole Markup Language*.
- MACOM: Mando Aéreo de Combate.
- MMSI : *Maritime Mobile Service Identity* (Número de Identificación del Servicio Móvil Marítimo).
- NuPIC : *Numenta Platform for Intelligent Computing*.
- OMI: Organización Marítima Internacional.
- ROC: *Receiver Operating Characteristic*.
- SDR: *Software Defined Radio* (Radio definida por software).
- SDR: Sparse Distributed Representation (Representación distribuida dispersa).
- SP : *spatial pooler*.
- SQL: *Structured Query Language* (lenguaje de consulta estructurada).
- TP : *temporal pooler*.
- VHF: *Very High Frequency*.

ANEXO II: DESCRIPCIÓN DEL RECEPTOR SDR Y MODIFICACIÓN DE LA ANTENA

El tipo de dispositivo que se ha utilizado para recibir los datos ADS-B fue diseñado en un principio para recibir y decodificar la señal de la Televisión Digital Terrestre, pero algunos desarrolladores de Linux, como Eric Fry, la comunidad del proyecto Osmocon y sobre todo el finlandés Antti Palosaari, se dieron cuenta de que, usando controladores y software adecuados, se podía usar como un receptor SDR que cubre la gama de frecuencias entre 26 y 1700 MHz [73] y [74].

Básicamente consta de dos circuitos. Uno, gobernado por el *chip* R820T, que se encarga de la parte analógica de radiofrecuencia y otro, con el chip RTL283U de la empresa Realtek, que convierte la señal analógica en digital y la envía a través del puerto USB [75]. Es un receptor muy básico como se puede esperar por su bajo precio, pero es perfectamente útil para el propósito de este trabajo.

En cuanto a la mejora llevada a cabo, originalmente el dispositivo venía con una antena de 11 centímetros que se conectaba al receptor con un cable coaxial, RG174U, de 3 metros de longitud. Buscando las especificaciones [76] se ve que las pérdidas que tiene este cable a 1Ghz es de 118db/100m, luego para 3 metros, hay casi 3,5db de pérdidas. Usando una calculadora online [77] se puede observar que el cable coaxial produce aproximadamente un 55% de pérdidas en la señal recibida, lo cual reduce la cobertura y calidad de los datos recibidos a menos de la mitad. Por lo tanto, se debe reducir la longitud del cable coaxial.

Además, buscando cómo optimizar la recepción del SDR para ADS-B se encuentra esta página web [78] donde explican una mejora que se realiza en este sentido. Se decide adaptar la mejora planteada y así construir una antena denominada *ground plane*, una de las características de esta antena es que es omnidireccional y tiene un lóbulo uniforme (véanse Figura 0-1 y Figura 0-2).

Para ello, se ha utilizado un recipiente metálico circular, con un diámetro de 7 centímetros, alojando el receptor SDR (reduciendo así el cable coaxial a menos de 10 centímetros) y se ha hecho una perforación en el centro donde se coloca una varilla de cobre de 69 milímetros. Se aísla la varilla de dicha superficie y se conecta al vivo del cable coaxial mientras que la malla está en contacto con la superficie metálica circular. La medida de 69 mm corresponde a un cuarto de la longitud de onda que a la frecuencia de 1090Mhz es de 27,5 centímetros.

Por último se ha llevado a cabo el modelado de la antena con el programa 4NEC2 [79] utilizando el tutorial [80] realizado por miembros de la Base Aérea de Morón. Usando este programa se pueden obtener los lóbulos de radiación (en el plano horizontal, vertical y en 3D) en los cuales se observa la uniformidad del lóbulo y que la ganancia máxima en dBi son 0.9.

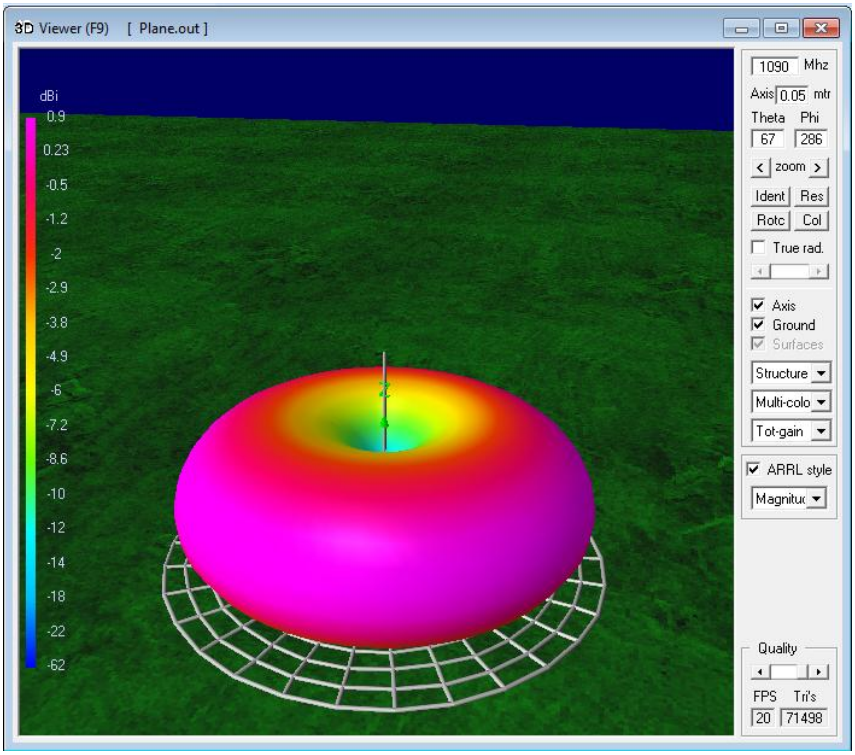

Figura 0-1 Diagrama de radiación 3D (fuente: propia).

