Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de un sistema de inteligencia artificial para la supervisión y detección de anomalías en rutas marítimas

Grado en Ingeniería Mecánica

ALUMNO: Carlos Arenas Pérez-Seoane

DIRECTORES: Miguel Rodelgo Lacruz

José María Núñez Ortuño

CURSO ACADÉMICO: 2015-2016

Universidad de Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de un sistema de inteligencia artificial para la supervisión y detección de anomalías en rutas marítimas

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

El presente trabajo surge de la necesidad de analizar y supervisar la información del tráfico marítimo mediante un sistema de Inteligencia Artificial. Para ello se ha desarrollado una aplicación que recibe un conjunto de datos geográficos de los buques que navegan por una zona determinada a través de mensajes AIS y, mediante técnicas de inteligencia artificial, genera un modelo que representa los patrones de movimiento habituales y detecta aquellos que se apartan del comportamiento normal (debido a localizaciones, rutas o velocidades anormales). El resultado del análisis (rutas y nivel de anomalía) se exportan para su presentación en un sistema de representación cartográfico.

La aplicación se ha desarrollado en el lenguaje de programación Python y está basada en NuPIC, una plataforma *Open Source* de inteligencia artificial desarrollada por Numenta para detección de anomalías espacio temporales. Los datos del tráfico marítimo se han recibido en forma de mensajes AIS a través de *API for AIS Data* de Marine Traffic.

Los resultados del estudio pretenden mostrar la posibilidad de utilizar sistemas de Inteligencia Artificial para analizar la gran cantidad de datos relacionados con el tráfico marítimo disponible.

ABSTRACT

This work has the objective of designing an Artificial Intelligence system able to analyze and supervise Maritime traffic related information. Because of that, an application that using Artificial Intelligence techniques generates a model which represents the movement patterns and detects the ones that have an unusual behavior has been developed. The application is fed with geographical data from ships in a fixed area received by AIS messages from Marine Traffic's API for AIS Data. The results from the analysis are presented in a cartographical representation system.

The application has been developed in Python and is based on NuPIC, an open-source platform for Artificial Intelligence owned by Numenta. It is space-temporal anomaly detection aimed.

The study results are intended to show the possibility of using Artificial Intelligence systems in order to analyze the great amount of available maritime traffic related data.

PALABRAS CLAVE

Inteligencia Artificial, Anomalía, Marítimo, NuPIC, AIS, Python

AGRADECIMIENTOS

En primer lugar quiero agradecer a los tutores de este trabajo, Miguel Rodelgo Lacruz y José María Núñez Ortuño, su paciencia, apoyo y consejo durante toda la realización del trabajo.

Quiero agradecer también a Numenta por permitirme la utilización de su software para el desarrollo del programa, así como a Matthew Taylor, *Open Source Community Flag-Bearer*, por su disposición a ayudar y sus consejos.

No quiero dejar de agradecer tambien a Marine Traffic la inclusión de este trabajo dentro de su *MarineTraffic Research Network* y especialmente a Miluše Tichavska, *Academic Relations Manager*, por proporcionarnos acceso a sus diferentes servicios de obtención de datos de manera generosa.

Por último, quiero agradecer a mi familia todo el apoyo, la inspiración y los consejos recibidos así como su comprensión y dedicación durante el tiempo dedicado a mi formación.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contextualización

Desde las primeras civilizaciones el entorno marítimo ha sido el medio principal para el desarrollo económico y militar de los países. Los antiguos fenicios ya comprendieron la importancia del control de los mares, ampliando su imperio comercial a todo el Mediterráneo. Posteriormente, griegos, romanos, vikingos y otras civilizaciones extendieron el uso de los mares a fines bélicos. Con el avance de las tecnologías navales aumentó el peso que el uso del medio marítimo ejercía en el desarrollo de los países. Pero con el incremento de la actividad marítima también surgieron nuevas actividades contrarias a la legislación de los países, tales como la piratería, el contrabando, etc. Esto obligó a regular el uso del mar, lo que conllevó el inicio de una vigilancia por parte de los países de sus aguas costeras, creando buques e instituciones dedicados al control de la mar.

En el mundo global en el que vivimos, de acuerdo con [1], el 90% del comercio internacional se lleva a cabo por mar. Si este se interrumpiese, las importaciones y exportaciones serían imposibles y el 50% del mundo moriría de hambre, mientras que el otro 50% se congelaría por falta de combustible. Sin embargo, la gestión de este tráfico marítimo supone un enorme desafío para los países. La propia dureza del medio marino hace que los riesgos físicos a los que se enfrenta este tráfico sean muy numerosos: colisiones, averías, incendios, condiciones meteorológicas adversas, etc. Un accidente en el medio marino puede suponer un gran impacto humano (Costa Concordia 2012), medioambiental (Prestige 2002) y/o económico, ya sea directamente o derivados de los anteriores. En el accidente del crucero Costa Concordia en enero de 2012 frente a las costas italianas, fallecieron 32 personas y 64 fueron heridas de consideración. Su complejo rescate se estima que costó más de 600 millones de euros. En noviembre del 2002 el petrolero Prestige se hundía frente a las costas gallegas liberando más de 15.000 toneladas de crudo, originando un desastre ecológico de grandes proporciones. De acuerdo con la Comisión Permanente de Investigación de Accidentes Marítimos del Ministerio de Fomento [2], en el año 2014 se produjeron 97 accidentes e incidentes marítimos en las aguas de responsabilidad españolas.

Para intentar reducir estos riesgos, durante el siglo XIX se fundaron diferentes organizaciones internacionales, como la Organización Marítima Internacional (OMI), que regulasen la normativa referente a todo vehículo que circule por el mar. Sin embargo, la responsabilidad del correcto cumplimiento de estas normas, así como la de tomar las medidas oportunas para prevenir los accidentes, recae en los países a cuya soberanía pertenecen las aguas en cuestión. Para poder analizar y controlar el tráfico marítimo que cruza por sus aguas, los diferentes países han creado instituciones propias encargadas de monitorizar el tráfico marítimo y detectar irregularidades.

Mientras tanto, la Organización del Tratado del Atlántico Norte (OTAN), consciente también de la importancia del control del mar en el desarrollo de los países, comprendió que la protección de las rutas marítimas se trataba de una dimensión esencial de la seguridad. De esta manera surgió un nuevo tipo de operaciones, denominadas Operaciones de Seguridad Marítima (MSO, *Maritime Security Operations*). En estas, la amenaza deja de ser una marina de guerra convencional y el objetivo pasa a ser la seguridad de las rutas marítimas. Este nuevo concepto de operaciones introdujo la necesidad de conocer y entender el medio marino para ser capaces de conducir las operaciones manera eficaz. Así surgió el concepto de *Maritime Situational Awareness* (MSA). Esta se ha convertido en una herramienta indispensable para poder actuar en el mar. Consiste en el conocimiento del entorno marítimo: saber que está pasando en la mar para lograr una mayor efectividad en el planeamiento y la ejecución de las operaciones [3].

En España se ha redactado una Estrategia de Seguridad Marítima Nacional [4], encuadrada dentro de la Estrategia de Seguridad Nacional, en la que se incluye una visión integral de la seguridad marítima, donde se resalta la importancia del tráfico marítimo para el desarrollo de España; un análisis de los riesgos y amenazas para la seguridad marítima nacional; se dictan unos objetivos, principios y líneas de acción para hacer frente a esos riesgos y amenazas y se encuadra la seguridad marítima en el Sistema de Seguridad Nacional.

El control del tráfico que atraviesa nuestras aguas es llevado a cabo paralelamente por cuatro instituciones. Salvamento Marítimo cuenta con 20 Centros de Coordinación ubicados por todo el litoral español (Figura 1 – 1), los cuales controlan en torno a 350.000 buques al año. Las Capitanías Marítimas, dependientes de la Dirección General de la Marina Mercante, también se encargan de monitorizar el tráfico marítimo en aguas españolas. Estas dos instituciones cuentan con el apoyo de la Guardia Civil, a través del Servicio Marítimo (Figura 1 – 1), y de la Armada, a través de Acción Marítima y, más concretamente, del Centro de Operaciones y Vigilancia de Acción Marítima (COVAM).

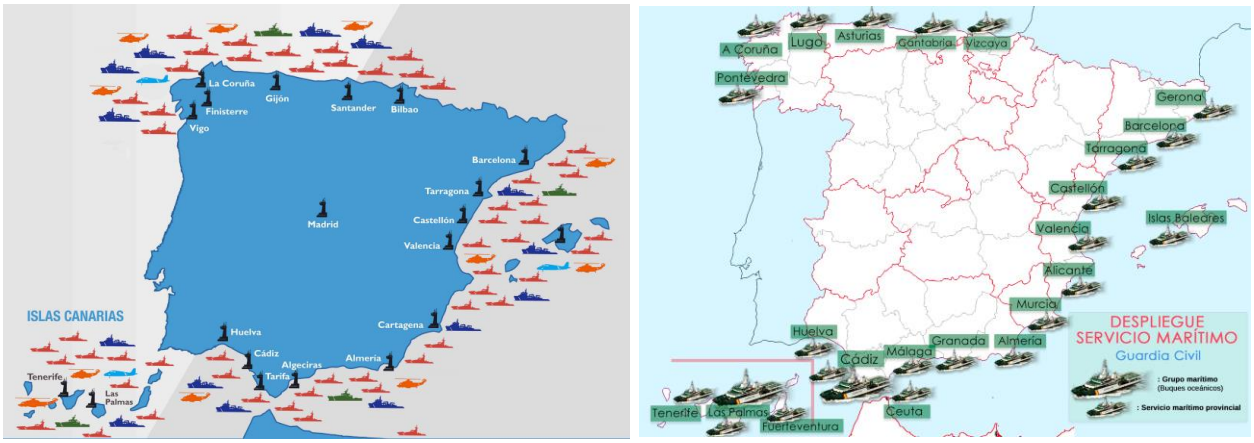

Figura 1-1 Red de Centros de Coordinación de Salvamento Marítimo y despliegue del Servicio Marítimo de la Guardia Civil

Estas instituciones se ven obligadas a desplegar una cantidad de medios materiales y humanos enorme con el fin de ser capaces de supervisar y analizar el tráfico marítimo frente a la costa española. Aun así, el volumen de la información recogida y el detalle de esta impiden que un operador humano sea capaz de procesar toda la información, centrándose sólo en aquella con un mayor valor inmediato. Este concepto de manejo de grandes cantidades de datos y su procesamiento para la obtención de algún tipo de conclusión o patrón se conoce como Datos a Gran Escala o, más comúnmente, como *Big Data* por su término en inglés.

Con el desarrollo de las tecnologías informáticas y de la información, parte de los avances logrados se han aplicado a la vigilancia marítima. El aprovechamiento de técnicas de procesado de datos y representación de la información, así como el uso de alarmas con parámetros preestablecidos, han aliviado en gran manera los esfuerzos humanos requeridos. El uso de los Servicios de Tráfico Marítimo (VTS, *Vessel Traffic System*) (Apartado 2.6) que integran todos los sensores y presentan la información de una manera intuitiva se encuentra muy extendido en puertos y rutas marítimas de importancia. Sin embargo, su capacidad para la detección de anomalías es muy limitada y sujeta en casi todos los casos al cumplimiento de unos parámetros preestablecidos. Apenas existen sistemas que hagan uso de la Inteligencia Artificial (AI, *Artificial Intelligence*) para cooperar con la capacidad humana. Todavía nos encontramos muy lejos de lograr obtener el máximo aprovechamiento de estas aplicaciones, que en otros ámbitos tienen una mayor penetración.

1.2 Motivación y Objetivos

Como se ha explicado en el anterior apartado, el control del tráfico marítimo supone un enorme esfuerzo para la Administración General del Estado, tanto en recursos materiales como en recursos humanos. Además, este esfuerzo no asegura que pequeñas anomalías puedan pasar inadvertidas a los ojos de un operador. La implantación de sistemas informáticos, de gestión de la información o de AI se presenta como una solución idónea para lograr un procesamiento eficaz y eficiente de toda la información disponible.

El uso de sistemas VTS se encuentra muy extendido, especialmente en puertos y rutas marítimas con un elevado tráfico, como los Dispositivos de Separación de Tráfico (DST) (Apartado 1.3). Sin embargo, la capacidad de estos sistemas para detectar anomalías, en caso de que exista, se ve sujeta al incumplimiento de unos parámetros preestablecidos. Existe un elevado número de estudios para la aplicación de técnicas de detección de anomalías en el entorno marítimo. A pesar de esto, los desarrollos prácticos son muy escasos y en la mayoría de los casos requieren un entrenamiento previo. Esto presenta inconvenientes como la necesidad de conocer las posibles anomalías de antemano o la imposibilidad de aplicar un mismo sistema en diferentes escenarios.

Es por esto que en este trabajo se persigue la creación de un sistema capaz de procesar el *Big Data* procedente de los sistemas VTS, aprendiendo las características del tráfico marítimo por sí sólo. Se busca que el sistema desarrollado no requiera un adiestramiento previo, si no que aprenda con la experiencia y sea capaz de extrapolar los datos de un escenario a otro, a través de un entendimiento del tráfico marítimo. La detección de anomalías es un factor clave en el conocimiento del entorno marítimo ya que sirve como herramienta para detectar cualquier comportamiento fuera de lo normal que pueda llevar a un posterior análisis de la situación. Podría ser utilizado como una herramienta o ayuda en MSO. Estos conocimientos, además de ser de gran utilidad en el mantenimiento de la seguridad marítima, podrían llegar a tener un gran valor en otros campos, como el económico o el estratégico.

El objetivo principal de este proyecto es la creación de un sistema sencillo de AI que a partir de una entrada de datos AIS (Sistema de Identificación Automática o *Automatic Identification System*) (Apartado 2.5) correspondientes a una zona geográfica concreta, mediante un sistema de redes neuronales, sea capaz de identificar los patrones de movimiento correspondientes a dicha zona y avise cuando un buque tenga un comportamiento anómalo. Para ello se creará una aplicación en Python utilizando el *framework* de NuPIC (Apartado 2.2.2), desarrollado por Numenta. La obtención de AIS se realizará desde la Interfaz de Programación de Aplicaciones (API, *Applications Programming Interface*) de Marine Traffic *API for AIS data*.

Para desarrollar la aplicación se ha partido de la aplicación *Geospatial Tracking* de Numenta (Apartado 2.3.1). Esta aplicación a su vez se basa en NuPIC, un proyecto de software libre basado en una teoría llamada Memoria Temporal Jerárquica o *Hierarchical Temporal Memory* (HTM) (Apartado 2.2.3).

Además de la creación de un sistema sencillo de AI de tratamiento de datos AIS, este proyecto persigue sentar una base para un desarrollo futuro de nuevos sistemas de AI para el tratamiento y análisis de información marítima. Se busca que en un futuro existan sistemas con una elevada capacidad de procesamiento capaces de recibir un mayor volumen de datos y realizar una asimilación más profunda de estos, llegando a crear patrones no solo locales, si no, a nivel global y con un carácter temporal histórico.

1.3 Marco geográfico

Aunque el objetivo final de la línea de trabajo aquí iniciada abarcaría todas las aguas sobre las que España tiene responsabilidad e intereses o cualquier otra zona de operaciones, para este trabajo vamos a centrarnos en un área geográfica determinada. Si el método de trabajo es efectivo para este área, suponemos que se podrá extender sin complicación a toda la costa española.

La región geográfica seleccionada para estudiar en este trabajo es el Dispositivo de Separación del Tráfico (DST) "a la altura de Finisterre". De acuerdo con [5], un DST es un método de regulación del flujo de buques moviéndose en diferentes direcciones. De acuerdo con la OMI, un DST es un esquema que separa el tráfico marítimo circulando en direcciones opuestas o casi opuestas mediante el uso de una línea, una zona de separación u otros medios.

Frente al cabo Finisterre, debido a la alta densidad del tráfico marítimo, la OMI implantó un DST para regular el tráfico en esa zona. A raíz del accidente del Prestige (2002) y la crisis medioambiental y económica que produjo, en el año 2003 la OMI decidió modificar este DST. A través de la Resolución A.957 (23) adoptada el 5 de diciembre de 2003 (Punto 17 del orden del día) [6] se decidió que el 1 de junio de 2004 entrase en vigor el nuevo DST, en el cual se añadían dos vías de circulación, quedando el DST conformado por cuatro vías de circulación. Las dos vías más al oeste son utilizadas por los buques que se dirigen hacia el sur y las dos situadas al este son utilizadas por los buques con rumbo norte (Figura 1 – 2). En ambos casos, la vía más exterior es utilizada por los buques que transportan cargas peligrosas a granel.