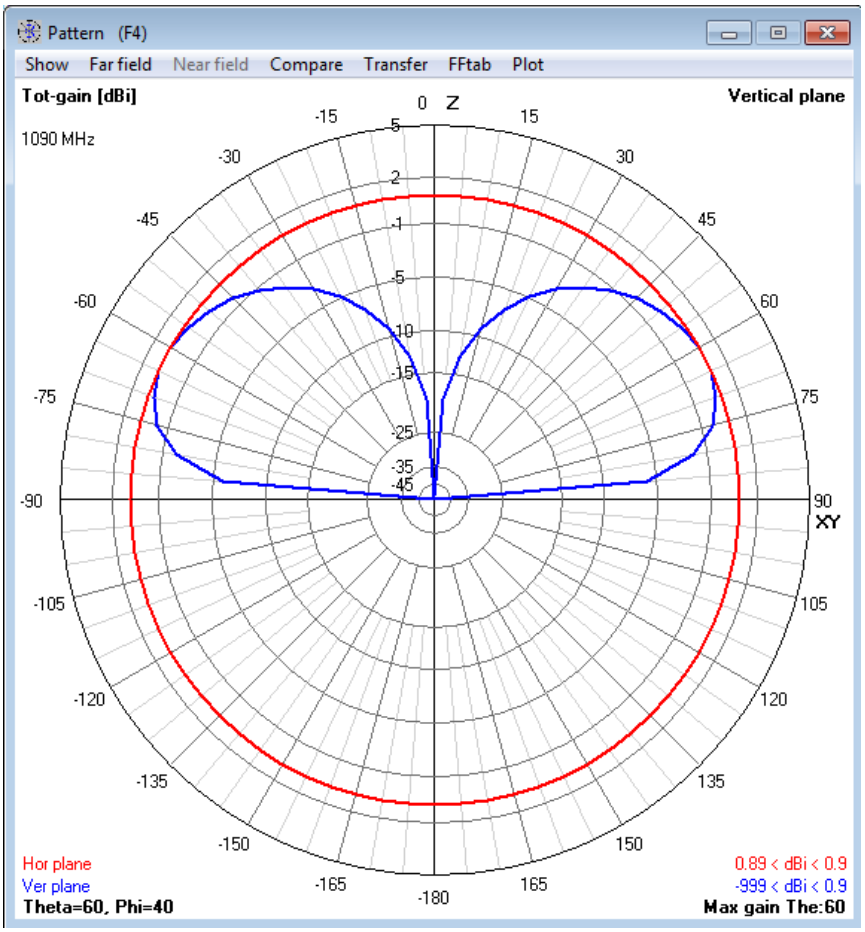

Figura 0-2 Diagrama de radiación vertical y horizontal (fuente: propia).

ANEXO III: MARITIME.PY

```
1. #!/usr/bin/env python
2. # -*- coding: utf-8 -*-
3.
4. import os
5. import sys
6. import datetime
7. import time
8. from optparse import OptionParser
9. from tools.preprocess_data import preprocess
10. from tools.anomaly_to_js_data import postprocess
11. from tools.anomaly_to_kml import anomalyrepresentation as KML
12. from model.geospatial_anomaly import runGeospatialAnomaly
13. from tools.postprocess_data import toAnalysis
14.
15. DEFAULT_OUTPUT_DIR = "output"
16. verbose = False
17. scriptDir = os.path.dirname(os.path.realpath(__file__))
18.
19. parser = OptionParser(
20.   usage="%prog <path/to/input/file> [options]\n\nRun NuPIC on specified "
21.         "location file, which should already be in the proper format "
22.         "(downloaded from the simulator)."
23. )
24. parser.add_option(
25.   "-m",
26.   "--manual-sequence",
27.   action="store_true",
28.   default=False,
29.   dest="manualSequence",
30.   help="Automatically breaks into sequences based upon time gaps."
31. )
32. parser.add_option(
33.   "-t",
34.   "--time-encoders",
35.   action="store_true",
36.   default=False,
37.   dest="useTimeEncoders",
```

```
38.   help="Adds time of day encoder to model params."
39. )
40. parser.add_option(
41.   "-v",
42.   "--verbose",
43.   action="store_true",
44.   default=False,
45.   dest="verbose",
46.   help="Print debugging statements."
47. )
48. parser.add_option(
49.   "-o",
50.   "--output-dir",
51.   default=DEFAULT_OUTPUT_DIR,
52.   dest="outputDir",
53.   help="Where to write the output file."
54. )
55. parser.add_option(
56.   "-s",
57.   "--scale",
58.   default=False,
59.   dest="scale",
60.   help="Meter resolution for Geospatial Coordinate Encoder (default 5m)."
61. )
62.
63. initime = datetime.datetime.now()
64. def runMaritime(inputPath, outputDir, useTimeEncoders, scale, autoSequence):
65.
66.   outputPath = os.path.abspath(outputDir)
67.   if not os.path.exists(outputPath):
68.     os.makedirs(outputPath)
69.
70.   preProcessedOutputPath = os.path.join(outputPath, "preprocessed_data.csv")
71.   if verbose: print "Pre-processing %s..." % inputPath
72.   preprocess(inputPath, preProcessedOutputPath, verbose=verbose)
73.
74.   anomalyOutputPath = os.path.join(outputPath, "anomaly_scores.csv")
75.   if verbose: print "Running NuPIC on %s..." % preProcessedOutputPath
76.   runGeospatialAnomaly(preProcessedOutputPath,
```

```
77.                         anomalyOutputPath,
78.                         scale=scale,
79.                         autoSequence=autoSequence,
80.                         useTimeEncoders=useTimeEncoders,
81.                         verbose=verbose)
82.
83.   fintime=datetime.datetime.now()
84.   print fintime - initime
85.
86.   visualizationOutputPath = os.path.join(scriptDir, "static/js/data.js")
87.   if verbose: print "Creating visualization at %s..." % visualizationOutputPath
88.   postprocess(anomalyOutputPath, visualizationOutputPath)
89.
90.   representationDataPath = os.path.join(outputPath, "anomaly_scores.csv")
91.   representationOutputPath = os.path.join(outputPath, "anomaly_representation.kml")
92.   if verbose: print "Creating visualization at %s..." % representationOutputPath
93.   KML(representationDataPath,representationOutputPath)
94.
95.   postProcessedOutputPath = os.path.join(outputPath, "data_to_analysis.csv")
96.   if verbose: print "Post-processing %s..." % anomalyOutputPath
97.   toAnalysis(anomalyOutputPath, postProcessedOutputPath, verbose=verbose)
98.
99. if __name__ == "__main__":
100.         (options, args) = parser.parse_args(sys.argv[1:])
101.         try:
102.           input_path = args.pop(0)
103.         except IndexError:
104.           parser.print_help(sys.stderr)
105.           sys.exit()
106.         verbose = options.verbose
107.         runMaritime(
108.           input_path,
109.           options.outputDir,
110.           options.useTimeEncoders,
111.           options.scale,
112.           not options.manualSequence)
```

ANEXO IV: PREPROCESS_DATA.PY

```
1. #!/usr/bin/env python
2. # -*- coding: utf-8 -*-
3. import csv
4. import datetime
5. import sys
6.
7. def preprocess(dataPath, outPath, verbose=False):
8.   with open(dataPath) as csvfile:
9.     reader = csv.reader(csvfile)
10.    writer = csv.writer(open(outPath, "wb"))
11.    lastTimestamp = None
12.    lastTimestampKept = None
13.    lastrow = None
14.    lastICAO= None
15.    numdat= 0
16.    numkeep= 0
17.    lastlat= 0
18.    lastlon = 0
19.
20.    reader.next()
21.    for row in reader:
22.      numdat += 1
23.      ICAO= row[0]
24.      lat = float(row[7])
25.      lon = float(row[8])
26.      timestamp = datetime.datetime.strptime(str(row[12]),"%Y-%m-%dT%H:%M:%S")
27.      keep = True
28.      if (ICAO != lastICAO):
29.    lastTimestamp = None
30.    lastTimestampKept = None
31.    lastlat= 0
32.        lastlon = 0
33.      if (lastTimestampKept and
34.          (timestamp - lastTimestampKept).total_seconds() < 5):  # Compara el tiempo del último dato
   valido con el del dato actual
35.        keep = False
```

```
36.       if abs(lat - lastlat) < 0.000833333 and abs(lon - lastlon ) < 0.000833333 :  # 0.000833333º
    son 90yds
37.     keep = False
38.       lastTimestamp = timestamp
39.       lastICAO = ICAO
40.       lastlat = lat
41.       lastlon = lon
42.       if keep:
43.     numkeep += 1
44.         lastrow = row
45.         lastTimestampKept = timestamp
46.         writer.writerow(row)
47.
48.       else:
49.         if verbose:
50.       print "Last row kept:\t{0}".format(lastrow)
51.           print "Discarding row:\t{0}".format(row)
52.   print "{0} de {1}".format(numkeep, numdat)
53.
54. if __name__ == "__main__":
55.   if len(sys.argv) < 3:
56.     print ("Usage: {0} "
57.            "/path/to/data.csv /path/to/outfile.csv").format(sys.argv[0])
58.     sys.exit(0)
59.
60.   dataPath = sys.argv[1]
61.   outPath = sys.argv[2]
62.   preprocess(dataPath, outPath)
```

ANEXO V: GEOSPATIAL_ANOMALY.PY

```
1. #!/usr/bin/env python
2.
3. import csv
4. import datetime
5. import sys
6. from nupic.frameworks.opf.modelfactory import ModelFactory
7. import model_params
8.
9. DEFAULT_DATA_PATH = "data/commute.csv"
10. DEFAULT_OUTPUT_PATH = "anomaly_scores.csv"
11. INTERVAL_THRESHOLD = 2040  # seconds = 34 min
12.
13. def addTimeEncoders(params):
14.   params["modelParams"]["sensorParams"]["encoders"]["timestamp_timeOfDay"] = {
15.     "fieldname": u"timestamp",
16.     "name": u"timestamp_timeOfDay",
17.     "timeOfDay": (51, 9.5),
18.     "type": "DateEncoder"
19.   }
20.   return params
21.
22. #Encoder RUMBO--------------
23. def addCourseEncoder(params):
24.    params["modelParams"]["sensorParams"]["encoders"]["course"]={
25.     "fieldname": "course",
26.     "maxval": 360,
27.     "minval": 0,
28.     "w":21,
29.     "n":200,
30.     "type": "ScalarEncoder",
31.         }
32.    return params
33.
34. #Encoder VELOCIDAD------------------- maxval= 333m/s; valmin= 0
35. def addSpeedEncoder(params):
36.    params["modelParams"]["sensorParams"]["encoders"]["speed"]={
```

```
    "fieldname": "speed",
    "maxval": 333,
    "minval": 0,
    "w":21,
    "n":50,
    "type": "ScalarEncoder",
        }
   return params

def setEncoderScale(params, scale):
  params["modelParams"]["sensorParams"]["encoders"]["vector"]["scale"] = \
    int(scale)
  return params

def createModel(useTimeEncoders, scale, verbose):
  params = model_params.MODEL_PARAMS
  params = addCourseEncoder(params)
  params = addSpeedEncoder(params)
  if useTimeEncoders:
    params = addTimeEncoders(params)
  if scale:
    params = setEncoderScale(params, scale)
  if verbose:
    print "Model parameters:"
    print params
  model = ModelFactory.create(params)
  model.enableInference({"predictedField": "vector"})
  return model

def runGeospatialAnomaly(dataPath, outputPath,
                         scale=False,
                         autoSequence=True,
                         useTimeEncoders=False,
                         verbose=False):

  model = createModel(useTimeEncoders, scale, verbose)

  with open (dataPath) as fin:
    reader = csv.reader(fin)
```

```
76.     csvWriter = csv.writer(open(outputPath,"wb"))
77.     csvWriter.writerow(["ICAO",
78.                 #"SQUAWK",
79.                 "FLIGHT",
80.                 "ALTITUDE",
81.                 "SPEED",
82.                 "HEADING",
83.                         "LATITUDE",
84.                         "LONGITUDE",
85.                 "TIMESTAMP",
86.                         "anomaly_score",
87.                         "new_sequence",
88.                 "Manual Anomaly"])
89.
90.     lastTimestamp = None
91.     lastICAO = None
92.     outputFormat = "%Y-%m-%dT%H:%M:%S"
93.     Nship = 1
94.     Ndata = 0
95.
96.     for _, record in enumerate(reader, start=1):
97.
98.       Ndata +=1
99.       ICAO = record[0]
100.              #sqwk = float(record[2])
101.              flight = str(record[3])
102.              altitude = str(record[4])
103.              speed = float(record[5]) * 46.3 / 90
104.              heading = int(record[6])
105.              latitude = float(record[7])
106.              longitude = float(record[8])
107.              #status = float(record[5])
108.              timestamp = datetime.datetime.strptime(str(record[12]),"%Y-%m-%dT%H:%M:%S")
109.
110.              if altitude == "grnd":
111.            altitude = 0
112.
113.              newSequence = False
114.              route= False
```

```
115.              manualAnomaly = False
116.
117.              # Handle the automatic sequence creation
118.              if autoSequence:
119.                if lastTimestamp and (
120.                   (timestamp - lastTimestamp).total_seconds() > INTERVAL_THRESHOLD):
121.              print "------New Route"
122.                   newSequence = True
123.              route = True
124.              else:
125.                if ICAO != lastICAO:
126.                   newSequence = True
127.
128.            else:
129.               if lastTimestamp and (
130.                   (timestamp - lastTimestamp).total_seconds() > INTERVAL_THRESHOLD):
131.                       if verbose :
132.                print "------New Route"
133.                newSequence = True
134.                route = True
135.
136.              if Ndata%500 == 0:
137.            print "-----------------------------------------------------Dato n:{0}.".format(Ndata)
138.
139.            #Manual Anomalies
140.              if ICAO == "999":
141.            manualAnomaly = True
142.
143.              lastTimestamp = timestamp
144.              lastICAO = ICAO
145.
146.              if newSequence:
147.
148.                if verbose:
149.                 if not route:
150.              print "\n          New airplane {0} - ICAO: {1}. -FLIGHT: ".format(Nship, ICAO, flight)
151.                model.resetSequenceStates()
152.            Nship +=1
153.              modelInput = {
```

```
154.              "vector": (speed, longitude, latitude, altitude)
155.            }
156.            modelInput["course"] = heading
157.            modelInput["speed"] = speed
158.            if useTimeEncoders:
159.              modelInput["timestamp"] = timestamp
160.
161.            result = model.run(modelInput)
162.
163.            anomalyScore = result.inferences["anomalyScore"]
164.            speed = speed * 90 / 46.3
165.            manualAnomaly ="https://es.flightaware.com/live/flight/{0}".format(flight)
166.            csvWriter.writerow([ICAO,
167.                        #sqwk,
168.                        flight,
169.                        altitude,
170.                    speed,
171.                    heading,
172.                    latitude,
173.                                longitude,
174.                    timestamp.strftime(outputFormat),
175.                                anomalyScore,
176.                                1 if newSequence else 0,
177.                    manualAnomaly])
178.            if verbose:
179.              print "[{0}] - Anomaly score: {1}.".format(timestamp, anomalyScore)
180.
181.        print "Anomaly scores have been written to {0}".format(outputPath)
182.
183.      if __name__ == "__main__":
184.        dataPath = DEFAULT_DATA_PATH
185.        outputPath = DEFAULT_OUTPUT_PATH
186.
187.        if len(sys.argv) > 1:
188.          dataPath = sys.argv[1]
189.
190.        if len(sys.argv) > 2:
191.          outputPath = sys.argv[2]
192.        runGeospatialAnomaly(dataPath, outputPath)
```