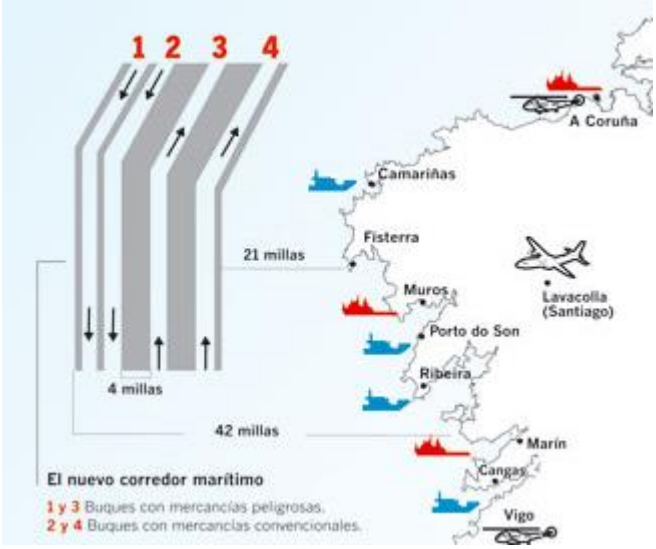

Figura 1-2 Dispositivo de Separación del Tráfico a la altura de Finisterre

Este DST es el segundo más transitado de España (por detrás del DST situado en el Estrecho de Gibraltar) ya que es atravesado por las líneas de tráfico que unen el norte de Europa con África y el Mediterráneo. El año pasado circularon por él más de 35000 barcos, aunque su uso ha decrecido en los últimos años (Figura 1-3) debido, posiblemente, a un endurecimiento de las medidas de seguridad, como consecuencia del accidente del Prestige (2002).

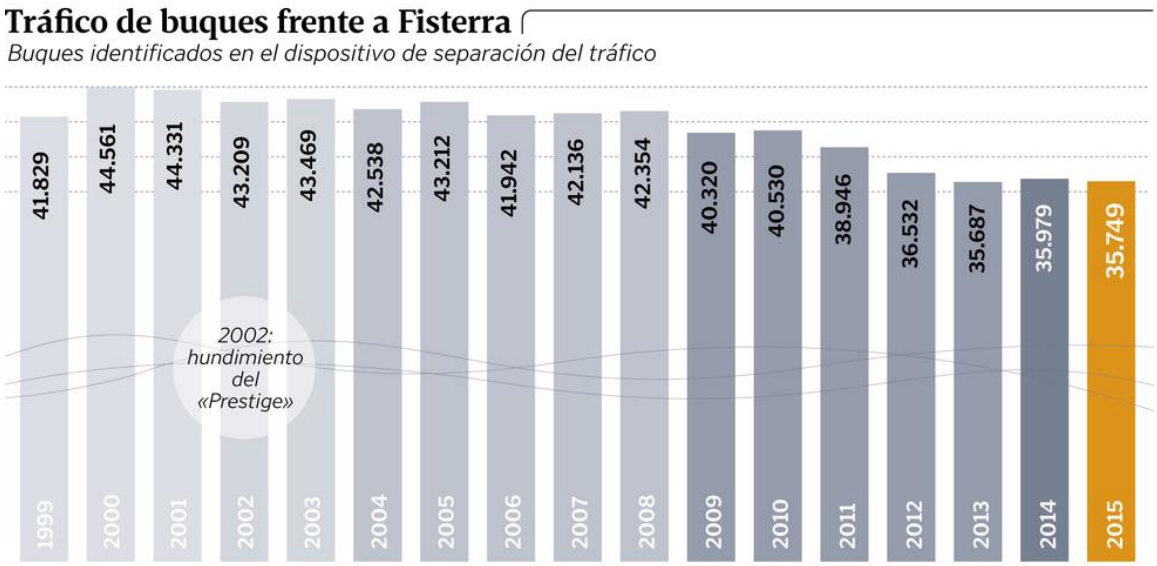

Figura 1-3 Número de buques identificados en el DST de Finisterre por año desde 1999 [7]

Para controlar el tráfico que circula por el DST existe Finisterre Tráfico, un centro con el que han de comunicar todos los buques que vayan a utilizar del DST antes de entrar en este. En este centro se recibe un gran flujo de información por diferentes medios, como radar, AIS, GMDSS (Sistema Mundial de Socorro y Seguridad Marítimos o *Global Maritime Distress Safety System*), información meteorológica, etc. Toda esta información ha de ser procesada por el personal del centro con el fin de comprobar el correcto funcionamiento del DST.

1.4 Organización de la Memoria

Una vez puesto este trabajo en contexto y habiendo definido los objetivos del mismo, se describe a continuación la organización del trabajo y el desglose de sus apartados. El trabajo se divide en seis capítulos, complementados por dieciséis anexos.

En el Capítulo 1 se ha introducido y puesto en contexto el trabajo. Además, se han expuesto las razones que lo han motivado y los objetivos que se han impuesto. Se ha realizado también una breve descripción del marco geográfico seleccionado para testar la aplicación a desarrollar.

En el Capítulo 2 se realiza una introducción teórica y se revisa el estado del arte:

- En primer lugar se define la Inteligencia Artificial, se repasan los hitos más importantes de su historia y se revisa su estado en la actualidad.
- En segundo lugar se habla de Numenta, explicando qué es y cuál es su misión. Se habla también de NuPIC, un proyecto de Numenta. A continuación se realiza una introducción a la Memoria Jerárquica Temporal.
- En tercer lugar se exponen las aplicaciones desarrolladas por Numenta, explicando *Geospatial Tracking* con mayor detenimiento. Se enuncian también otras aplicaciones basadas en Memoria Jerárquica Temporal no desarrolladas por Numenta.
- En cuarto lugar se explica qué es y cómo funciona el Sistema de Funcionamiento Automático (AIS). Se habla también de los Sistemas de Gestión del Tráfico Marítimo (VTS) y se citan ejemplos de estos.
- En quinto lugar se exponen los trabajos previos realizados sobre la detección de anomalías en el entorno marítimo.
- En sexto lugar se explica brevemente en qué consisten los Archivos Keyhole Markup Language (KML)

En el Capítulo 3 se desarrolla el proyecto. Inicialmente se describe el desarrollo del entorno necesario para generar la aplicación. A continuación se desgrana la aplicación, explicando el funcionamiento de cada uno de los módulos que la componen.

En el Capítulo 4 se describen las diferentes pruebas realizadas y se analizan los resultados obtenidos, buscando posibles soluciones y mejoras a los problemas encontrados.

En el Capítulo 5 se realiza una valoración del trabajo y se exponen las conclusiones obtenidas. Además se enuncian las líneas de trabajo a desarrollar en un futuro.

El Capítulo 6 incluye las referencias bibliográficas y los recursos web consultados para el desarrollo del trabajo.

Por último, se adjuntan dieciséis anexos que incluyen:

- Anexo I: diccionario de siglas y acrónimos.
- Anexos II, III y IV: ejemplos de aplicaciones HTM y VTS.
- Anexos V, VI, VII, VIII, IX, X y XI: código en Python que forma la aplicación.
- Anexos XII, XIII, XIV, XV y XVI: extractos de ejemplos de los archivos CSV y KML de entrada, de salida e intermedios de la aplicación.

2 INTRODUCCIÓN TEÓRICA Y ESTADO DEL ARTE

En esta sección se describen los principios teóricos así como las diferentes tecnologías utilizadas en este trabajo, entre las que se encuentran NuPIC, el software de AI desarrollado por Numenta, y los archivos KML. Se enuncian, además, los diferentes estudios y aplicaciones existentes acerca del tratamiento de la información y la detección de amenazas referentes al entorno marítimo.

2.1 Inteligencia Artificial

2.1.1 Definición

De acuerdo con J. McCarthy, del departamento de Ciencias Informáticas de la Universidad de Stanford [8], la AI es la ciencia de construir maquinas inteligentes y, más específicamente, programas de ordenador inteligentes. Está relacionada con la tarea de usar ordenadores para entender la inteligencia humana, aunque la AI no tiene por qué reducirse a métodos que sean observables biológicamente.

Esta definición queda incompleta si no va acompañada de la definición de inteligencia. El mismo McCarthy la define cómo la parte computacional de la habilidad de lograr objetivos en el mundo. Según McCarthy, se dan diferentes tipos y grados de inteligencia en la personas, en muchos animales y en algunas máquinas.

Sin embargo, existen muchas otras orientaciones a la hora de definir la AI. Actualmente el criterio más aceptado es el de los cuatro enfoques de Russell-Norvig [9]:

- Sistemas que actúan como humanos

  El estudio de cómo lograr que los ordenadores realicen tareas que, de momento, la gente hace mejor (Rich y Knight, 1991)

- Sistemas que piensan como humanos

  El esfuerzo de hacer que los ordenadores piensen… máquinas con mentes en el más amplio sentido literal (Haugeland, 1985)

- Sistemas que piensan racionalmente

  El estudio de las facultades mentales a través del estudio de modelos computacionales (Charniak y McDermott, 1985)

- Sistemas que actúan racionalmente

  El estudio que busca explicar y emular el comportamiento inteligente en términos de procesos computacionales (Shalkoff, 1990)

2.1.2 Historia

Se puede situar el nacimiento de la AI en 1943, cuando W. McCulloch y W. Pitts definieron el primer modelo de neurona [10]. En 1951, M. Minsky y D. Edmons construyeron la primera máquina de redes neuronales, SNARC. En 1950 A. Turing enunció su famoso test para medir si una máquina es inteligente [11] en el cual establecía que una máquina es inteligente cuando esta sea capaz de engañar a un evaluador humano. La efectividad del test de Turing ha sido bastante cuestionada y en 1980 J. Searle propuso el experimento de la "habitación china" [12] en el que ponía en entredicho que una máquina fuese inteligente por el hecho de realizar acciones asociadas a la inteligencia.

Sin embargo, no fue hasta 1956 en la Conferencia de Dartmouth que se consolidó la AI como campo de actividad. Para esta conferencia pusieron como conjetura inicial que cada aspecto del aprendizaje o de cualquier otra característica de la inteligencia puede ser descrito con suficiente precisión para que una máquina pueda simularlo [13].

Durante los años cincuenta, debido a los éxitos iniciales, se instauró una actitud optimista en el ámbito de la AI. Programas como el Juego de damas de A. Samuel o el *Logic Theorist* y el *General Problem Solver* de Newell, Simon y Shaw de la *RAND Corporation* condujeron a un clima de confianza plena en la AI para alcanzar las metas propuestas. Así, se llegó a afirmar que "en un par de años las máquinas serían capaces de realizar cualquier tarea realizada por el hombre" [14].

Estos programas funcionaban bien para problemas muy específicos, pero seguían fallando en problemas reales que los seres humanos resolvemos de manera sencilla, como la traducción (*ALPAC report*, 1966). Se dio inicio así a lo que se denominaría como el invierno de la AI. Ante la carencia de resultados inmediatos y el fracaso de los modelos conexionistas, los fondos destinados a la AI se redujeron drásticamente.

A durante los años 80 la AI vuelve a ponerse en el punto de mira. Los sistemas expertos desarrollados a finales de los años 70 fueron adoptados por empresas a lo largo de todo el mundo y el conocimiento se convierte en el foco de la investigación en AI. Este rápido crecimiento generó unas expectativas demasiado altas. La incapacidad para cumplir dichas expectativas y la aparición de sistemas más económicos, como los ordenadores de sobremesa de Apple e IBM, supusieron un desencanto por parte de las fuentes de financiación y una estigmatización del término "Inteligencia Artificial".

2.1.3 Inteligencia Artificial en la actualidad

El desencanto sufrido por la sociedad a finales de los 80 y principios de los 90 ha supuesto dos grandes consecuencias en el desarrollo de la AI.

En primer lugar, los investigadores desistieron del optimismo inicial de este campo. Ante la complejidad de los problemas a los que se enfrentan se ha optado por fragmentar la investigación en diferentes ramas, como representación del conocimiento, planificación autónoma, juegos, control autónomo, diagnosis, planificación logística, robótica, procesamiento de lenguaje y resolución de problemas entre otras.

En segundo lugar, debido a la condición de tabú adquirida por el término "Inteligencia Artificial", numerosos desarrollos de la AI han optado por usar diferentes eufemismos para así escapar del rechazo comercial a la AI. De esta manera, aunque no seamos conscientes, la AI inunda el mundo moderno. Algunos ejemplos de aplicaciones de la AI en diferentes ámbitos son los *bots* (programa informático que imita el comportamiento de un humano) utilizados por los videojuegos, el motor de búsqueda de Google, sistemas de reconocimiento de lenguaje, como Cortana, sistemas de piloto automático, como el utilizado por Tesla o el sistema ORES, que usa Wikipedia para comprobar las ediciones, entre otros.

2.2 Numenta

2.2.1 Introducción

Numenta es una compañía con base en Redwood City, California, que ha desarrollado una teoría conjunta, tecnología de código fuente y numerosas aplicaciones basadas en los principios que rigen el neocórtex [15]. Numenta basa toda su tecnología en la *Hierarchical Temporal Memory* o Memoria Temporal Jerárquica (HTM), una teoría computacional del neocórtex. Fue fundada el 24 de marzo de 2005 por Jeff Hawkins, el fundador de Palm, Donna Dubinsky y Dileep George. Numenta sigue un modelo de negocio basado en el licenciamiento de su software y propiedad intelectual. Creen que la HTM puede lograr un gran impacto en el mundo y que una estrategia de licencias extendida permitirá a esta tecnología expandirse a muchas aplicaciones diferentes.

Figura 2-1 Logo Numenta [16]

Para ellos, debido a su naturaleza, los ordenadores de hoy en día solo pueden hacer aquello que se les ha indicado. Por el contrario, las máquinas inteligentes aprenden patrones de su entorno continuamente, sin la necesidad de ser programados. Esto les permite afrontar los problemas de manera novedosa.

2.2.2 Memoria Jerárquica Temporal (HTM)

En Numenta consideran al cerebro como el mejor ejemplo de un sistema inteligente. Es por eso que utilizan el cerebro, y más en concreto el neocórtex, como modelo para construir máquinas inteligentes. El neocórtex controla un amplio rango de funciones usando un conjunto de principios comunes. Numenta ha centrado sus esfuerzos en descubrir esos principios y a partir de ellos ha creado sus algoritmos de aprendizaje.

Numenta usa tecnología Memoria Jerárquica Temporal (HTM) [17], una detallada teoría computacional sobre el neocórtex. La HTM es una tecnología de aprendizaje de máquina que pretende capturar las propiedades estructurales y algorítmicas del neocórtex. Fue enunciada por primera vez por Jeff Hawkins en [18]. En el corazón de la HTM están los algoritmos de aprendizaje con base temporal que almacenan y recuperan los patrones espaciales y temporales. La HTM está especialmente diseñada para afrontar problemas con las siguientes características: flujo de datos en vez de bases de datos estáticas, patrones en la variación de los datos con el tiempo, múltiples fuentes de información en las que modelos separados manualmente no serían prácticos, patrones no evidentes que difícilmente pueden ser identificados por humanos o secuencias temporales.

Las HTM, a diferencia de los ordenadores tradicionales, no están programadas para solucionar problemas específicos, sino que sus capacidades se determinan en función de los datos a los que son expuestas. Las HTM son un tipo de red neuronal en el que las neuronas, denominadas células, se organizan en columnas, capas, regiones y en una jerarquía. Las HTM también difieren de los ordenadores en el modo de organizar la memoria. La memoria HTM es jerárquica y está basada en el concepto de tiempo. La información se guarda de manera distribuida y el usuario no tiene control sobre dónde y cómo se almacena.

Una red HTM está formada por regiones organizadas en una jerarquía. Cada región HTM se corresponde con un nivel en la jerarquía. Según se asciende la jerarquía existe una convergencia y,

debido a las conexiones de retroalimentación, al descender la información también diverge. En el caso de obtener datos de varias fuentes o sensores se pueden combinar múltiples redes HTM. Esta jerarquía reduce el tiempo de entrenamiento, el uso de memoria y aportan una forma nueva de generalización. Aun así, muchos de los problemas simples que forman la predicción se pueden resolver con una sola región HTM.

En las regiones HTM, la información se representa por un pequeño número de células activas, es decir, por una Representación Distribuida Dispersa (SDR, *Spread Distributed Representation*), como se puede apreciar en la Figura 2-2. Para ello, la región HTM transforma los datos de entrada en una representación interna donde solo un porcentaje pequeño de los bits permanecería activo, sin importar el número de bits activos al inicio.

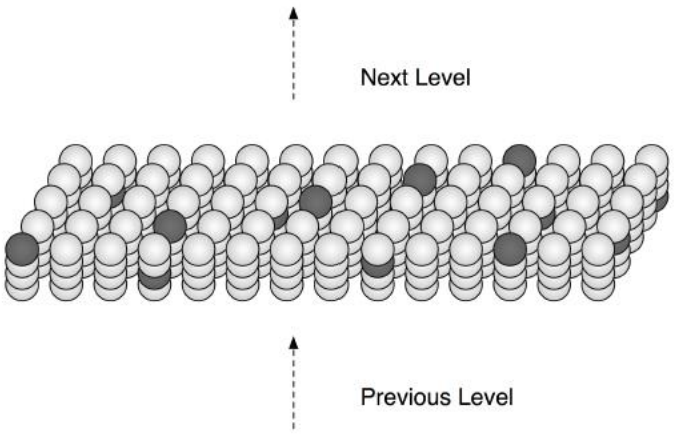

Figura 2-2 Una región HTM mostrando activación celular distribuida dispersa [17]

Uno de los factores determinantes en el aprendizaje, la inferencia y la predicción es el tiempo. Para que se pueda producir el aprendizaje por un sistema HTM, este necesita una sucesión de entradas sensoriales continuas en el tiempo. Cada región busca combinaciones de entradas que se den simultáneamente con periodicidad, denominadas patrones espaciales. Posteriormente busca con que secuencia se dan estos patrones en el tiempo, lo que se denomina patrones temporales.