ANEXO VI: ANOMALY_TO_KML.PY

```
1.  #!/usr/bin/env python
2.
3.  import csv
4.  import xml.dom.minidom
5.  import sys
6.
7.  def extractCoordinates(row):
8.    # This extracts the coordinates from a row and returns it as a list. This requires knowing
9.    # ahead of time what the columns are that hold the address information.
10.   return '%s,%s,%s' % (row[6], row[5],row[2])
11.
12. def createPlacemark(kmlDoc, row, order):
13.   # This creates a <Placemark> element for a row of data.
14.   # A row is a dict.
15.   placemarkElement = kmlDoc.createElement('Placemark')
16.   extElement = kmlDoc.createElement('ExtendedData')
17.   placemarkElement.appendChild(extElement)
18.
19.   # Loop through the columns and create a <Data> element for every field that has a value.
20.   for i in range(0, len(order)):
21.       dataElement = kmlDoc.createElement('Data')
22.       dataElement.setAttribute('name', order[i])
23.       valueElement = kmlDoc.createElement('value')
24.       dataElement.appendChild(valueElement)
25.       valueText = kmlDoc.createTextNode(row[i])
26.       valueElement.appendChild(valueText)
27.       extElement.appendChild(dataElement)
28.
29.   if float(row[8]) <= 0.25:
30.     styleElement = kmlDoc.createElement('styleUrl')
31.     styleElement.appendChild(kmlDoc.createTextNode('#green'))
32.     placemarkElement.appendChild(styleElement)
33.   elif float(row[8]) <= 0.65:
34.     styleElement = kmlDoc.createElement('styleUrl')
35.     styleElement.appendChild(kmlDoc.createTextNode('#yellow'))
36.     placemarkElement.appendChild(styleElement)
37.   else:
```

```
38.     styleElement = kmlDoc.createElement('styleUrl')
39.     styleElement.appendChild(kmlDoc.createTextNode('#red'))
40.     placemarkElement.appendChild(styleElement)
41.   pointElement = kmlDoc.createElement('Point')
42.   placemarkElement.appendChild(pointElement)
43.   coordinates = extractCoordinates(row)
44.   altElement = kmlDoc.createElement('gx:altitudeMode')
45.   altElement.appendChild(kmlDoc.createTextNode('relativeToSeaFloor'))
46.   pointElement.appendChild(altElement)
47.   coorElement = kmlDoc.createElement('coordinates')
48.   coorElement.appendChild(kmlDoc.createTextNode(coordinates))
49.   pointElement.appendChild(coorElement)
50.   return placemarkElement
51.
52.
53. def createKML(csvReader, csvreader1, fileName, order):
54.   # This constructs the KML document from the CSV file.
55.   kmlDoc = xml.dom.minidom.Document()
56.
57.   kmlElement = kmlDoc.createElementNS('http://www.google.com/kml/ext/2.2', 'kml')
58.   kmlElement.setAttribute('xmlns:gx', 'http://www.google.com/kml/ext/2.2')
59.   kmlElement = kmlDoc.appendChild(kmlElement)
60.   documentElement = kmlDoc.createElement('Document')
61.   styleElement = kmlDoc.createElement('Style')
62.   styleElement.setAttribute('id', 'green')
63.   iconstyleElement = kmlDoc.createElement('IconStyle')
64.   scaleElement = kmlDoc.createElement('scale')
65.   scaleText = kmlDoc.createTextNode('0.5')
66.   scaleElement.appendChild(scaleText)
67.   iconstyleElement.appendChild(scaleElement)
68.   iconElement = kmlDoc.createElement('Icon')
69.   hrefElement = kmlDoc.createElement('href')
70.   hrefText = kmlDoc.createTextNode('http://maps.google.com/mapfiles/kml/paddle/grn-blank.png')
71.   hrefElement.appendChild(hrefText)
72.   iconElement.appendChild(hrefElement)
73.   iconstyleElement.appendChild(iconElement)
74.   styleElement.appendChild(iconstyleElement)
75.   documentElement.appendChild(styleElement)
76.   styleElement = kmlDoc.createElement('Style')
```

```
77.   styleElement.setAttribute('id', 'yellow')
78.   iconstyleElement = kmlDoc.createElement('IconStyle')
79.   scaleElement = kmlDoc.createElement('scale')
80.   scaleText = kmlDoc.createTextNode('0.6')
81.   scaleElement.appendChild(scaleText)
82.   iconstyleElement.appendChild(scaleElement)
83.   iconElement = kmlDoc.createElement('Icon')
84.   hrefElement = kmlDoc.createElement('href')
85.   hrefText = kmlDoc.createTextNode('http://maps.google.com/mapfiles/kml/paddle/ylw-blank.png')
86.   hrefElement.appendChild(hrefText)
87.   iconElement.appendChild(hrefElement)
88.   iconstyleElement.appendChild(iconElement)
89.   styleElement.appendChild(iconstyleElement)
90.   documentElement.appendChild(styleElement)
91.   styleElement = kmlDoc.createElement('Style')
92.   styleElement.setAttribute('id', 'red')
93.   iconstyleElement = kmlDoc.createElement('IconStyle')
94.   scaleElement = kmlDoc.createElement('scale')
95.   scaleText = kmlDoc.createTextNode('0.8')
96.   scaleElement.appendChild(scaleText)
97.   iconstyleElement.appendChild(scaleElement)
98.   iconElement = kmlDoc.createElement('Icon')
99.   hrefElement = kmlDoc.createElement('href')
100.          hrefText = kmlDoc.createTextNode('http://maps.google.com/mapfiles/kml/paddle/red-
    blank.png')
101.          hrefElement.appendChild(hrefText)
102.          iconElement.appendChild(hrefElement)
103.          iconstyleElement.appendChild(iconElement)
104.          styleElement.appendChild(iconstyleElement)
105.          documentElement.appendChild(styleElement)
106.          documentElement = kmlElement.appendChild(documentElement)
107.
108.          # Skip the header line.
109.          csvReader.next()
110.          usedtracks = ["0"]
111.          for row in csvReader:
112.            trackName = str(row['ICAO'])
113.            check = 0
114.            for track in usedtracks:
```

```
115.              if track == trackName:
116.                 check += 1
117.            if check == 0:
118.                folderElement = kmlDoc.createElement('Folder')
119.            nameElement = kmlDoc.createElement('name')
120.            nameText = kmlDoc.createTextNode(trackName)
121.            nameElement.appendChild(nameText)
122.            folderElement.appendChild(nameElement)
123.                csvreader1.remove(csvreader1[0])
124.            for item in csvreader1:
125.              if item[0] == trackName:
126.                placemarkElement = createPlacemark(kmlDoc, item, order)
127.                folderElement.appendChild(placemarkElement)
128.                usedtracks.append(trackName)
129.            documentElement.appendChild(folderElement)
130.
131.          kmlFile = open(fileName, 'w')
132.          kmlFile.write(kmlDoc.toprettyxml('  ', newl = '\n', encoding = 'utf-8'))
133.
134.        def anomalyrepresentation(dataPath, outPath):
135.          # This reader opens up 'anomaly_scores.csv'.
136.          # It creates a KML file called 'anomaly_representation.kml'.
137.
138.          # If an argument was passed to the script, it splits the argument on a comma
139.          # and uses the resulting list to specify an order for when columns get added.
140.          # Otherwise, it defaults to the order used in the sample.
141.
142.          order = ['ICAO','Name Flight','altitude (pies)','speed
    (kts)','heading','latitude','longitude','timestamp','anomaly_score','new_sequence','web']
143.          csvreader = csv.DictReader(open(dataPath),order)
144.          csvreader1 = []
145.          for row in csvreader:
146.            line = []
147.            for key in order:
148.              line.append(str(row[key]))
149.            csvreader1.append(line)
150.          csvreader = csv.DictReader(open(dataPath),order)
151.          kml = createKML(csvreader, csvreader1, outPath, order)
152.
```

```
153.      if __name__ == "__main__":
154.        if len(sys.argv) < 3:
155.          print ("Usage: {0} "
156.                 "/path/to/data.csv /path/to/outfile.csv").format(sys.argv[0])
157.          sys.exit(0)
158.
159.        dataPath = sys.argv[1]
160.        outPath = sys.argv[2]
161.        anomalyrepresentation(dataPath, outPath)
```