Al recibir una entrada, una HTM mapea los patrones espaciales y temporales que ha aprendido anteriormente. Para ello utiliza una de las propiedades más importantes que cumplen las distribuciones dispersas, qué es que solo es necesario que coincida una parte del patrón para saber que el mapeo es relevante. Este mapeo permite a una región generar predicciones sobre las próximas entradas. Esta predicción se da de manera continua y se produce en cada región y a cada nivel de la jerarquía. Estas predicciones, además, son dependientes del contexto, una misma entrada genera diferentes predicciones en función de las entradas anteriores. Las predicciones varían con diferentes ratios según el nivel en la jerarquía, generando así una estabilidad en la predicción del sistema.

Es a partir de estas predicciones que los sistemas HTM adquieren su capacidad de detección de anomalías. Cada vez que la HTM recibe una entrada, la compara con su correspondiente predicción y, en caso de que exista una desviación con ella, la identifica como una anomalía. Pero estas predicciones no tienen solo un empleo final. Las predicciones son utilizadas por la HTM para rellenar los huecos que puedan quedar en la entrada, haciendo así al sistema mucho más robusto frente al ruido.

Para poder implementar la HTM se han desarrollado unos Algoritmos de Aprendizaje Cortical (CLA, *Cortical Learning Algorithms*). Estos algoritmos recogen los datos de entrada y forman una distribución dispersa de ellos. Para generar estas distribuciones utiliza los denominados codificadores o *encoders*, los cuales toman datos sensoriales de entrada y generan una matriz binaria de activaciones de células. A continuación forman una representación de la entrada sobre el contexto de las entradas

previas y finalmente realizan una predicción basada en la entrada actual y el contexto de las entradas previas [17].

2.2.3 NuPIC

Numenta Platform for Intelligent Computing (NuPIC) [15] es un proyecto de software libre basado en HTM. Partes de la teoría HTM han sido implementadas, probadas y usadas en aplicaciones y otras están todavía siendo desarrolladas. NuPIC incluye grupos de debate en teoría HTM, investigación para extender los algoritmos HTM y código fuente para aplicaciones completas basadas en HTM. NuPIC es la apuesta de software libre para el desarrollo de la HTM de Numenta.

Para su funcionamiento, NuPIC implementa algoritmos CLA [19]. Estos algoritmos describen las operaciones realizadas en una única capa de neuronas corticales. Estas "capas" pueden aprender patrones espaciales y temporales a partir de los datos de entrada por si solas. Las "capas" pueden unirse de una manera jerarquizada para lograr sistemas más complejos que aprendan patrones de alto nivel. Cada columna representa un significado y cada célula dentro de cada columna representa el mismo significado pero con un contexto diferente. El aprendizaje se produce formando y deshaciendo conexiones entre neuronas.

NuPIC está formado por [20]:

- Componentes sensoriales, que reciben la información del exterior, ya sea de archivos CSV, bases de datos, APIs, etc.
- Codificadores, que convierten la información sensorial en representaciones binarias relativamente distribuidas similares a las SDR.
- Una jerarquía de regiones, donde cada una de las regiones aprende a identificar los patrones espaciales de sus entradas (*spatial pooler*, SP) y los patrones temporales de esos patrones espaciales (*temporal pooler*, TP). A continuación entrega representaciones de los patrones espaciales y de las predicciones que realiza sobre futuras entradas.
- Un clasificador, que es usado para obtener información valiosa de las salidas del sistema. Puede incluir una clasificación de la entrada actual, una serie de predicciones de los valores de la entrada en los futuros ciclos y un indicador de la anomalía o nivel de sorpresa de la entrada actual.

A través del *anomaly score*, NuPIC es capaz de proveer una métrica que represente al grado en el que cada nuevo dato es predecible [21]. Un cero representa un valor completamente predecible, mientras que un cero representa un valor completamente anómalo. Esto se calcula como el porcentaje de columnas de la SDR que fueron incorrectamente predichas:

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

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

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

2.3 Aplicaciones de Numenta

Numenta, tanto en solitario como mediante alianzas comerciales con otras empresas, ha desarrollado aplicaciones basadas en sus CLA en diferentes ámbitos. Ejemplos de estas aplicaciones son *Grok, HTM for Stocks, Rogue Behaviour Detection* o *Geospatial Tracking*. El funcionamiento de las tres primeras aplicaciones se desarrolla en el Anexo II.

2.3.1 Geospatial Tracking

La aplicación *Geospatial Tracking* [22] utiliza HTM para, de manera automática, crear modelos de patrones de movimiento y velocidad e identificar anomalías en los desplazamientos geográficos. *Geospatial Tracking* puede servir de base para generar aplicaciones relacionadas con la monitorización

de personal, control de flotas, logística, prevención de contrabando o seguridad aérea entre otras. Esta aplicación posee un alto grado de flexibilidad, escalabilidad y precisión comparado con otras técnicas de monitorización geoespacial debido a la generación automática de modelos de patrones de movimiento a nivel unidad. Codificando los datos GPS y la velocidad en SDR, que posteriormente son modeladas por los CLA, se aprenden los patrones de movimiento y las anomalías son detectadas.

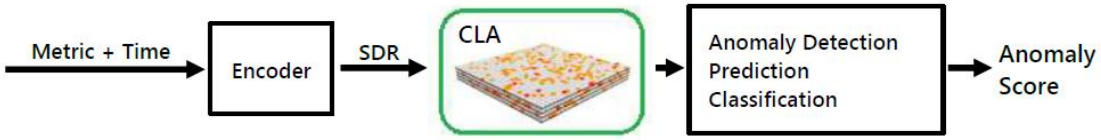

Figura 2-3 Esquema de funcionamiento de *Gesopatial Tracking* [22]

Para generar las SDR de las posiciones GPS, *Geospatial Tracking* usa uno de los codificadores de NuPIC denominado *Geospatial Coordinate Encoder* o Codificador de Coordenadas Geoespaciales. Este codificador cumple dos condiciones: las posiciones geográficamente próximas han de tener bits solapándose en las SDR y la resolución del movimiento ha de ser proporcional a la velocidad. Para generar la SDR el codificador crea una cuadricula en la zona del mapa donde el objeto se está desplazando y asigna a cada recuadro un valor aleatorio entre 1 y 0 y una posición en el SDR de la salida. El tamaño de la celda viene definido por la escala, que indica la máxima resolución del codificador. Es un valor crítico y depende de cada escenario. Una escala demasiado alta restringirá la detección de variaciones en el movimiento mientras que una demasiado pequeña aumentará el tiempo necesario para realizar la codificación y la sensibilidad al ruido. A continuación localiza la cuadricula dentro de la que se encuentra la posición actual y dibuja un cuadro alrededor de esta de tamaño proporcional a la velocidad. Finalmente se selecciona un número preestablecido de recuadros que se eligen siguiendo una función fija y se activan sus bits correspondientes en la salida (Figura 2-4).

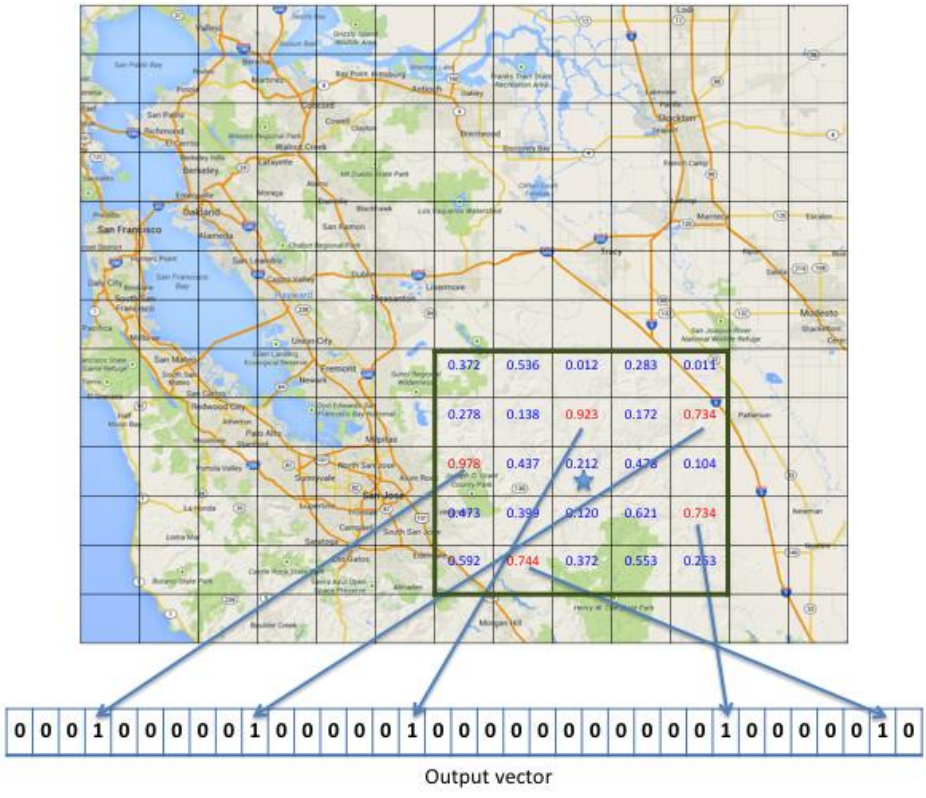

Figura 2-4 Generación de una SDR a partir de una entrada de latitud, longitud y velocidad por el Codificador de Coordenadas Geoespaciales [22]

2.4 Otras aplicaciones HTM

2.4.1 Numenta HTM Challenge

El *Numenta HTM Challenge* [23] es una competición online de 6 semanas en la que los participantes han de crear aplicaciones basadas en HTM que utilicen bases de datos del mundo real. En el Anexo III se enuncian los trabajos presentados en su primera edición.

2.4.2 ATAD. Air Traffic Anomaly Detector

ATAD [24] es una aplicación que usa información de vuelos para detectar y predecir anomalías de geoposición, velocidad y tiempo. La aplicación está construida sobre MoClu (motor HTM para aplicaciones java). De esta manera se ha podido añadir hardware según las necesidades para trabajar con cantidades enormes de datos de vuelos en tiempo real. En el lado del interfaz de usuario se ha usado *AngularJS* que se integra con *Lift 3* y *Comet actors*, una implementación de Angular para Google Maps llamada angular-google-maps y cartas D3. El *backend* se ha construido utilizando dos servidores web diferentes.

2.4.3 NuPIC Hackathon

Desde el otoño de 2013, NuPIC celebra cada año dos "hackatones[1]" en otoño y primavera. En ellos, los participantes disponen 32 horas para crear una aplicación basada en HTM. Se pueden comprobar las aplicaciones desarrolladas en cada edición en el enlace [25].

Figura 2-5 Logo del *NuPIC Hackathon* de otoño del 2014

Especial mención merece *Smart Harbor*, la demo número 7 del *2014 Fall NuPIC Hackathon*, desarrollada por D. Ducro y E. Wietses, de *Pionect*. Esta aplicación utiliza HTM para monitorizar las posiciones geográficas de los buques mercantes en el puerto de Róterdam para una posterior detección de anomalías. En el enlace [26] se puede ver un video de la presentación de la aplicación durante la competición.

[1] Un *hackathon* o "hackatón", es un término usado en las comunidades hacker para referirse a un encuentro de programadores cuyo objetivo es el desarrollo colaborativo de software, aunque en ocasiones puede haber también un componente de hardware.

2.5 Estudios previos sobre detección de anomalías en el entorno marítimo

Desde la proliferación del uso de sistemas AI para la detección de anomalías se ha identificado al ámbito de la vigilancia marítima como uno de los campos en el que más se podrían aprovechar estas técnicas. Los estudios teóricos sobre el uso de sistemas AI para la detección de anomalías en el entorno marítimo son numerosos. Sin embargo, este hecho choca con la carencia de aplicaciones reales que hagan uso de estas tecnologías.

En los estudios existentes se proponen diferentes técnicas para la detección de anomalías. Así pues, Janssens, Postma y van den Herik proponen en [31] un sistema de detección de anomalías basado en la densidad. Este sistema permitiría la detección de anomalías sin requerir un entrenamiento previo y sería capaz de acomodarse a la situación del tráfico en cada momento. Sin embargo, requiere la existencia de un elevado tráfico para localizar una anomalía.

Otro enfoque diferente es el utilizado por el *Swedish ICT* en su *Statistical Anomaly Detection and Visualization for Maritime Domain Awareness* [32]. En este caso las anomalías se detectan a partir de un histórico de datos estadísticos y de unos parámetros introducidos por el usuario. El mayor inconveniente de este sistema es que requiere una configuración específica para cada área determinada, así como un conocimiento de que parámetros son anómalos por parte del usuario.

Por su parte, C. Brax [33] propone utilizar el *State-Based Anomaly Detection Method* (SBADM), un sistema en el que en una fase de entrenamiento se crea un modelo con el que posteriormente se compararan los casos reales para identificar la anomalías. Los fundamentos teóricos del SBADM se presentan en el capítulo 4 de [34]. Para testarlo realizó experimentos junto con la Administración Marítima Sueca (Sjöfartsverket).

Existen trabajos, como [35] donde se analizan las necesidades y los retos a los que se enfrenta cualquier sistema de detección de anomalías. Además, se listan algunos de los tipos de anomalías más importantes y la capacidad de diferentes técnicas para identificar cada uno de los tipos.

En [36], el estudio se centra en la detección de anomalías en los datos de trayectoria. Tras hacer un breve análisis de los algoritmos existentes, el autor propone un nuevo algoritmo, diferentes sistemas para testar su algoritmo y muestra los resultados obtenidos.

2.6 Sistema de Identificación Automática (AIS, *Automatic Identification System*)

De acuerdo con [27] el AIS es un sistema de comunicaciones marítimas para la seguridad en la navegación estandarizado por la Unión Internacional de Telecomunicaciones (UIT) y adoptada por la Organización Marítima Internacional (OMI). El AIS provee información sobre buques, incluyendo su identidad, tipo, posición, rumbo, velocidad, estado y más información relacionada con la seguridad, de manera automática a estaciones en tierra debidamente equipadas, otros buques y aviones; recibe automáticamente esa información de buques igualmente equipados; monitoriza y realiza seguimiento de buques e intercambia información con instalaciones basadas en tierra.

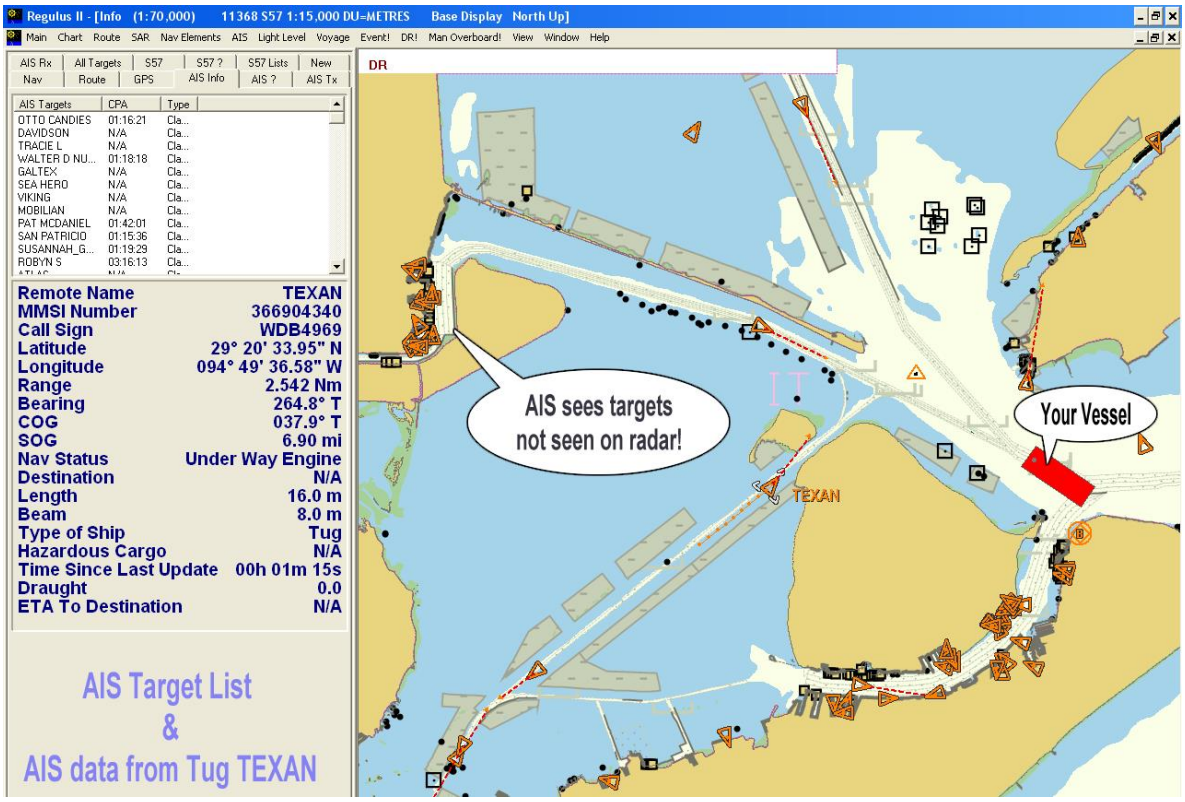

Figura 2-6 Ejemplo de visualización de la presentación de un equipo AIS [28]

El AIS actúa como un transpondedor, operando en la banda del VHF marítimo, que es capaz de manejar más de 4500 mensajes por minuto y que se actualiza cada dos segundos. Para alcanzar esta alta capacidad de radiodifusión y asegurar una operatividad barco-barco de confianza, el AIS utiliza una tecnología de Acceso Múltiple con División de Tiempo Auto-Organizada (SOTDMA, *Self-Organized Time Division Multiple Access*) [29].