ANEXO VII: POSTPROCESS_DATA.PY

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.
4.  import csv
5.  import datetime
6.  import sys
7.
8.  def toAnalysis(dataPath, outPath, verbose=False):
9.    with open(dataPath) as csvfile:
10.     reader = csv.reader(csvfile)
11.     writer = csv.writer(open(outPath, "wb"))
12.     writer.writerow (["MMSI",
13.                       "anomaly_score",
14.                "Manual Anomaly"])
15.     lastrow = None
16.     lastmmsi= None
17.     numdat= 0
18.     numkeep= 0
19.     Nremove = 0
20.     toAnalysis = False
21.
22.     reader.next()
23.     for row in reader:
24.       numdat += 1
25.       Nremove -= 1
26.       mmsi= row[0]
27.       anomaly = float(row[7])
28.       newSequence = int (row[8])
29.       keep = True
30.       if numdat > 10000:      # A partir del dato 10000 realiza limpieza de los falsos positivos
31.        if (newSequence == 1):   #Si hay nueva sequencia y las 1 siguientes
32.          Nremove = 1
33.          keep = False
34.       if numdat > 11000:
35.         toAnalysis = True
36.       if keep and Nremove < 0:
```

```
37.         numkeep += 1
38.         lastrow = row
39.         writer.writerow([mmsi,
40.                          anomaly,
41.              1 if toAnalysis else 0])
42.
43.   print "{0} de {1}".format(numkeep, numdat)
44. if __name__ == "__main__":
45.   if len(sys.argv) < 3:
46.     print ("Usage: {0} "
47.            "/path/to/data.csv /path/to/outfile.csv").format(sys.argv[0])
48.     sys.exit(0)
49.
50.   dataPath = sys.argv[1]
51.   outPath = sys.argv[2]
52.   toAnalysis(dataPath, outPath)
```

ANEXO VIII: MODEL_PARAMS.PY

```
MODEL_PARAMS = {
    # Type of model that the rest of these parameters apply to.
    'model': "CLA",