2.7 Sistemas de Gestión del Tráfico Marítimo (VTS, *Vessels Traffic System*)

De acuerdo con la OMI [30], los VTS son sistemas ubicados en tierra que realizan tareas desde proveer a los buques mensajes de información simples, como la posición de otros buques o avisos meteorológicos, hasta gestionar el tráfico de un puerto o ruta marítima.

Los VTS recopilan la información recogida por los diferentes sensores disponibles (AIS, radar, CCTV (Circuito Cerrado de Televisión), GMDSS, estaciones meteorológicas, etc.) y, tras procesarla, la presentan al operador de una manera unificada. Esto simplifica en gran medida el trabajo de los operadores, reduciendo el número de consultas necesarias. La mayoría de estos sistemas permiten fijar a través de parámetros fijos alarmas para la detección de anomalías.

En algunos casos cuentan con sistemas de AI que les permiten realizar predicciones tanto del tráfico en general, como del comportamiento de buques en particular, permitiendo una mayor precisión en la detección de anomalías. Sin embargo, este campo se encuentra todavía dando sus primeros pasos y los VTS que lo incluyen hacen uso de sistemas todavía rígidos y que requieren un entrenamiento previo.

El hecho de que la gran mayoría de puertos y rutas marítimas de importancia estén haciendo uso de VTS ha generado un amplio mercado de estos sistemas. Actualmente todas las empresas de tecnología del ámbito marítimo ofrecen un sistema VTS. En el Anexo IV se indican algunos ejemplos.

2.8 Archivos *Keyhole Markup Language* (KML)

KML es una gramática XML y un formato de archivo para la creación de modelos y el almacenamiento de funciones geográficas como puntos, líneas, imágenes, polígonos y modelos que se mostrarán en Google Earth, Google Maps y otras aplicaciones [37]. KML es un estándar abierto oficial denominado el OpenGIS® KML Encoding Standard (OGC KML). Es mantenido por el Open Geospatial Consortium, Inc. (OGC).

Los archivos KML son procesados por Google Earth de una manera similar a como los navegadores web procesan los archivos HTML y XML. Al igual que los archivos HTML, los KML cuentan con una estructura basada en etiquetas con nombres y atributos utilizados para poder visualizarlos. Los archivos KML son utilizados por muchas aplicaciones, como Google Earth, Google Maps, Google Maps para móviles, NASA WorldWind, ESRI ArcGIS Explorer, Adobe PhotoShop, AutoCAD o Yahoo! [38]. Se dispone de un ejemplo de archivo KML en el Anexo XVI.

3 DESARROLLO DEL TFG

3.1 Desarrollo del entorno

3.1.1 Sistema operativo

Para el desarrollo de este trabajo se optó por trabajar sobre Ubuntu. Para tomar esta decisión fue determinante el hecho de que NuPIC, a pesar de su reciente ampliación a Windows en la versión 0.3.6, ha sido desarrollado para Ubuntu e IOS principalmente. Así pues, utilizando Ubuntu, tenemos acceso a una mayor cantidad de recursos y documentación. La versión de Ubuntu utilizada fue la 14.04 LTS.

3.1.2 Instalación NuPIC

Para la instalación de NuPIC se siguieron las directrices de la Wiki de NuPIC en GitHub [39] y más concretamente el video tutorial [40]. Nuestro punto de partida es una máquina virtual de Ubuntu de 64bits para VirtualBox con la distribución estándar.

Antes de proceder a la instalación de NuPIC, este requiere las siguientes dependencias:

- Python 2.7 & development headers
- pip
- wheel
- numpy
- C++ compiler

Ubuntu ya cuenta con un compilador de C++ (gcc).

En caso de carecer de Python 2.7 se puede instalar siguiendo las instrucciones de [41]. Además, hay que instalar también los *development headers*:

```
hg clone https://hg.python.org/cpython
hg update 2.7
sudo apt-get build-dep python2.7
sudo apt-get install python-dev
```

Para instalar pip servirá con introducir el siguiente comando en el terminal:

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

En algunos casos es necesario actualizar *apt-get* antes de realizar la instalación. El resto de dependencias serán instaladas de manera automática junto a NuPIC. El siguiente paso es la instalación de los *NuPIC bindings*, la cual realizaremos mediante pip. Para ello introduciremos el siguiente código:

```
pip install https://s3-us-west-2.amazonaws.com/artifacts.numenta.org/numenta/nupic.core/releases/nupic.bindings/nupic.bindings-0.3.1-cp27-none-linux_x86_64.whl --user
```

Con este comando, además de los *bidings* instalaremos otras dependencias, como numpy. Finalizada la instalación el sistema nos devolverá un listado de las dependencias instaladas:

```
Successfully installed nupic.bindings pytest-xdist numpy pytest pytest-cov execnet py cov-core apipkg coverage
```

A continuación procedemos a la instalación de NuPIC mediante pip:

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

Al igual que en el caso anterior, aquí se instalan otras dependencias además de NuPIC:

```
Successfully installed nupic PyMySQL pyproj validictory mock ordereddict DBUtils unittest2 coverage prettytable PyYAML python-dateutil asteval psutil
```

Ya tenemos instalado NuPIC. Lo siguiente que haremos será correr unos tests. Para ello instalamos git y clonamos el directorio de NuPIC en GitHub en nuestro ordenador. Antes de correr los tests también necesitaremos instalar py.test:

```
sudo apt-get install git

git clone https://github.com/numenta/nupic.git
```

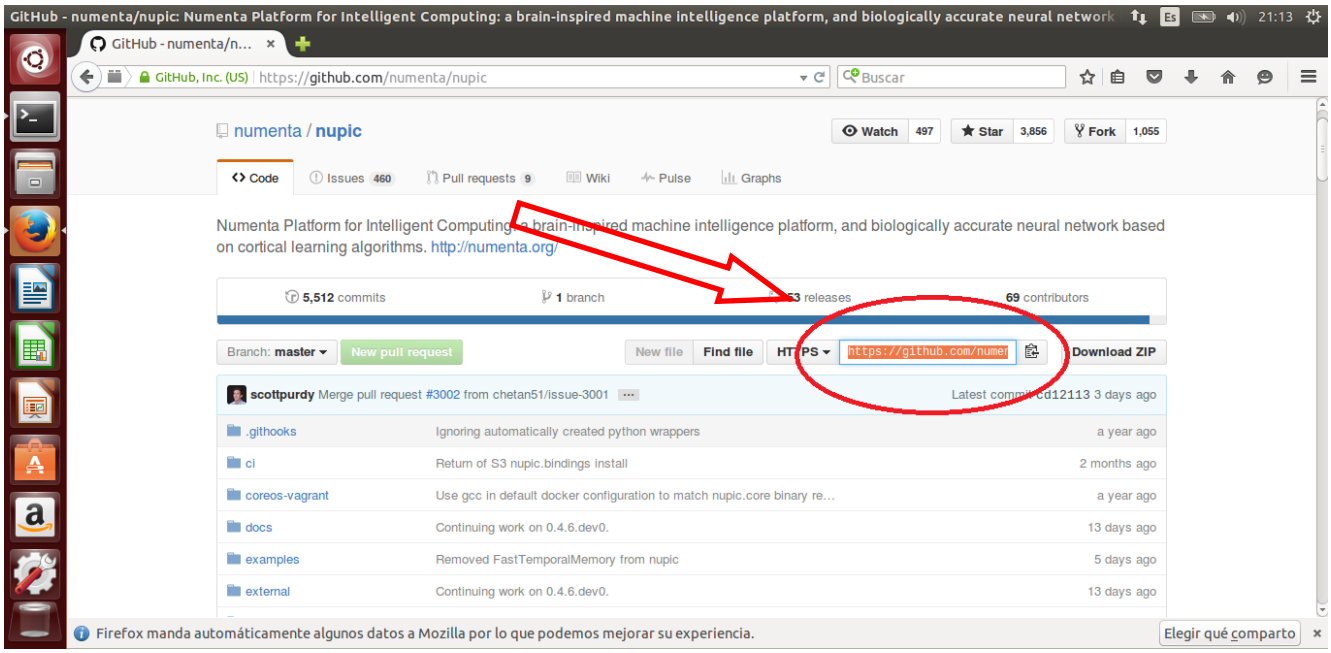

Figura 3-1 Localización de la dirección del directorio de NuPIC dentro de su página en GitHub [42]

```
sudo apt-get install python-pytest
```

Entramos en el fichero *nupic* y nos aseguramos de que la variable de entorno apunta al directorio *nupic*. A continuación, ejecutamos los test contenidos en el fichero *scripts*. Si añadimos -u al final será una prueba unitaria o de unidad, si añadimos -i será de integración y si añadimos –w será *swarming*. Los tests deben de ser pasados o saltados:

```
cd nupic

export NUPIC=`pwd`

cd $NUPIC

./scripts/run_nupic_tests -u
```

Una vez ejecutado el resultado será algo similar a la Figura 3-2.

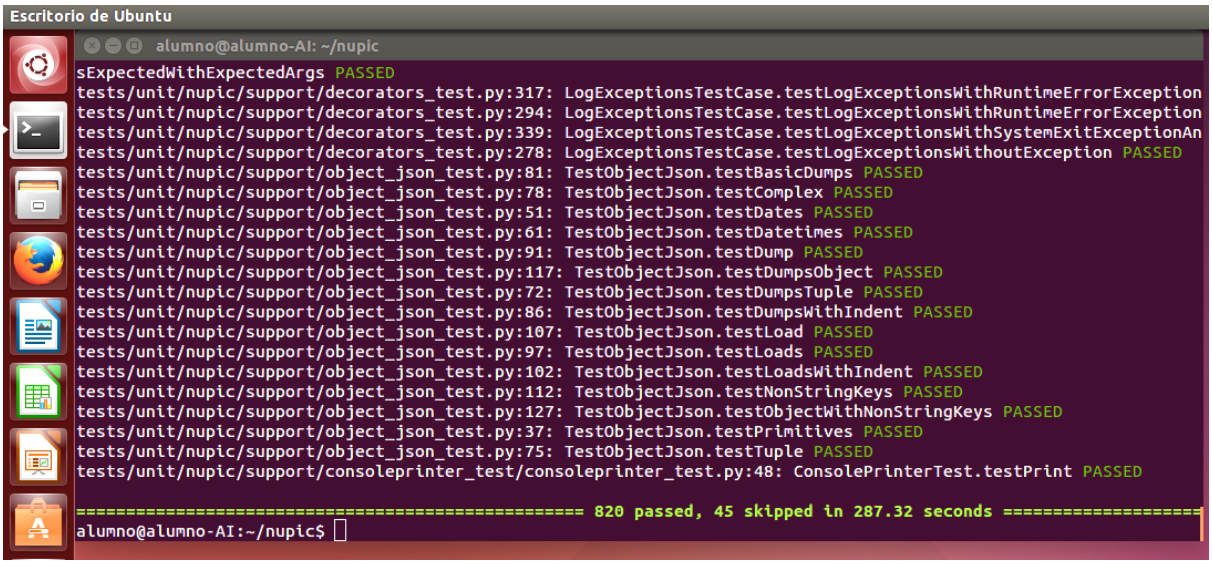

Figura 3-2 Captura de pantalla de los resultados del test de unidad

Existen otros métodos para la instalación de NuPIC también descritos en su Wiki en GitHub [39]. Además, en GitHub también se dispone del código fuente por si se desea colaborar en el desarrollo de NuPIC.

Para poder hacer uso de *Geospatial Tracking* o de la aplicación desarrollada es necesario descargar el resto de requerimientos de Python para *nupic.geospatial*. Para ello, tras haber clonado su directorio de GitHub en nuestro ordenador, ejecutamos el siguiente comando:

```
git clone https://github.com/numenta/nupic.geospatial.git

pip install -r requirements.txt
```

Todo el proceso es susceptible de requerir algún tipo de actualización o modificación requerido por Ubuntu durante el mismo.

3.2 Adquisición de datos

Para la adquisición de datos se ha hecho uso de la *API for AIS Data* de Marine Traffic [43]. Este servicio se ha facilitado de manera gratuita por Marine Traffic al encontrarse este trabajo dentro de la *MarineTraffic Research Network*, un servicio de apoyo a los estudios académicos en el sector marítimo. *API for AIS Data* permite obtener cada dos minutos un mensaje AIS simple (Número de Identificación del Servicio Móvil Marítimo (MMSI, *Maritime Mobile Service Identity*), latitud, longitud, velocidad, proa, rumbo, estado e instante de tiempo) y uno extendido (MMSI, latitud, longitud, velocidad, rumbo, instante de tiempo, 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) cada hora.

Para realizar la descarga de los datos se ha utilizado el servidor Malaspina perteneciente al Centro Universitario de la Defensa de Marín, al que se ha accedido por SSH (Intérprete de Órdenes Seguro o *Secure Shell*). Se ha creado el siguiente script:

```
wget http://services.marinetraffic.com/api/exportvessels/106833382f96e788ba3e6d9c1b
1de719a31fa215/timestamp:15/protocol:csv -O /home/alumnos/careper/temp.csv

sed -i -e '$a\' /home/alumnos/careper/temp.csv

cat /home/alumnos/careper/temp.csv >> /home/alumnos/careper/salida.csv
```

A continuación se le conceden permisos de ejecución al fichero:

```
chmod +x descarga.sh
```

Para automatizar la descarga se añade la tarea al cron, un administrador regular de procesos en segundo plano, de acuerdo a la Figura3-3:

```
crontab -e
```

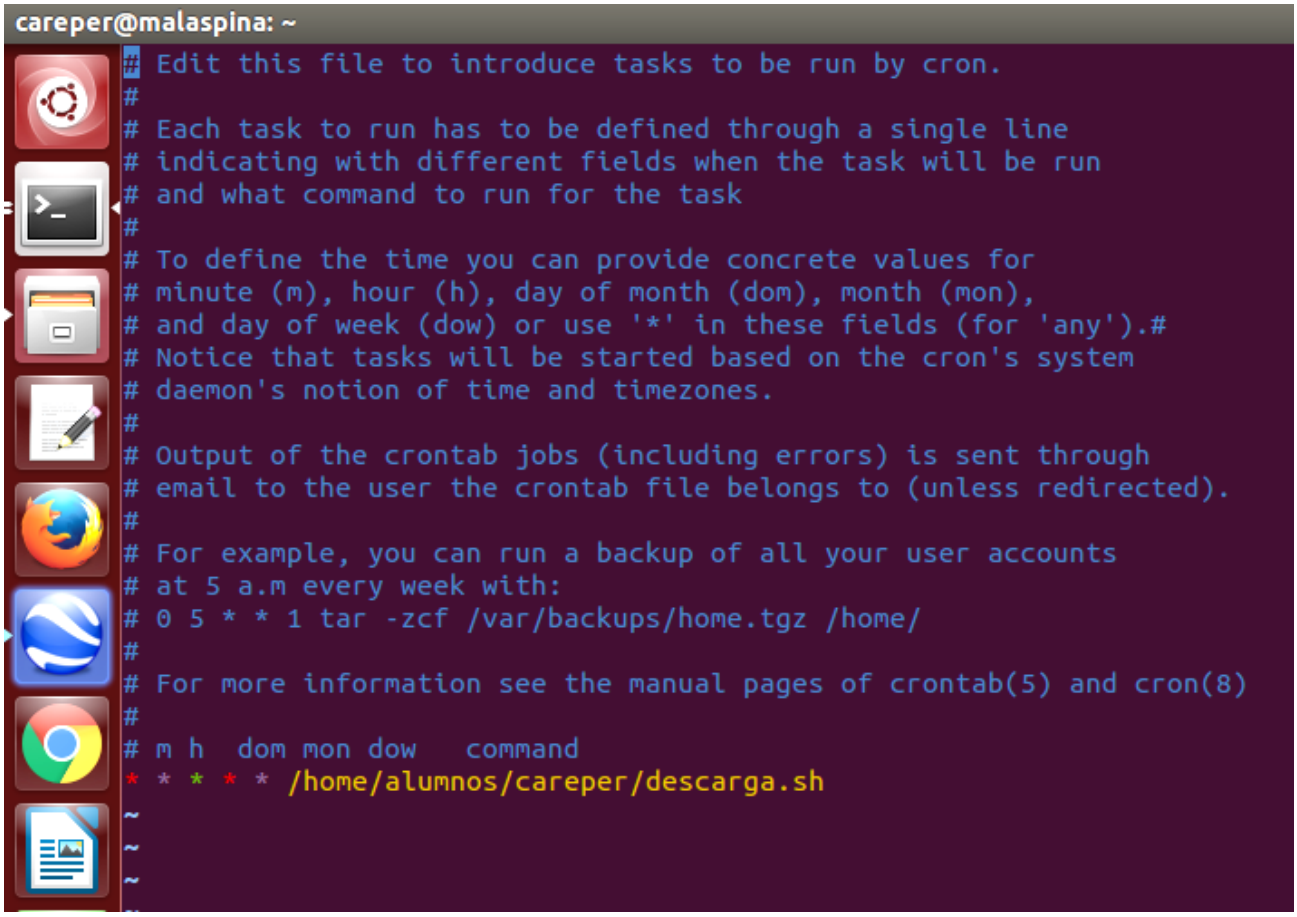

Figura 3-3 Introduciendo descarga.sh en el cron

Posteriormente se han descargado los datos del servidor mediante el comando:

```
scp careper@labs.cud.uvigo.es:/home/alumnos/careper/salida* .
```

```
alumno@alumno-AI:~$ scp careper@labs.cud.uvigo.es:/home/alumnos/careper/salida.c
sv .
careper@labs.cud.uvigo.es's password:
salida.csv                                    100%  708     0.7KB/s   00:00
```

Figura 3-4 Ejemplo de descarga de datos desde el servidor Malaspina

3.3 Ejecución del programa

A continuación se desgrana el programa desarrollado y se explica el funcionamiento de los principales módulos que lo componen. En la Figura 3-5 se muestra un esquema del funcionamiento de la aplicación, así como la relación entre los diferentes módulos y los archivos generados.