    # Version that specifies the format of the config.
    'version': 1,

    # Intermediate variables used to compute fields in modelParams and also
    # referenced from the control section.
    'predictAheadTime': None,

    # Model parameter dictionary.
    'modelParams': {
        # The type of inference that this model will perform
        'inferenceType': 'TemporalAnomaly',

        'sensorParams': {
            # Sensor diagnostic output verbosity control;
            # if > 0: sensor region will print out on screen what it's sensing
            # at each step 0: silent; >=1: some info; >=2: more info;
            # >=3: even more info (see compute() in py/regions/RecordSensor.py)
            'verbosity' : 0,

            # Example:
            #     dsEncoderSchema = [
            #       DeferredDictLookup('__field_name_encoder'),
            #     ],
            #
            # (value generated from DS_ENCODER_SCHEMA)
            'encoders': {
                u'vector':    {
                    'fieldname': u'vector',
                    'n': 2048,
                    'w': 51,
                    'scale': 5,
                    'timestep': 10,
```

```
                    'name': u'vector',
                    'type': 'GeospatialCoordinateEncoder'
                },
            },

            # A dictionary specifying the period for automatically-generated
            # resets from a RecordSensor;
            #
            # None = disable automatically-generated resets (also disabled if
            # all of the specified values evaluate to 0).
            # Valid keys is the desired combination of the following:
            #  days, hours, minutes, seconds, milliseconds, microseconds, weeks
            #
            # Example for 1.5 days: sensorAutoReset = dict(days=1,hours=12),
            'sensorAutoReset' : None,
        },

        'spEnable': True,

        'spParams': {
            # SP diagnostic output verbosity control;
            # 0: silent; >=1: some info; >=2: more info;
            'spVerbosity' : 0,

            # Spatial Pooler implementation selector.
            # Options: 'py', 'cpp' (speed optimized, new)
            'spatialImp' : 'cpp',

            'globalInhibition': 1,

            # Number of columns in the SP (must be same as in TP)
            'columnCount': 2048,

            'inputWidth': 0,

            # SP inhibition control (absolute value);
            # Maximum number of active columns in the SP region's output (when
            # there are more, the weaker ones are suppressed)
            'numActiveColumnsPerInhArea': 40,
```

```
76.
77.             'seed': 1956,
78.
79.             # potentialPct
80.             # What percent of the columns's receptive field is available
81.             # for potential synapses.
82.             'potentialPct': 0.8,
83.
84.             # The default connected threshold. Any synapse whose
85.             # permanence value is above the connected threshold is
86.             # a "connected synapse", meaning it can contribute to the
87.             # cell's firing. Typical value is 0.10.
88.             'synPermConnected': 0.1,
89.
90.             'synPermActiveInc': 0.0001,
91.
92.             'synPermInactiveDec': 0.0005,
93.
94.             'maxBoost': 1.0,
95.         },
96.
97.         # Controls whether TP is enabled or disabled;
98.         # TP is necessary for making temporal predictions, such as predicting
99.         # the next inputs.  Without TP, the model is only capable of
100.                # reconstructing missing sensor inputs (via SP).
101.                'tpEnable' : True,
102.
103.                'tpParams': {
104.                    # TP diagnostic output verbosity control;
105.                    # 0: silent; [1..6]: increasing levels of verbosity
106.                    # (see verbosity in nta/trunk/py/nupic/research/TP.py and TP10X*.py)
107.                    'verbosity': 0,
108.
109.                    # Number of cell columns in the cortical region (same number for
110.                    # SP and TP)
111.                    # (see also tpNCellsPerCol)
112.                    'columnCount': 2048,
113.
114.                    # The number of cells (i.e., states), allocated per column.
```

```
115.                 'cellsPerColumn': 32,
116.
117.                 'inputWidth': 2048,
118.
119.                 'seed': 1960,
120.
121.                 # Temporal Pooler implementation selector (see _getTPClass in
122.                 # CLARegion.py).
123.                 'temporalImp': 'cpp',
124.
125.                 # New Synapse formation count
126.                 # NOTE: If None, use spNumActivePerInhArea
127.                 #
128.                 # TODO: need better explanation
129.                 'newSynapseCount': 20,
130.
131.                 # Maximum number of synapses per segment
132.                 #  > 0 for fixed-size CLA
133.                 # -1 for non-fixed-size CLA
134.                 #
135.                 # TODO: for Ron: once the appropriate value is placed in TP
136.                 # constructor, see if we should eliminate this parameter from
137.                 # description.py.
138.                 'maxSynapsesPerSegment': 32,
139.
140.                 # Maximum number of segments per cell
141.                 #  > 0 for fixed-size CLA
142.                 # -1 for non-fixed-size CLA
143.                 #
144.                 # TODO: for Ron: once the appropriate value is placed in TP
145.                 # constructor, see if we should eliminate this parameter from
146.                 # description.py.
147.                 'maxSegmentsPerCell': 128,
148.
149.                 # Initial Permanence
150.                 # TODO: need better explanation
151.                 'initialPerm': 0.21,
152.
153.                 # Connected Permanence
```

```
154.                 'connectedPerm': 0.5,
155.
156.                 # Permanence Increment
157.                 'permanenceInc': 0.1,
158.
159.                 # Permanence Decrement
160.                 # If set to None, will automatically default to tpPermanenceInc
161.                 # value.
162.                 'permanenceDec' : 0.1,
163.
164.                 'globalDecay': 0.0,
165.
166.                 'maxAge': 0,
167.
168.                 # Minimum number of active synapses for a segment to be considered
169.                 # during search for the best-matching segments.
170.                 # None=use default
171.                 # Replaces: tpMinThreshold
172.                 'minThreshold': 3,
173.
174.                 # Segment activation threshold.
175.                 # A segment is active if it has >= tpSegmentActivationThreshold
176.                 # connected synapses that are active due to infActiveState
177.                 # None=use default
178.                 # Replaces: tpActivationThreshold
179.                 'activationThreshold': 6,
180.
181.                 'outputType': 'normal',
182.
183.                 # "Pay Attention Mode" length. This tells the TP how many new
184.                 # elements to append to the end of a learned sequence at a time.
185.                 # Smaller values are better for datasets with short sequences,
186.                 # higher values are better for datasets with long sequences.
187.                 'pamLength': 3,
188.             },
189.
190.             # Don't create the classifier since we don't need predictions.
191.             'clEnable': False,
192.             'clParams': None,
```