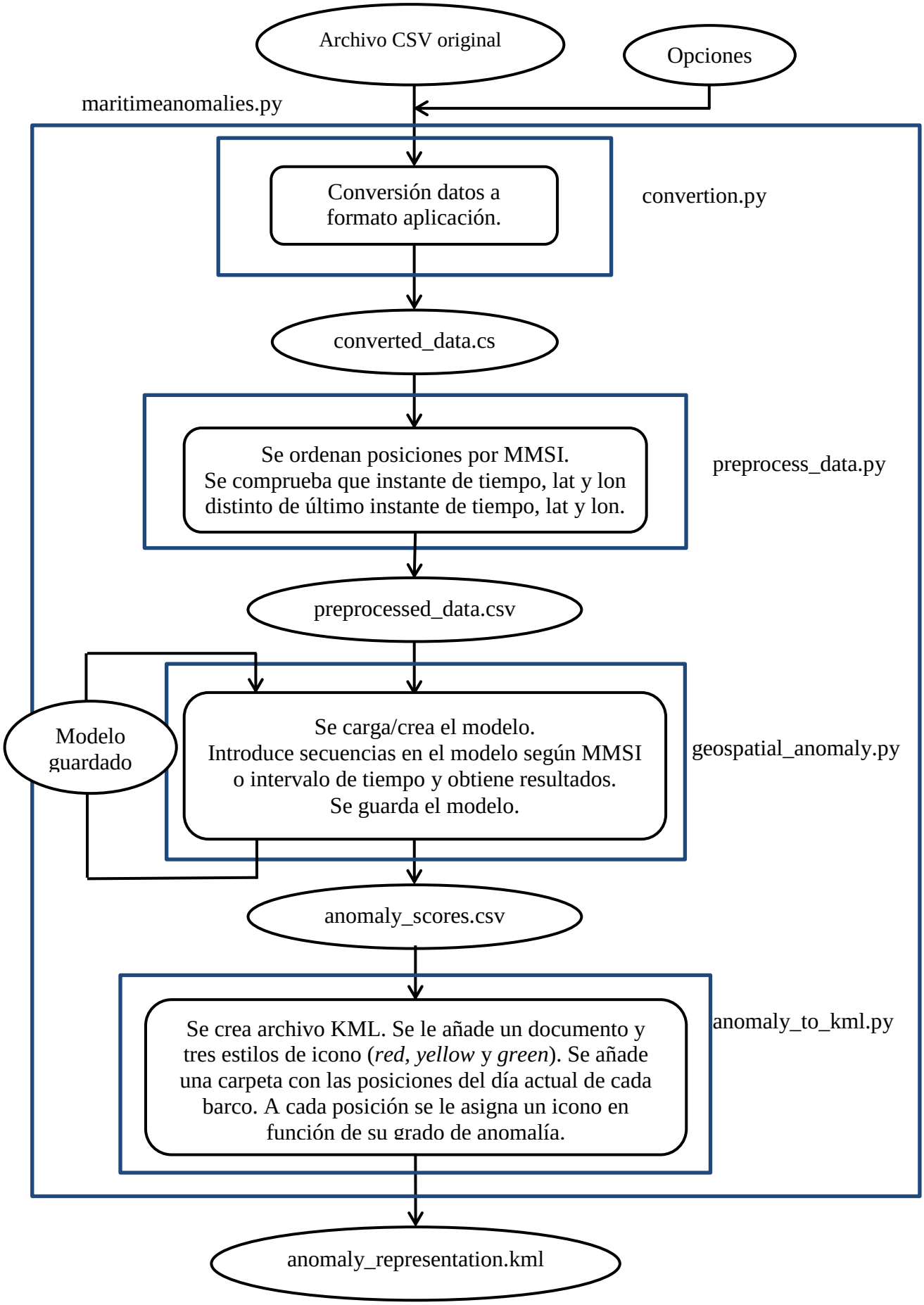

Figura 3-5 Esquema de funcionamiento de la aplicación

3.3.1 Llamar al programa

Para poder llamar a la aplicación debemos de encontrarnos en su directorio. En primer lugar se ha de indicar que la aplicación va a ser abierta en Python. A continuación se introduce el nombre de la aplicación y el archivo de entrada. Por último, existe la posibilidad de añadir una serie de opciones a través de los comandos indicados en la Tabla 3-1, entre las que se encuentran: *help* (muestra un mensaje de ayuda y se cierra), *manual sequence* (divide las rutas por nombre, en vez de hacerlo por saltos temporales), *time encoders* (añade el codificador "momento del día" a los *model params*), *verbose* (muestra las líneas de *debbuging*), *output-dir* (selecciona el directorio de salida), *scale* (si no se indica toma 5m por defecto), *initial* (indica si ha de iniciar un nuevo modelo). El código quedaría de la siguiente manera:

```
python maritimeanomalies.py archivo_de_entrada.csv [-h –i -m -t -v -o outputdir -s scale]
```

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*
Initial	-i, --initial

Tabla 3-1 Correspondencia de opciones con comandos

Este grupo de comandos ejecuta el módulo *maritimeanomalies.py*, cuyo código se encuentra en el Anexo V. Este módulo importa las librerías de Python *os* y *sys*, así como las funciones *OptionParser* de la librería *optparse*, *run* del módulo *run.py*, *convertion* del módulo *convertion.py* y *anomalyrepresentation* del módulo *anomaly_to_kml.py*. A continuación ejecuta la función *maritimeanomalies* con el archivo CSV que se haya indicado como entrada. Acepta también como entrada cualquiera de las opciones de la Tabla 3-1 mediante la función *OptionParser*. *maritimeanomalies* en primer lugar define el fichero de salida. Si no ha sido indicado toma *output* por defecto y si no existe lo crea. A continuación llama a la función *convertion* del módulo *convertion.py*, cuyo código se detalla en el Anexo VII, para convertir el archivo de entrada al formato requerido por la aplicación. Una vez convertido el formato, llama a la función *run* del módulo *run.py*, cuyo código se encuentra en el Anexo VI, que pone en funcionamiento el sistema HTM y devuelve un nuevo archivo CSV, denominado *anomaly_scores.csv*, en el que quedan registrados los grados de anomalía correspondientes a cada posición. Por último, llama a la función *anomalyrepresentation* del módulo *anomaly_to_kml.py*, que recibe como entrada *anomaly_scores.csv* y devuelve un archivo en KML para la representación gráfica de los resultados obtenidos.

3.3.2 Conversión del archivo de entrada

Los datos son recibidos de la API en un archivo CSV con el formato indicado en la documentación aportada por Marine Traffic [43]. Al ser llamada la función *convertion* por *marineanomalies* se ejecuta el módulo *convertion.py*, cuyo código se encuentra en el Anexo VII, que devuelve un archivo CSV con el formato requerido por la aplicación. Este módulo está formado por cuatro funciones: *readais*, *aisconvertion*, *writeais* y *convertion*. Tras importar la librerías de Python *csv*, *time* y *calendar*, se llama a la función *convertion* con los archivos de origen y de salida como entrada. Esta función llama a *readais* con el archivo de origen como entrada. *readais* recorre el archivo almacenando su información

en una lista de listas. Al finalizar devuelve esta lista. A continuación se llama a la función *aisconvertion,* que crea una nueva lista en la que almacena los datos relevantes de la lista inicial con las unidades y el orden requerido por la aplicación y devuelve la nueva lista. Por último se llama a la función *writeais*, introduciendo como entrada el nuevo archivo y la nueva lista. Esta función escribe la nueva lista en el archivo *converted_data.csv* (extracto de un ejemplo en el Anexo XIII).

3.3.3 Ejecución sistema HTM

Cuando se llama a la función *run*, incluida en el módulo *run.py*, cuyo código se encuentra en el Anexo VI, se ejecuta este módulo. En primer lugar se importan las librerías *os* y *sys*, además de las funciones *preprocess,* del módulo *preprocess_data.py,* y *runGeospatialAnomaly,* del módulo *geospatialanomaly.py*. A continuación, se ejecuta la función *run* con el archivo de origen y de salida como entrada, así como las opciones de la Tabla 3-1 introducidas. Esta llama a la función *preprocess*, que realiza el preprocesado y filtrado de los datos a utilizar por el sistema HTM. Una vez se poseen los datos preprocesados se ejecuta la función *runGeospatialAnomaly*. Esta función recibe los datos preprocesados y devuelve el grado de anomalía correspondiente a cada posición.

3.3.4 Preprocesado de los datos

El preprocesado de los datos lo realiza el módulo *preprocess_data.py*, del que se ha incluido el código en el Anexo VIII, al ser llamada la función *preprocess* en el módulo *run.py* con el archivo de origen y de destino como entrada. En este preprocesado, tras importar las librerías *csv, datetime* y *sys,* se ordenan los datos de entrada por buques en función de su MMSI. Además, se eliminan todas aquellas situaciones con un instante de tiempo, latitud o longitud igual que el instante de tiempo, la latitud o la longitud de la posición anterior. Así aseguramos eliminar las posiciones repetidas o aquellas que no se han actualizado correctamente, además de aquellas que se corresponden con buques atracados en puerto. Finalmente, se devuelven los datos preprocesados en el archivo *preprocessed_data.csv* (se ha incluido un extracto de un ejemplo en el Anexo XIV).

3.3.5 Detección de anomalías

El módulo *geospatial_anomaly.py* es el verdadero núcleo de la aplicación. Este es ejecutado a través de la función *runGeospatialAnomaly* desde la función *run.* Este módulo está formado por las funciones *addTimeEncoders, setEncoderScale, createModel* y *runGeospatialAnomalies.* Además, importa las librerías *csv, datetime* y *sys*, la librería *ModelFactory*, de *nupic.frameworks.opf. modelfactory* y el módulo *model_params.py*. Al ejecutar la función *runGeospatialAnomalies* con los archivos de origen y salida como entrada, en caso de haber añadido la opción *initial* de la Tabla 3-1, esta ejecuta en primer lugar la función *createModel*, que, con los parámetros aportados por *model_params.py* y en función de las opciones seleccionadas (Tabla 3-1), crea el modelo de CLA utilizado para encontrar los patrones de movimiento y detectar las anomalías. En caso de no habérsele indicado que cree un modelo, cargará el modelo guardado.

A continuación, en función de si se ha seleccionado la creación de secuencias manual o automática (Tabla 3-1), divide las secuencias según el nombre del barco o según el intervalo de tiempo respectivamente. Cada vez que se finaliza una "ruta" se resetea la secuencia. Se introducen los datos de cada posición en el modelo y este devuelve el nivel de anomalía. Este se escribe, junto con un indicador de si la posición inicia una nueva secuencia y el resto de los datos, en un nuevo archivo: *anomaly_scores.csv* (se dispone de un extracto de un ejemplo en el Anexo XV). Por último, se guarda el modelo actual en el fichero *model.*

De esta manera, una vez que se ha generado un modelo, los patrones generados por este se almacenan y al introducir una nueva secuencia de datos se hace uso de ellos. Sin embargo, no se puede decir que exista una fase de aprendizaje y otra de análisis, ambas van unidas. Para cada nueva posición el modelo genera una previsión en función a los datos pasados y, dependiendo de si coincide o no con

la posición real, devuelve el grado de anomalía. El aprendizaje es dinámico y se realiza durante todo el proceso.

3.3.6 Presentación gráfica de los resultados

Para la presentación de los resultados se ha optado por utilizar archivos KML y la aplicación Google Earth. Para ello se ha generado un módulo en Python, *anomaly_to_kml.py*, cuyo código se encuentra en el Anexo XI, derivado del código propuesto por Google en [44]. A partir de la información contenida en el archivo *anomaly_scores.csv* generado por la aplicación, *anomaly_to_kml*.py crea un archivo KML denominado *anomaly_representation.kml* (se dispone de un extracto de un ejemplo en el Anexo XVI). Este archivo contiene un punto para cada posición analizada perteneciente al día en que se realice el análisis. Al ser cargado en Google Earth, cada punto es representado por un icono que varía su color en función del grado de anomalía correspondiente a esa posición.

El módulo *anomaly_to_kml.py* utiliza las librerías *csv, xml.dom.minidom, sys, time, datetime* y *calendar*. Está compuesto por las funciones *extractCoordinates, createPlacemark, createKML* y *anomalyrepresentation*. Al ser llamada *anomalyrepresentation* desde *maritimeanomalies* con los archivos de origen y salida como entrada, este carga los datos contenidos en el archivo de origen y llama a *createKML*. Esta función genera el archivo KML y escribe en él la cabecera del archivo. A continuación, crea un documento y, dentro de este, tres estilos de icono: *green, yellow* y *red*. Prosigue comprobando si las posiciones obtenidas del archivo de origen son del día en el que se encuentra y, en caso positivo, crea un nuevo fichero para cada barco. En este fichero se genera un *Placemark* para cada posición de ese barco tomada en el día en el que se encuentra. Esto lo hace mediante la función *createPlacemark*. Esta función crea para la posición en cuestión un *Placemark* en el que incluye la información referente a esa posición (*trackName, timestamp, longitude, latitude, speed, anomaly_score* y *new_secuence*), la posición geográfica (latitud y longitud) y el tipo de icono con el que se señalará la posición: rojo si el nivel de anomalía es mayor a 0.5, amarillo si es menor de 0.5 y mayor de 0.25 y verde si es menor de 0.25.

3.4 Selección de los parámetros del modelo

Los modelos CLA disponen de un elevado número de parámetros variables que describen el tamaño y las conexiones de la red cortical. Sin embargo, se ha decidido utilizar los valores originales de *Geospatial Tracking*, realizando modificaciones en los parámetros más significativos para comprobar cómo esta variación afecta en la respuesta del sistema. Los parámetros que se han decidido modificar son los siguientes: *spatial pooler* (SP); *temporal pooler* (TP); las dimensiones del vector SDR *u* generado por el codificador, siendo *n* el número de bits dedicados al vector y *w* el número de bits activo en cada momento; número de columnas activas por área de inhibición; número de células por columna; nueva cuenta de sinapsis; máximas sinapsis por segmento y máximo número de segmentos por células.

La selección de los parámetros del modelo se ha realizado de manera empírica, testando diferentes combinaciones de valores y analizando cual ofrecía una mayor convergencia del modelo, es decir, posee una mayor capacidad para localizar los patrones espaciales y temporales que rigen los diferentes escenarios. Para calcular la convergencia del modelo se han calculado los ratios de anomalía (nº de anomalías / nº de posiciones introducidas), considerándose anomalía toda aquella que poseyese una *anomaly_score* mayor de 0.5.

Para realizar las pruebas para la selección de los parámetros del modelo se han utilizado las posiciones del buque Volcán de Tindaya, cuyos datos se detallan en la Figura 3-6, en su tránsito de Fuerteventura a Lanzarote los días 22 y 23 de febrero de 2016.