```
193.
194.                 'anomalyParams': {  u'anomalyCacheRecords': None,
195.             u'autoDetectThreshold': None,
196.             u'autoDetectWaitRecords': 2184},
197.
198.                 'trainSPNetOnlyIfRequested': False,
199.             },
200.         }
```

ANEXO IX: FICHERO SALIDA DUMP1090 SIN TRATAR

```
#[H#[2JHex    Mode  Sqwk  Flight   Alt   Spd  Hdg   Lat     Long   Sig  Msgs   Ti|
--------------------------------------------------------------------------------
341646  S                    384  250               23     1    1
4CA8D4  S              36650  424  311   39.707   -1.469   34    10    0
4CA891  S                                 23     2    1
4CA34B  S    1277      31000  383  276   39.748   -0.336   70    35    0
#[H#[2JHex    Mode  Sqwk  Flight   Alt   Spd  Hdg   Lat     Long   Sig  Msgs   Ti|
--------------------------------------------------------------------------------
341646  S                    384  250               23     1    1
4CA8D4  S              36675  424  311   39.707   -1.469   36    11    0
4CA891  S                                 23     2    1
4CA34B  S    1277      31000  383  276   39.748   -0.336   78    36    0
#[H#[2JHex    Mode  Sqwk  Flight   Alt   Spd  Hdg   Lat     Long   Sig  Msgs   Ti/
--------------------------------------------------------------------------------
341646  S                    384  250               23     1    2
4CA8D4  S              36675  424  311   39.707   -1.469   37    12    0
4CA891  S                                 23     2    2
4CA34B  S    1277      31000  383  276   39.748   -0.336   78    38    0
#[H#[2JHex    Mode  Sqwk  Flight   Alt   Spd  Hdg   Lat     Long   Sig  Msgs   Ti/
--------------------------------------------------------------------------------
341646  S                    384  250               23     1    2
4CA8D4  S              36675  424  311   39.707   -1.469   37    12    0
4CA891  S                                 23     2    2
4CA34B  S    1277      31000  383  276   39.748   -0.336   78    38    0
#[H#[2JHex    Mode  Sqwk  Flight   Alt   Spd  Hdg   Lat     Long   Sig  Msgs   Ti/
--------------------------------------------------------------------------------
341646  S                    384  250               23     1    2
4CA8D4  S              36675  424  311   39.707   -1.469   37    12    0
4CA891  S                                 23     2    2
4CA34B  S    1277      31000  383  276   39.749   -0.339   83    39    0
#[H#[2JHex    Mode  Sqwk  Flight   Alt   Spd  Hdg   Lat     Long   Sig  Msgs   Ti/
--------------------------------------------------------------------------------
```

ANEXO X: FICHERO SALIDA DUMP1090 TRATADO

3C496D,S,1033,,40000,456,034,42.115,-8.351,72,10,0,2017-02-10T14:15:18
3C496D,S,1033,,40000,456,034,42.169,-8.301,54,20,1,2017-02-10T14:15:51
3C496D,S,1033,BER994M,40000,455,035,42.188,-8.283,75,30,0,2017-02-10T14:16:00
3C496D,S,1033,BER994M,40000,455,035,42.208,-8.265,79,37,0,2017-02-10T14:16:13
3C496D,S,1033,BER994M,40000,455,034,42.248,-8.228,63,43,1,2017-02-10T14:16:34
3C49C6,S,4530,,34975,439,219,42.296,-8.480,83,23,1,2017-02-10T10:46:16
3C49C6,S,4530,,35000,439,219,42.281,-8.496,80,30,0,2017-02-10T10:46:26
3C49C6,S,4530,,35000,440,219,42.275,-8.503,100,43,0,2017-02-10T10:46:33
3C49C6,S,4530,BER353C,35000,440,219,42.260,-8.519,105,57,0,2017-02-10T10:46:39
3C49C6,S,4530,BER353C,35000,441,219,42.247,-8.532,111,68,0,2017-02-10T10:46:46
3C49C6,S,4530,BER353C,35000,441,219,42.239,-8.541,71,85,0,2017-02-10T10:46:52
3C49C6,S,4530,BER353C,35000,442,219,42.229,-8.552,105,93,0,2017-02-10T10:46:59
3C49C6,S,4530,BER353C,35000,442,219,42.213,-8.568,1231,16,0,2017-02-10T10:47:06
3C49C6,S,4530,BER353C,35000,442,219,42.212,-8.570,871,24,0,2017-02-10T10:47:12
3C49C6,S,4530,BER353C,35000,442,219,42.194,-8.589,721,36,0,2017-02-10T10:47:19
3C49C6,S,4530,BER353C,35000,442,219,42.186,-8.598,1111,51,0,2017-02-10T10:47:25
3C49C6,S,4530,BER353C,35000,442,219,42.180,-8.603,741,61,0,2017-02-10T10:47:32
3C49C6,S,4530,BER353C,35025,442,214,42.057,-8.718,451,77,0,2017-02-10T10:48:41
3C49E8,S,2524,,33000,437,219,42.289,-8.488,68,18,0,2017-02-10T09:39:32
3C49E8,S,2524,,33000,439,219,42.274,-8.503,77,22,0,2017-02-10T09:39:41
3C49E8,S,2524,,33000,440,219,42.265,-8.513,86,31,0,2017-02-10T09:39:50
3C49E8,S,2524,,33000,442,219,42.220,-8.561,88,55,0,2017-02-10T09:40:12
3C49E8,S,2524,CFG2LV,33000,442,219,42.206,-8.575,92,71,0,2017-02-10T09:40:20
3C49E8,S,2524,CFG2LV,33000,442,219,42.206,-8.575,92,71,0,2017-02-10T09:40:25
3C49E8,S,2524,CFG2LV,33000,441,215,42.193,-8.590,79,78,0,2017-02-10T09:40:33
3C56EF,S,,EWG9T,35000,439,211,41.467,-7.459,44,12,0,2017-02-11T14:11:03
3C56EF,S,,EWG9T,35000,439,211,41.277,-7.610,33,16,0,2017-02-11T14:12:52
3C56EF,S,,EWG9T,35025,439,211,41.060,-7.781,33,20,0,2017-02-11T14:14:56
3C5CB2,S,4526,,39050,442,219,42.261,-8.517,114,36,0,2017-02-10T15:45:34