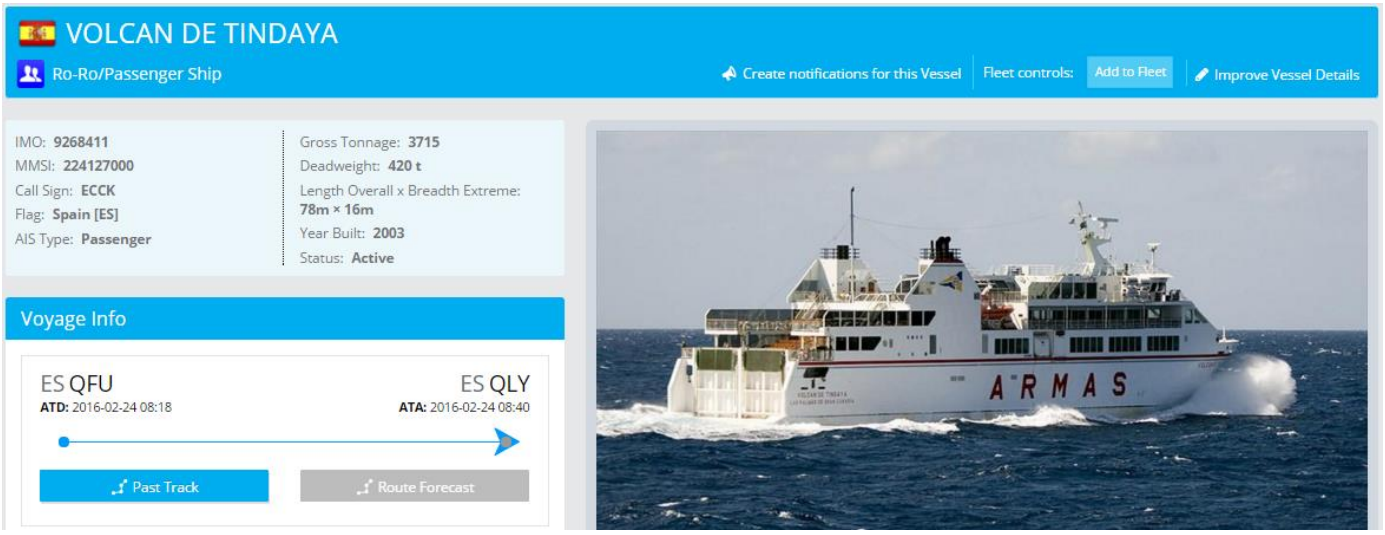

Figura 3-6 Ficha del Volcán de Tindaya en Marine Traffic [45]

Para ello, se ejecuta el sistema sin introducir la etiqueta –m, para que las secuencias sean temporales, y con la etiqueta -s seguida de la escala deseada. Se van modificando a su vez los parámetros del módulo *model_params.py*, obteniendo los resultados indicados en la Tabla 3-2. Comparando los resultados se puede afirmar que, entre los parámetros que se han testado, los mejores resultados de convergencia se logran seleccionando TP = SP = 2048 y u: n = 4096 y w = 101. A continuación se aumentan *numActiveColumnsPerInhArea* a 40, *cellsPerColumn* a 64, *newSynapseCount* a 40, *maxSynapsesPerSegment* a 64 y *maxSegmentsPerCell* a 276, obteniendo los resultados indicados en la Tabla 3-3. Al observarse un decremento en el ratio de anomalías, se mantiene la modificación realizada.

TP = SP = 2048				***TP = SP = 4096***			
Escala	**Nº de posiciones**	**Nº de anomalías**	**Ratio de anomalías (a > 0.5)**	**Escala**	**Nº de posiciones**	**Nº de anomalías**	**Ratio de anomalías (a > 0.5)**
u: n=2048 w=51				**u: n=2048 w=51**			
200	226	48	0,212389381	200	228	62	0,27192982
150	221	63	0,285067873	150	224	90	0,40178571
100	221	106	0,479638009	100	225	143	0,63555556
50	220	190	0,863636364	50	226	201	0,88938053
10	234	233	0,995726496	10	234	234	1
u: n=4096 w=51				**u: n=4096 w=51**			
200	223	46	0,206278027	200	223	61	0,2735426
150	219	48	0,219178082	150	224	84	0,375
100	220	68	0,309090909	100	219	139	0,6347032
50	229	180	0,786026201	50	229	200	0,87336245
10	230	227	0,986956522	10	234	234	1
u: n=4096 w=101				**u: n=4096 w=101**			
200	216	41	0,189814815	200	215	48	0,22325581
150	216	46	0,212962963	150	215	60	0,27906977
100	221	79	0,357466063	100	211	98	0,46445498
50	223	169	0,757847534	50	218	179	0,82110092
10	230	230	1	10	234	234	1

Tabla 3-2 Ratio de anomalías según parámetros. Primera modificación

Escala	Nº de posiciones	Nº de anomalías	Ratio de anomalías (a > 0.5)
200	219	39	0,18
150	220	44	0,2
100	218	51	0,23
50	219	139	0,63
10	229	227	0,99

Tabla 3-3 Ratio de anomalías según parámetros. Segunda modificación

Una vez identificada la mejor combinación de parámetros, se realizan pruebas empíricas con los datos del ferry y de la zona de Finisterre para encontrar la escala óptima. Para ello se utilizan las posiciones del buque Volcán de Tindaya (Figura 3-6) en su tránsito de Fuerteventura a Lanzarote los días 22 y 23 de febrero del 2016 y las posiciones de los buques dirigiéndose hacia el norte entre los días 20 y 21 de febrero del 2016 a través del DST de Finisterre. En el caso que concierne al Volcán de Tindaya (Tabla 3-4) se puede apreciar en los resultados presentados en la Figura 3-7 cómo el ratio de anomalías se dispara a partir de una escala inferior a 100. Por esta razón se utilizará este valor para el resto de pruebas que se realicen para este barco. Los resultados en el DST de Finisterre (Tabla 3-5), sin embargo, no presentan un punto de inflexión tan claro como en el caso anterior. Atendiendo a la Figura 3-8 se ha decidido utilizar un rango de 200.

Al comparar la Tabla 3-4 y la Tabla 3-5 se observa que en el caso del Volcán de Tindaya, con una escala menor y un menor número de posiciones, se logra un menor ratio. Así se demuestra que el rendimiento de la aplicación es mayor al crear un modelo para un único contacto, si existe esta posibilidad.

Escala	Nº de posiciones	Nº de anomalías	Ratio de anomalías (a > 0.5)
200	388	47	0,121
175	378	54	0,143
150	378	51	0,135
125	375	55	0,147
100	378	60	0,159
75	377	106	0,281
50	380	170	0,447
25	390	294	0,754
10	393	373	0,949

Tabla 3-4 Ratio de anomalías del Volcán de Tindaya según la escala seleccionada

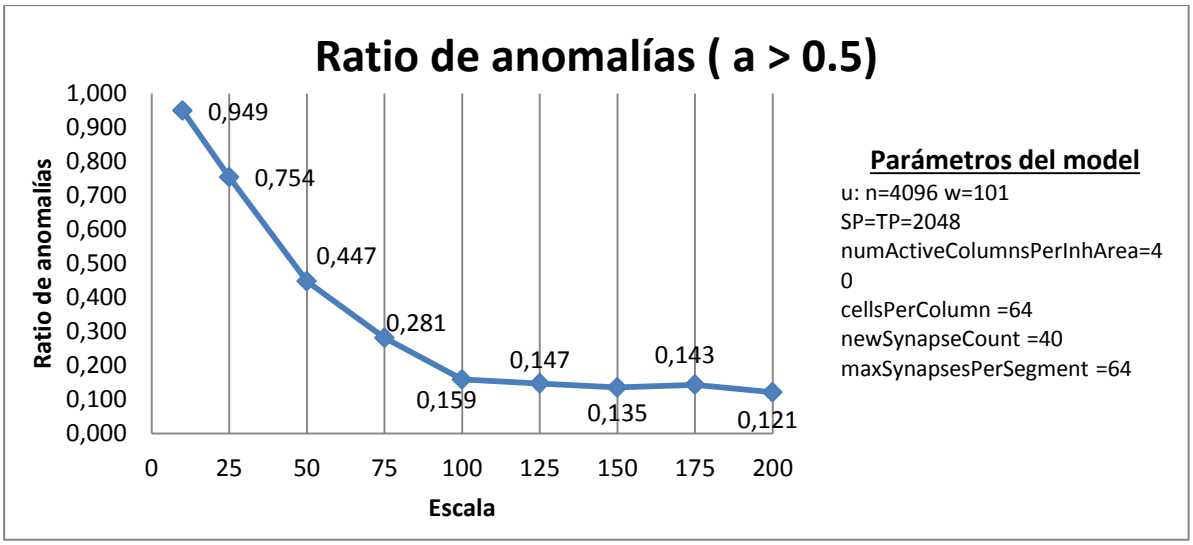

Figura 3-7 Relación del ratio de anomalías del Volcán de Tindaya y la escala seleccionada

Escala	Nº de posiciones	Nº anomalías	Ratio de anomalías (a > 0.5)
500	7323	653	0,089
400	7252	909	0,125
300	7170	1449	0,202
250	7091	1832	0,258
200	7078	2661	0,376
100	7063	5287	0,749

Tabla 3-5 Ratio de anomalías en el DST de Finisterre según la escala seleccionada

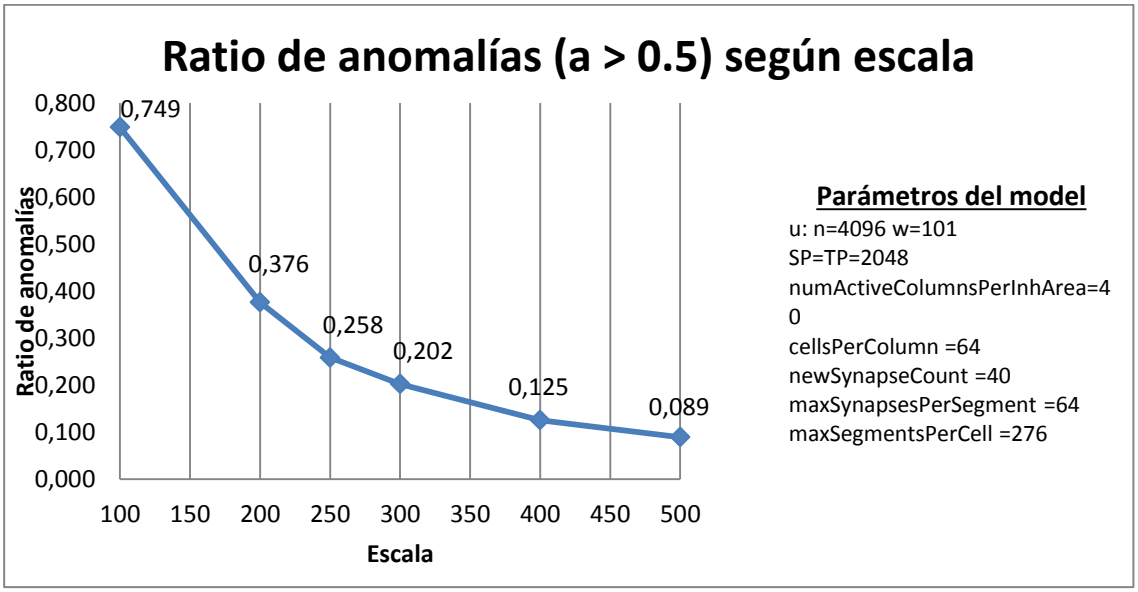

Figura 3-8 Relación del ratio de anomalías en el DST de Finisterre y la escala seleccionada

4 EVALUACIÓN DE LA APLICACIÓN

4.1 Verificación por anomalías sintéticas

Para evaluar la aplicación, debido a la carencia de anomalías reales conocidas en los escenarios de estudio, se ha decidido generar trazas anómalas de manera sintética. De esta manera se puede controlar la propiedad del contacto que lo convierte en anómalo y comprobar la respuesta de la aplicación a anomalías de un carácter determinado. Se han seleccionado dos escenarios para realizar la evaluación: uno de amplia extensión con un tráfico elevado, pero ordenado, y otro de extensión reducida con un único barco que realiza una ruta similar repetidamente. En el primer caso, como ya se indicó en el Apartado 1.1.3, se ha seleccionado el DST de Finisterre. En el segundo caso se ha decidido monitorizar los tránsitos del ferry Volcán de Tindaya, cuyos datos se encuentran recogidos en la Figura 3-6.

4.1.1 DST de Finisterre

Para la evaluación en el DST de Finisterre el estudio se ha centrado en las dos vías más orientales, las que tienen un sentido de circulación norte. Para ello, se ha realizado un filtrado de las posiciones recibidas por rumbo, seleccionando todas aquellas cuyo rumbo oscilase entre norte y este (000 – 090) o entre oeste y norte (270 – 360). Las simulaciones parten de un modelo con los parámetros indicados en el Apartado 3.4 (TP = SP = 2048, u: n = 4096 y w = 101, *numActiveColumnsPerInhArea* = 40, *cellsPerColumn* = 64, *newSynapseCount* = 40, *maxSynapsesPerSegment* = 64, *maxSegmentsPerCell* = 276 y escala = 200. En el modelo se han introducido datos reales de buques entre los días 16 y 25 de febrero del 2016.

Se han diseñado cinco anomalías, cada una de ellas para comprobar la respuesta de la aplicación a un suceso diferente. En la primera anomalía, el buque se encuentra navegando con rumbo oeste, es decir, en perpendicular al sentido del tráfico. Así pues, tenemos dos anomalías en el mismo caso. En primer lugar, el rumbo es anómalo durante todo el recorrido. En segundo lugar, el buque termina abandonando la vía a mitad de su navegación, entrando en la zona de separación.

Como se puede observar en la Figura 4-1, la aplicación ha señalado una anomalía cuando el barco se ha aproximado y rebasado a línea divisoria entre la vía y la zona de separación. Sin embargo, en el resto de la travesía no detecta ningún tipo de anomalía, a excepción de las dos posiciones de inicio, que como se podrá comprobar más adelante, es una falsa alarma entregada por el sistema en todos los casos.

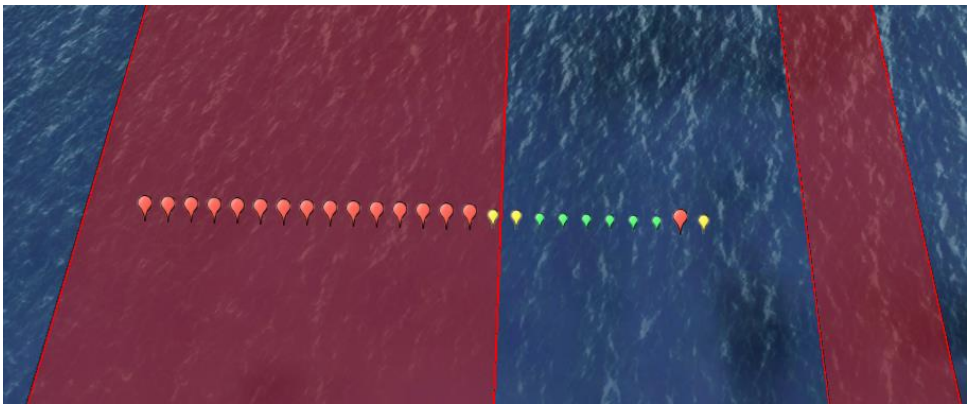

Figura 4-1 Resultado mostrado por la aplicación para la primera anomalía

En la segunda anomalía se ha generado un cambio de rumbo brusco e improcedente. De esta manera se vuelven a generar dos anomalías diferentes, el tránsito a través de una zona de separación y el cambio de rumbo.

Al igual que en el caso anterior, la aplicación detecta una anomalía al salir de la vía y al iniciar el tránsito. Detecta también una anomalía a mitad del primer tramo, pero se trata de una falsa alarma. Sin embargo, no identifica el brusco cambio de rumbo como una conducta anómala.

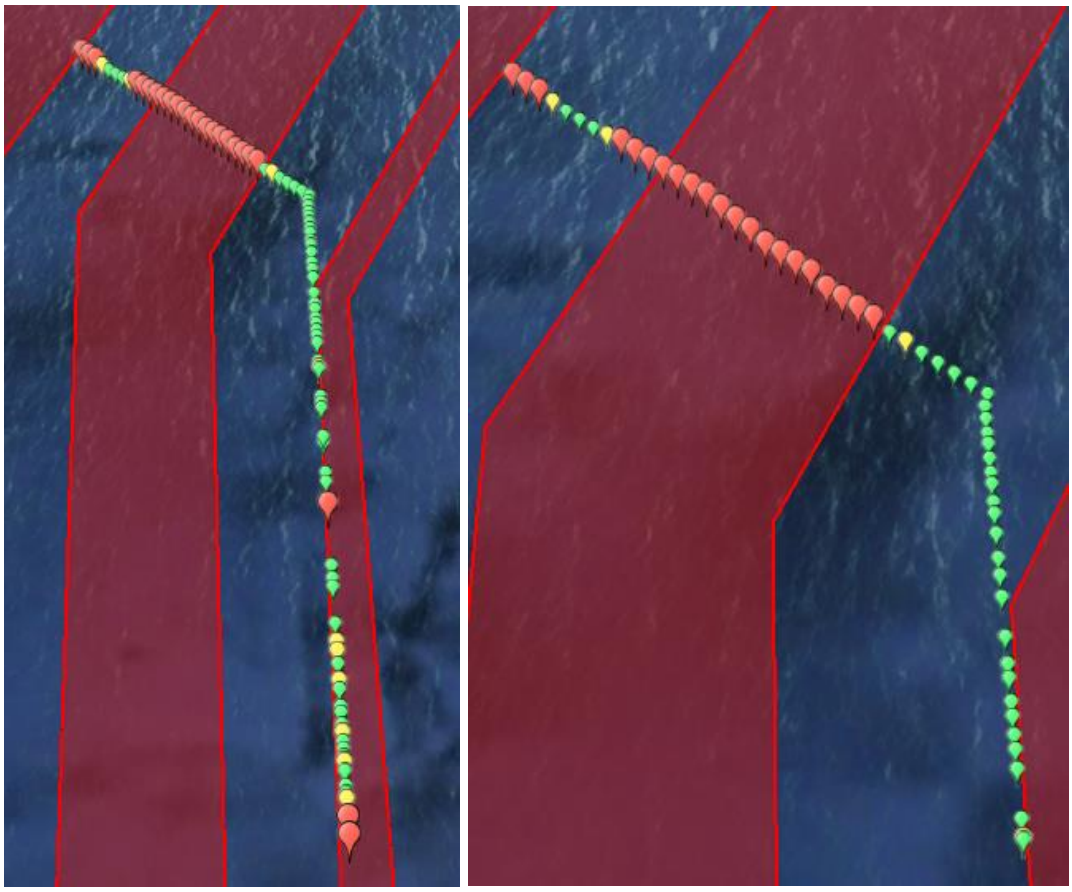

Figura 4-2 Resultado mostrado por la aplicación para la segunda anomalía

En la tercera simulación el barco incrementa su velocidad hasta los treinta nudos durante diez minutos (zona indicada en la Figura 4-3). Como se puede apreciar en la Figura 4-3, el sistema no identifica el incremento en la velocidad. Sí que indica, sin embargo, las falsas alarmas señaladas en los puntos anteriores.

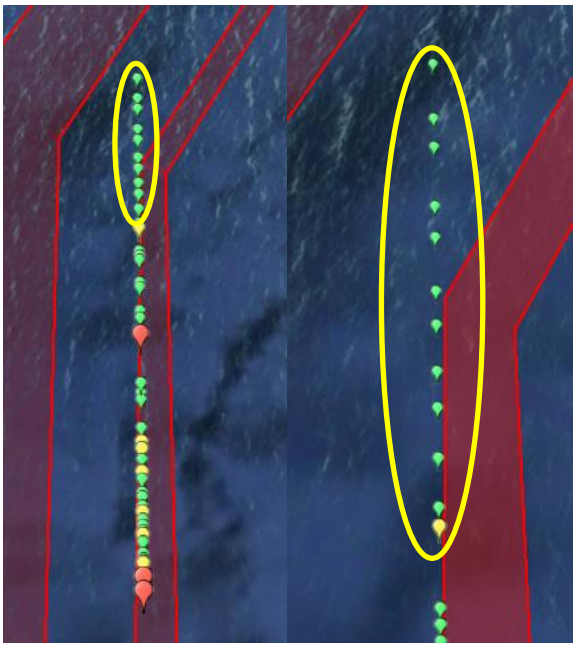

Figura 4-3 Resultado mostrado por la aplicación para la tercera anomalía. Rodeada en amarillo la zona de mayor velocidad

En el caso de la cuarta simulación existen también dos anomalías. La primera de ellas consiste en un rumbo constante en una zona donde el tráfico general realiza un giro a estribor (hacia la derecha). La segunda es una reducción de la velocidad a diez nudos durante un tiempo reducido (en la zona señalada en la Figura 4-4).

Como se puede apreciar en la Figura 4-4, la aplicación no señala una anomalía para ninguna de las dos situaciones anómalas señaladas anteriormente. Sí la señala, sin embargo, para los dos casos ya indicados en la simulación anterior: las dos primeras posiciones y la falsa alarma a mitad del primer tramo de la vía.

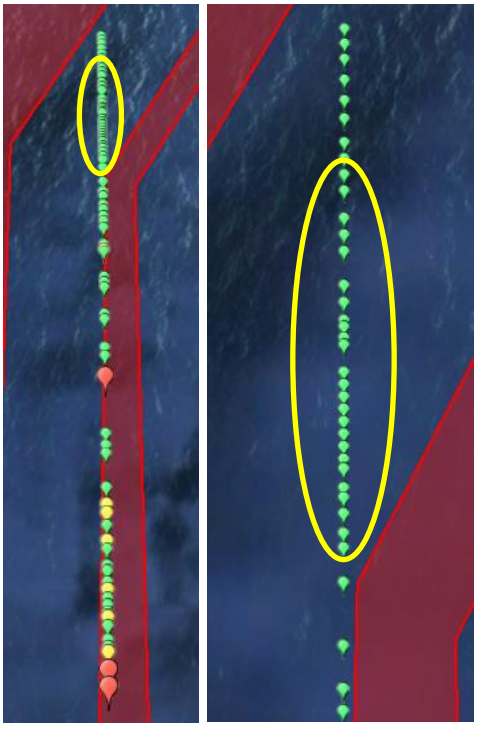

Figura 4-4 Resultado mostrado por la aplicación para la tercera anomalía. Rodeada en amarillo la zona de menor velocidad

En el quinto y último caso, además de la anomalía referente al cambio de rumbo, el barco ha permanecido detenido durante veinte minutos detenido, tal y como se indica con el círculo amarillo en la Figura 4-5.

El resultado ha sido muy similar al del caso anterior. Se han dado las mismas falsas alarmas que en el caso anterior. A estas hay que sumar una anomalía producida al recuperar el barco su velocidad original después de haber parado. Al igual que el anterior caso, no detecta la anomalía en el no realizado cambio de rumbo. Tampoco detecta como una anomalía la detención del buque durante veinte minutos.

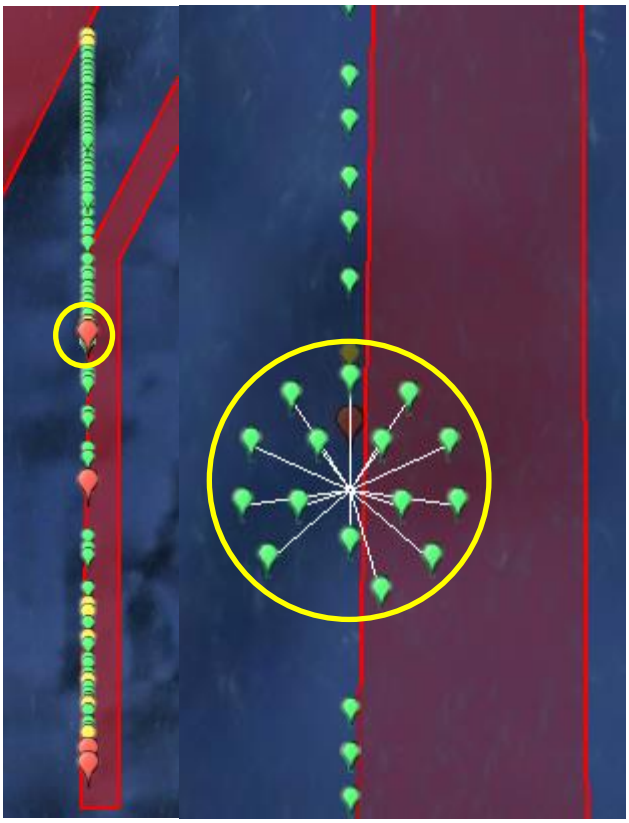

Figura 4-5 Resultado mostrado por la aplicación para la cuarta anomalía. Rodeada en amarillo la posición del buque al detenerse

4.1.2 Volcán de Tindaya

Para la evaluación del barco Volcán de Tindaya, el estudio se ha centrado en su tránsito de Fuerteventura a Lanzarote. Para ello, se ha realizado un filtrado de las posiciones recibidas por rumbo, seleccionando todas aquellas cuyo rumbo oscilase entre norte y este (000 – 090) o entre oeste y norte (270 – 360). Las simulaciones parten de un modelo con los parámetros indicados en el Apartado 3.4 (TP = SP = 2048, u: n = 4096 y w = 101, *numActiveColumnsPerInhArea* = 40, *cellsPerColumn* = 64, *newSynapseCount* = 40, *maxSynapsesPerSegment* = 64, *maxSegmentsPerCell* = 276 y escala = 100. En el modelo se han introducido datos reales del buque entre los días 14 y 25 de febrero del 2016.

Para este escenario se han generado cinco trayectos con diferentes anomalías. En el primer caso se ha realizado una ruta que se aleja francamente hacia el este de la ruta convencional y que posteriormente regresa al puerto de destino, como se puede apreciar en la Figura 4-6. Como ya se mencionó en el anterior apartado, el sistema da una falsa alarma al iniciarse la ruta. Se puede apreciar que enseguida se detecta que el rumbo es anómalo. Se mantiene así hasta que al final de la ruta vuelve a mostrar un patrón de navegación convencional.

En el segundo caso, la desviación de la ruta es bastante más suave. Se puede apreciar en la Figura 4-6 que, a pesar de ser una irregularidad poco marcada, el sistema la identifica correctamente.

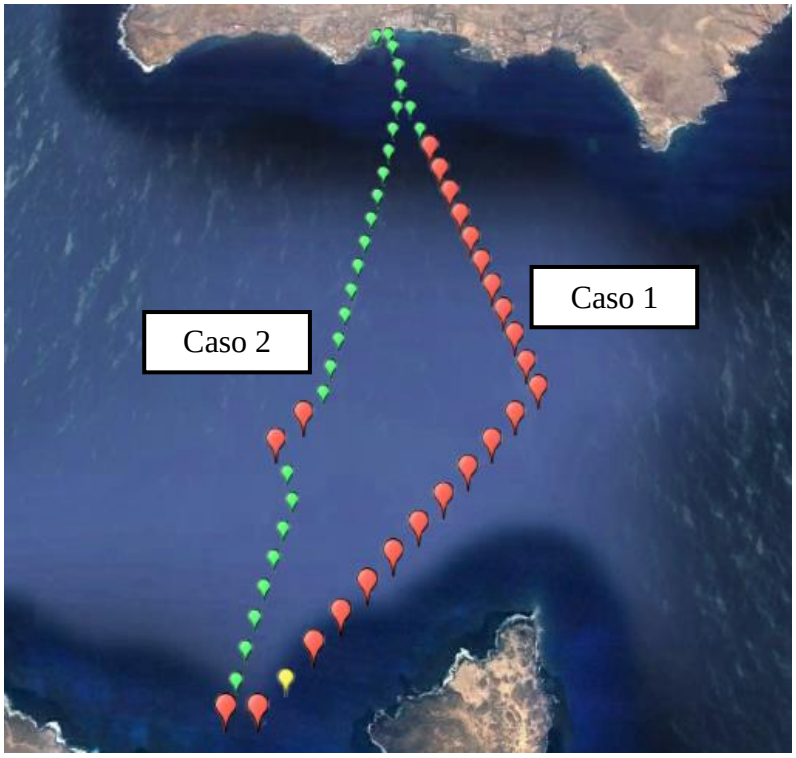

Figura 4-6 Resultado mostrado por la aplicación para los casos 1 y 2

En el tercer caso propuesto, el buque se ha detenido a mitad de la travesía, tal como se puede ver en la Figura 4-7. En la figura se puede apreciar también la falsa alarma inicial. Sin embargo, no presenta ninguna anomalía al quedar detenido el buque.

Figura 4-7 Resultado mostrado por la aplicación para el caso 3

En el cuarto caso el barco ha duplicado su velocidad al alcanzar un tercio del trayecto. En la Figura 4-8 se puede comprobar cómo la aplicación reacciona correctamente al aumento de velocidad, señalándolo como una anomalía.

Figura 4-8 Resultado mostrado por la aplicación para el caso 4

En el quinto y último caso propuesto el barco reduce su velocidad a cinco nudos al haber recorrido dos tercios del trayecto. Como se observa en la Figura 4-9, la aplicación no genera ninguna anomalía debido a la reducción en la velocidad. Sin embargo, sí que reconoce como inesperado el último cambio de rumbo.

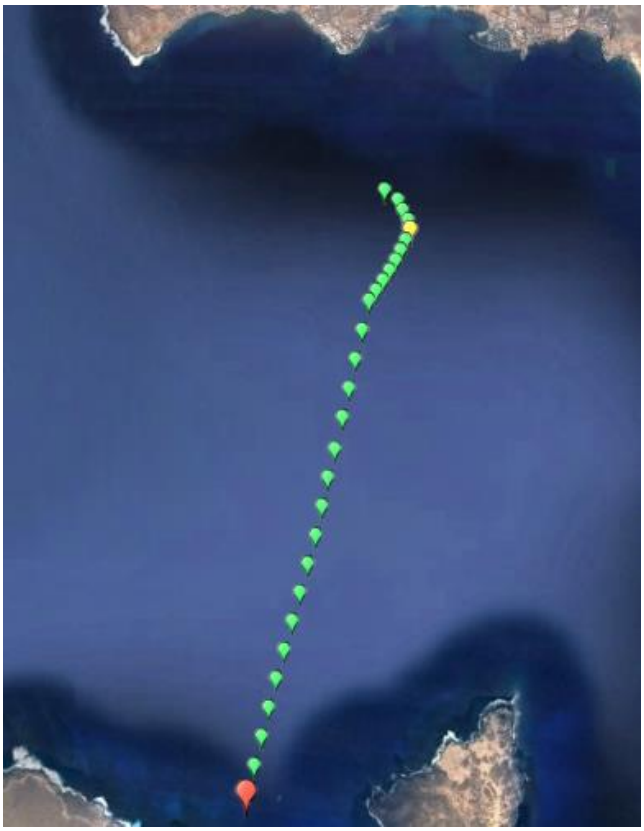

Figura 4-9 Resultado mostrado por la aplicación para el caso 5

4.2 Análisis de los resultados

A pesar de que los datos para testar la aplicación han sido generados sintéticamente y que no cubren el amplio espectro de las anomalías a las que se puede llegar a enfrentar el sistema, son lo suficientemente robustos como para permitir obtener una serie de conclusiones acerca del funcionamiento de la aplicación. Hay que considerar también que para evaluar el sistema se ha realizado previamente un filtrado por rumbo. Esto ha requerido un conocimiento previo mínimo del escenario de estudio.

En primer lugar, resulta evidente el sistema es mucho más eficaz cuando se genera un modelo para un único barco. En el escenario del Volcán de Tindaya el modelo es capaz de converger utilizando una escala menor y un menor número de posiciones. Además, es capaz de detectar una mayor variedad de anomalías y presenta un menor índice de falsa alarma. Esto se debe a que la aplicación parte de *Geospatial Tracking*, diseñada para analizar los desplazamientos de un único sujeto por modelo.

Se aprecia también en todos los casos que se genera una falsa alarma al comienzo de cada ruta. Esto se podría solucionar descartando el primer resultado de cada trayecto.

Otra característica deducible del sistema es su incapacidad para identificar la reducción en la velocidad o la detención como anomalía. Esto es una debilidad evidente si tenemos en cuenta que uno de los primeros síntomas que suele presentar un barco que sufre una emergencia es un descenso en la velocidad. Por el contrario, sí que asocia con una anomalía los aumentos de velocidad cuando el modelo está aplicado a un único buque. El caso de la reducción de velocidad puede deberse al valor de la escala.

En el caso de las anomalías relacionadas con el rumbo, ya sean producidas por un rumbo atípico o una variación brusca y no previsible de este, el sistema no es capaz de detectarlas cuando se aplica a un área amplia y con un tráfico alto. Sin embargo, sí que es capaz de detectarlas con precisión cuando existe un único barco para cada modelo.

En los dos escenarios propuestos la aplicación identificó de manera eficaz las anomalías relacionadas con el posicionamiento geográfico. En todos los casos se generó una anomalía cada vez que el barco salía de las rutas marítimas convencionales. Es preciso destacar que no se ha introducido ninguna información en el modelo de cuáles son las vías de separación del DST, sino que lo ha aprendido automáticamente a partir de las posiciones de los barcos.

Así pues, se puede concluir que la aplicación ha demostrado su capacidad para la detección de anomalías. Bien es cierto que esta se ve mermada al utilizar un único modelo para un elevado número de barcos. También hay que tener en cuenta que, aunque es capaz de identificar anomalías de distintas naturalezas, existen algunos tipos que no son detectados por el sistema, lo que podría suponer una vulnerabilidad si no se cubren de alguna manera.