3C5CB2,S,4526,GMI3356,39050,442,219,42.247,-8.532,124,42,0,2017-02-10T15:45:40
3C5CB2,S,4526,GMI3356,39050,442,219,42.238,-8.542,125,48,0,2017-02-10T15:45:45
3C5CB2,S,4526,GMI3356,39050,443,219,42.221,-8.560,82,61,0,2017-02-10T15:45:56

ANEXO XI: EXTRACTO FICHERO ANOMALY_SCORE.CSV

ICAO;FLIGHT;ALTITUDE;SPEED;HEADING;LATITUDE;LONGITUDE;TIMESTAMP;anomaly_score;new_sequence

3C496D;;40000;456;34;42,169;-8,301;2017-02-10T14:15:51;1;1

3C496D;BER994M;40000;455;35;42,188;-8,283;2017-02-10T14:16:00;1;0

3C496D;BER994M;40000;455;35;42,208;-8,265;2017-02-10T14:16:13;1;0

3C496D;BER994M;40000;455;34;42,248;-8,228;2017-02-10T14:16:34;0,975;0

3C49C6;;34975;439;219;42,296;-8,48;2017-02-10T10:46:16;1;1

3C49C6;;35000;439;219;42,281;-8,496;2017-02-10T10:46:26;1;0

3C49C6;;35000;440;219;42,275;-8,503;2017-02-10T10:46:33;1;0

3C49C6;BER353C;35000;440;219;42,26;-8,519;2017-02-10T10:46:39;0,775;0

3C49C6;BER353C;35000;441;219;42,247;-8,532;2017-02-10T10:46:46;0,875;0

3C49C6;BER353C;35000;441;219;42,239;-8,541;2017-02-10T10:46:52;0,825;0

3C49C6;BER353C;35000;442;219;42,229;-8,552;2017-02-10T10:46:59;0,675;0

3C49C6;BER353C;35000;442;219;42,213;-8,568;2017-02-10T10:47:06;0,7;0

3C49C6;BER353C;35000;442;219;42,212;-8,57;2017-02-10T10:47:12;0,6;0

3C49C6;BER353C;35000;442;219;42,194;-8,589;2017-02-10T10:47:19;0,65;0

3C49C6;BER353C;35000;442;219;42,186;-8,598;2017-02-10T10:47:25;0,625;0

3C49C6;BER353C;35000;442;219;42,18;-8,603;2017-02-10T10:47:32;0,25;0

3C49C6;BER353C;35025;442;214;42,057;-8,718;2017-02-10T10:48:41;0,525;0

3C49E8;;33000;437;219;42,289;-8,488;2017-02-10T09:39:32;1;1

3C49E8;;33000;439;219;42,274;-8,503;2017-02-10T09:39:41;0,525;0

3C49E8;;33000;440;219;42,265;-8,513;2017-02-10T09:39:50;0,45;0

3C49E8;;33000;442;219;42,22;-8,561;2017-02-10T09:40:12;0,625;0

3C49E8;CFG2LV;33000;442;219;42,206;-8,575;2017-02-10T09:40:20;0,375;0

3C49E8;CFG2LV;33000;441;215;42,193;-8,59;2017-02-10T09:40:33;0,225;0

3C56EF;EWG9T;35000;439;211;41,467;-7,459;2017-02-11T14:11:03;1;1

3C56EF;EWG9T;35000;439;211;41,277;-7,61;2017-02-11T14:12:52;1;0

3C56EF;EWG9T;35025;439;211;41,06;-7,781;2017-02-11T14:14:56;0,95;0

3C5CB2;;39050;442;219;42,261;-8,517;2017-02-10T15:45:34;1;1

3C5CB2;GMI3356;39050;442;219;42,247;-8,532;2017-02-10T15:45:40;1;0

3C5CB2;GMI3356;39050;442;219;42,238;-8,542;2017-02-10T15:45:45;1;0

3C5CB2;GMI3356;39050;443;219;42,221;-8,56;2017-02-10T15:45:56;0,8;0

3C5CB2;GMI3356;39050;442;219;42,193;-8,59;2017-02-10T15:46:14;0,775;0

ANEXO XII: EXTRACTO FICHERO ANOMALY_REPRESENTATION.KML

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <kml xmlns:gx="http://www.google.com/kml/ext/2.2">
3.    <Document>
4.      <Style id="green">
5.        <IconStyle>
6.          <scale>0.5</scale>
7.          <Icon>
8.            <href>http://maps.google.com/mapfiles/kml/paddle/grn-blank.png</href>
9.          </Icon>
10.       </IconStyle>
11.     </Style>
12.     <Style id="yellow">
13.       <IconStyle>
14.         <scale>0.6</scale>
15.         <Icon>
16.           <href>http://maps.google.com/mapfiles/kml/paddle/ylw-blank.png</href>
17.         </Icon>
18.       </IconStyle>
19.     </Style>
20.     <Style id="red">
21.       <IconStyle>
22.         <scale>0.8</scale>
23.         <Icon>
24.           <href>http://maps.google.com/mapfiles/kml/paddle/red-blank.png</href>
25.         </Icon>
26.       </IconStyle>
27.     </Style>
28.     <Folder>
29.       <name>3C496D</name>
30.       <Placemark>
31.         <ExtendedData>
32.           <Data name="ICAO">
33.             <value>3C496D</value>
34.           </Data>
35.           <Data name="Name Flight">
```

```
36.             <value></value>
37.           </Data>
38.           <Data name="altitude (pies)">
39.             <value>40000</value>
40.           </Data>
41.           <Data name="speed (kts)">
42.             <value>456.0</value>
43.           </Data>
44.           <Data name="heading">
45.             <value>34</value>
46.           </Data>
47.           <Data name="latitude">
48.             <value>42.169</value>
49.           </Data>
50.           <Data name="longitude">
51.             <value>-8.301</value>
52.           </Data>
53.           <Data name="timestamp">
54.             <value>2017-02-10T14:15:51</value>
55.           </Data>
56.           <Data name="anomaly_score">
57.             <value>1.0</value>
58.           </Data>
59.           <Data name="new_sequence">
60.             <value>1</value>
61.           </Data>
62.           <Data name="web">
63.             <value>https://es.flightaware.com/live/flight/</value>
64.           </Data>
65.         </ExtendedData>
66.         <styleUrl>#red</styleUrl>
67.         <Point>
68.           <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
69.           <coordinates>-8.301,42.169,40000</coordinates>
70.         </Point>
71.       </Placemark>
```