La disminución en la capacidad del sistema al aumentar la entidad del escenario de estudio demuestra que algunos de los errores detectados no se deben a un funcionamiento incorrecto del sistema, sino a un dimensionamiento insuficiente.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Revisión de los objetivos

De acuerdo con los objetivos fijados al comienzo del trabajo podemos concluir que estos se han alcanzado satisfactoriamente. Se ha logrado desarrollar una aplicación que, recibiendo datos AIS de la *API for AIS Data* de Marine Traffic y utilizando NuPIC como motor, es capaz de tanto analizar el tráfico marítimo de una zona geográfica como monitorizar el desplazamiento de un buque. La aplicación ha demostrado además su capacidad para detectar anomalías sintéticas en ambos casos. A continuación se enuncian los aspectos positivos y puntos fuertes de esta aplicación frente a otros sistemas de análisis del tráfico marítimo así como las desventajas y debilidades que posee:

- Ventajas y puntos fuertes: aprendizaje autónomo y continuo, no requiere conocimiento previo del tipo de anomalías que pueden existir ni de las características de la zona de estudio, detecta el incumplimiento de normas de circulación sin la necesidad de conocer dichas normas, capacidad para detectar anomalías sutiles, como cambios de rumbo o velocidad, al aplicarse un único buque por modelo y gran margen de mejora.
- Desventajas y puntos débiles: pierde capacidades en áreas extensas y con un número elevado de casos de estudio dentro de un único modelo, no detecta reducciones en la velocidad y no permite fijar criterios de anomalía.

5.2 Conclusiones

Más allá del desarrollo de la aplicación, de este trabajo se pueden obtener cuatro grandes conclusiones en lo referente al tratamiento de los datos del tráfico marítimo y la utilización de sistemas de AI en su gestión y aprovechamiento:

- **Alta disponibilidad de información y datos.** Los mensajes AIS utilizados en este trabajo no son más que una pequeña parte de la gran cantidad de información existente referente a la mar y sus usuarios. El gran número y diversidad de sensores dedicados a la vigilancia marítima, unido a las extensas bases de datos, ofrece un elevadísimo potencial a las diferentes tecnologías de la información o de la AI especializadas en el tratamiento de *Big Data*.
- **Mejora del conocimiento del entorno marítimo (CME) mediante AI.** En la actualidad, una de las mayores limitaciones de la AI es el acceso a una fuente de información lo suficientemente grande. El tratamiento del *Big Data* referente a la mar por parte de sistemas de AI permitirá un mayor entendimiento del entorno marítimo, siendo capaces de detectar patrones y relaciones inaccesibles al análisis humano.

- **Futura herramienta de la Armada para mejorar su MSA.** Esta aplicación abre la puerta a la adopción por parte de la Armada de una herramienta de AI que le permita alcanzar un mayor nivel de CME a partir de los medios y datos que dispone.
- **Idoneidad de este tipo de aplicaciones para el control del tráfico marítimo.** Actualmente la mayoría de VTS carecen de la capacidad de predicción y detección de anomalías, más allá de aquellas determinadas por una serie de parámetros fijos. La aplicación de este tipo de herramientas a VTS y otros sistemas de control del tráfico marítimo les permitiría lograr una predicción temporal del tráfico, facilitando la gestión de recursos, y les ofrecería una detección de anomalías flexible y dinámica que, al combinarse con los métodos ya existentes, lograría una muy elevada precisión.

5.3 Líneas futuras

Este trabajo pretende sentar las bases para un futuro desarrollo de un sistema de vigilancia marítima por AI. A pesar del correcto funcionamiento de la aplicación desarrollada, todavía queda camino que recorrer hasta poseer una aplicación que sea capaz de operar de manera eficaz en un entorno real.

En primer lugar, se recomienda realizar un seguimiento a NuPIC con el fin de conocer los avances de este y poderlos llevar a la aplicación. Se aconseja mantener una participación activa en su comunidad, aportando los progresos logrados, de manera que se consiga un beneficio mutuo.

En cuanto a la propia aplicación, existen varias líneas de mejora. Un hito a lograr es el análisis de datos en tiempo real o inmediato. Los datos utilizados tienen un tiempo de refresco de dos minutos. Sin embargo, la automatización de la aplicación para analizar los datos de entrada cada dos minutos ya supondría un gran avance. Esta capacidad permitiría reducir el tiempo de reacción ante cualquier anomalía. Una posible solución a este problema sería permitir que el sistema aprenda a partir de datos históricos y posteriormente alimentarlo en tiempo real creando un modelo nuevo para cada buque.

Se contempla a su vez la utilización de técnicas *swarm* para la optimización de los parámetros del modelo. Esto permitiría lograr modelos adaptados a cada escenario que optimizasen el funcionamiento del sistema para cada caso concreto.

Sería interesante también realizar en el preprocesado un filtrado por clase de buque más exhaustivo, creando para cada tipo concreto de barco un modelo propio. Esto permitiría lograr un mayor entendimiento del comportamiento específico de cada clase de barco y ser capaces de detectar anomalías inherentes a cada una de ellas. También convendría comprobar el funcionamiento de la aplicación sin realizar un filtrado previo por rumbo.

En este trabajo la aplicación ha sido testada únicamente en zonas con un tráfico regulado y ordenado. Sería interesante lograr, a medida que avanza y mejora el sistema de AI, la adaptación de la aplicación a zonas más amplias y con un tráfico más complejo. La meta que se persigue es la creación de un sistema capaz de analizar todo el tráfico marítimo de relevancia para un país o institución, siendo capaz de localizar patrones comunes a diferentes escenarios y de aplicar lo aprendido de una zona a otra.

En lo referente a la presentación gráfica, se podría buscar diferentes plataformas con un mayor enfoque a al entorno marítimo, como OpenCPN, que permitan integrar otras fuentes de información, buscando finalmente la integración de esta aplicación en un sistema VTS.

6 BIBLIOGRAFÍA

En esta sección figuran todas las referencias utilizadas en este trabajo.

[1] «*International Chamber of Shipping. Representing the Global Shipping Industry*» Purestone, 2015. [En línea]. Available: http://www.ics-shipping.org/shipping-facts/shipping-facts. [Último acceso: Enero 2016].

[2] «CIAIM - Memoria Anual 2014» de Comisión Permanente de Investigación de Accidentes e Incidentes Marítimos, Madrid, 2014.

[3] MARCOM, «*Maritime Command*» [En línea]. Available: http://www.mc.nato.int/about/Documents/Newcomers/N5%20%20Northwood%20Strategic%20position%202012.pdf.

[4] Presidencia de Gobierno, «Estrategia de Seguridad Marítima Nacional» 2013.

[5] «USLegal» [En línea]. Available: http://definitions.uslegal.com/t/traffic-separation-scheme/.

[6] Organización Marítima Internacional, Modificación del Dispositivo de Separación del Tráfico "a la altura de Finisterre", vol. ASAMBLEA 23º periodo de sesiones. Punto 17 del orden del día, 2004.

[7] «La Voz de Galicia» [En línea]. Available: http://www.lavozdegalicia.es/galicia/2016/01/05/00161451950424183948589.htm. [Último acceso: Febrero 2016].

[8] J. McCarthy, "*What is artificial Intelligence?*" 12 Nobiembre 2007. [Online]. Available: http://www-formal.stanford.edu/jmc/whatisai/whatisai.html. [Accessed 15 Enero 2016].

[9] S. J. Russell and P. Norvig. Contributing writers: J. F. Canny, J. M. Malik, D. D. Edwards, *Artificial Intelligence: a Modern Approach*, Emglewood Cliffs, New Jersey: Prentice Hall, Inc., 1995.

[10] W. S. McCulloch and W. Pitts, «*A Logical Calculus of the Ideas Immanent in Nervous Activity*» *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115 - 133, 1943.

[11] A. M. Turing, «*Computing Machinary and Intelligence*» MIND. *A Quarterly Review of Psychology and Philosophy*, vol. LIX, nº 236, pp. 433 - 460, 1950.

[12] J. R. Searle, «*Minds, Brains and Programs*» *Behavioral and Brain Sciences*, vol. 3, nº 3, pp. 417 - 457, 1980.

[13] J. McCarthy, M. L. Minsky, N. Rochester and C. E. Shannon, «*A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*» de *Dartmouth Reasearch Project on Artificial Intelligence*, Dartmouth, 1955.

[14] H. A. Simon, *The Shape of Automation for Men and Management*, Harper & Ror, 1965.

[15] Numenta, «NuPIC. *Numenta Platform for Intelligent Computing*» 2016. [En línea]. Available: http://numenta.com/. [Último acceso: Febrero 2016].

[16] «Numenta» 2016. [En línea]. Available: http://numenta.com. [Último acceso: Febrero 2016].

[17] Numenta, «Numenta» 2016. [En línea]. Available: http://numenta.com/learn/hierarchical-memory-white-paper.html. [Último acceso: Febreo 2016].

[18] J. Hawkins, *On Intelligence*, Times Books, 2007.

[19] C. Surpur, «*Cortical Learning Algorithm as Implemented in NuPIC*» 2014. [En línea]. Available: http://chetansurpur.com/slides/2014/5/4/cla-in-nupic.html#1.

[20] Numenta, «*CLA for ML AI Researchers*» [En línea]. Available: https://github.com/numenta/nupic/wiki/CLA-for-ML-AI-Researchers.

[21] Numenta, «*Anomaly Detection and Anomaly Scores*» 28 Marzo 2014. [En línea]. Available: https://github.com/numenta/nupic/wiki/Anomaly-Detection-and-Anomaly-Scores. [Último acceso: Febrero 2016].

[22] Numenta, «Numenta» 2016. [En línea]. Available: http://numenta.com/assets/pdf/whitepapers/Geospatial%20Tracking%20White%20Paper.pdf. [Último acceso: Febrero 2016].

[23] Devpost, «*Numenta HTM Challenge*» [En línea]. Available: http://htmchallenge.devpost.com/submissions. [Último acceso: Febrero 2016].

[24] Devpost, «*ATAD Air Traffic Anomaly Detector*» [En línea]. Available: http://devpost.com/software/air-traffic-anomaly-detector. [Último acceso: Febrero 2016].

[25] Devpost, «*Come hack on NuPIC with us in New York City!*» [En línea]. Available: http://nupic2015spring.devpost.com/. [Último acceso: Febrero 2016].

[26] «YouTube. Smart Harbor [DEMO #7] (2014 Fall NuPIC Hackathon)» 22 Octubre 2014. [En línea]. Available: https://www.youtube.com/watch?v=vpAQUHc8_4U. [Último acceso: Febrero 2016].

[27] *U.S. Government Publishing Office*, Part 164.46 *Automatic Identification System*, vol. Titulo 33 Capítulo I Subcapítulo P, Washington, DC, 2016.

[28] *U.S. Coast Guard Navigation Center*, «*Navigation Center*» 2016. [En línea]. Available: http://www.navcen.uscg.gov/images/WhatYouSeeWithAIS.jpg. [Último acceso: 2016].

[29] R. Hirche, AIS. *Automatic Identification System*. Sistema de Identificación Automática. Teoría y Práctica, Tutor, 2010.

[30] *International Maritime Organization* (IMO), «IMO *International Maritime Organization*,» 2016. [En línea]. Available: http://www.imo.org/en/OurWork/Safety/Navigation/Pages/VesselTrafficServices.aspx. [Último acceso: Febrero 2016].

[31] J. Janssens, E. Postma, and J. van den Herik, «*Chapter 8 Density-Based Anomaly Detection in the Maritime Domain*» de *Situation Awareness with Systems of Systems*, New York, Springer, 2013.

[32] Drupal, «Swedish ICT - SICS» 2016. [En línea]. Available: https://www.sics.se/projects/sadv-statistical-anomaly-detection-and-visualization-for-maritime-domain-awareness. [Último acceso: Febrero 2016].

[33] C. Brax, «*Maritime Anomaly Detection*» de *Maritime Security Seminar, Swedish National Defence College*, Stockholm, 2011.

[34] C. Brax, «*Anomaly Detection in the Surveillance Domain*» Örebro University, Örebro, 2011.

[35] E. Martineau, J. Roy and DRDC Valcartier, «*Maritime Anomaly Detection: Domain Introduction and Review of Selected Literature*» Defence R&D Canada, Valcaltier, 2011.

[36] R. Laxhammar, «*Anomaly Detection in Trajectory Data for Surveillance Applications*» Örebro University, Örebro, 2011.

[37] Google, «Ayuda de Google Earth. Acerca de KML» 2016. [En línea]. Available: https://support.google.com/earth/answer/148118?hl=es. [Último acceso: Febrero 2016].

[38] Google Developers, «*Keyhole Markup Language*» 2016. [En línea]. Available: https://developers.google.com/kml/documentation/?hl=es. [Último acceso: Febrero 2016].

[39] NuPIC Community, «*Installing and Building NuPIC*» GitHub, Inc, 2016. [En línea]. Available: https://github.com/numenta/nupic/wiki/Installing-and-Building-NuPIC. [Último acceso: 2016].

[40] NuPIC Open Source, «YouTube - Installing NuPIC on Ubuntu 14 LTS» 13 Septiembre 2015. [En línea]. Available: https://www.youtube.com/watch?v=1fIpgXHXAZA. [Último acceso: 2016].

[41] Python Software Foundation, «*Python Developer's Guide*» 2016. [En línea]. Available: https://docs.python.org/devguide/setup.html#build-dependencies. [Último acceso: 2016].

[42] GitHub Inc., «GitHub - *Numenta Platform for Intelligent Computing*» 2016. [En línea]. Available: https://github.com/numenta/nupic.

[43] MarineTraffic.com, «Marine Traffic - AIS data API Documentation» 2016. [En línea]. Available: http://www.marinetraffic.com/en/ais-api-services/documentation/_:0504a2156e06e629ea85176e7fe3dc24.

[44] Google Developers, «*Keyhole Markup Language*» 2016. [En línea]. Available: https://developers.google.com/kml/articles/csvtokml.

[45] «Marine Traffic» 2016. [En línea]. Available: http://www.marinetraffic.com/en/ais/details/ships/shipid:164947/mmsi:224127000/imo:9268411/vessel:VOLCAN_DE_TINDAYA. [Último acceso: Febrero 2016].

[46] L. Avik Partners, «Grok» 2015. [En línea]. Available: http://grokstream.com. [Último acceso: Febrero 2016].

[47] Numenta, «HTM for Stocks» 2016. [En línea]. Available: http://numenta.com/htm-for-stocks/. [Último acceso: 2016].

[48] Google, «Google Play. HTM for Stocks» 2016. [En línea]. Available: https://play.google.com/store/apps/details?id=com.numenta.taurus. [Último acceso: Febrero 2016].

[49] Numenta, «Numenta» 2016. [En línea]. Available:

http://numenta.com/assets/pdf/whitepapers/Rogue%20Behavior%20Detection%20White%20Paper.pdf. [Último acceso: Febrero 2016].

[50] Indra, «*Indra Vessels Traffic Services (VTS)*» 2016. [En línea]. Available: http://www.indracompany.com/en/vessel-traffic-services-vts. [Último acceso: Febrero 2016].

[51] I. Héctor D. Polo Morales | CIO - International MarConsult, «International MarConsult» 2016. [En línea]. Available: http://marconsult.com.pa/index.php/vts-servicio-de-monitoreo-maritimo. [Último acceso: 2016].

[52] Xpert Solutions Technologiques Inc., INNAV - VTMIS White Paper, 2014.

[53] Navielektro, «Navielektro MARITAS» 2015. [En línea]. Available: http://www.navielektro.fi/maritas.html. [Último acceso: Febrero 2016].

[54] Navielektro, «Navielektro MATIS» 2015. [En línea]. Available: http://www.navielektro.fi/matis.html. [Último acceso: Febrero 2016].

ANEXO I: DICCIONARIO DE SIGLAS Y ACRÓNIMOS

En este anexo se recogen todas las siglas y acrónimos utilizados a lo largo de este trabajo. Aquellos que no poseen una traducción reconocida en español se presentan únicamente en el idioma original. En el caso de que la sigla o acrónimo esté en otro idioma pero posea una traducción de uso común en español se ha presentado en español con el significado en el idioma original entre paréntesis.

AI: Inteligencia Artificial (*Artificial Intelligence*)

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

API: Interfaz de Programación de Aplicaciones (*Application Programming Interface*)

CCTV: Circuito Cerrado de Televisión

CLA: Algoritmos de Aprendizaje Cortical (*Cortical Learning Algorithms)*

CME: Conocimiento del Entorno Marítimo

COVAM: Centro de Operaciones y Vigilancia de Acción Marítima

CSV: Valores Separados por Coma (*Comma-Separated Values*)

DST: Dispositivos de Separación de Tráfico

DWT: Tonelaje de Peso Muerto (*Deadweight Tonnage*)

ETA: Tiempo Estimado de Llegada al Destino (*Estimated Time of Arrival*)

GMDSS: Sistema Mundial de Socorro y Seguridad Marítimos (*Global Maritime Distress Safety System*)

GRT: Arqueo de Registro Bruto (*Gross Register Tonnage*)

HTM: Memoria Temporal Jerárquica (*Hierarchical Temporal Memory*)

KML: *Keyhole Markup Language*

MADIS: *Maritime Anomaly Detection Information System*

MARITAS: *Maritime Traffic Authority System*

MATIS: *Maritime Awareness Tactical Information System*

MMSI: Número de Identificación del Servicio Móvil Marítimo (*Maritime Mobile Service Identity*)

MSA: *Maritime Situational Awareness*

MSO: Operaciones de Seguridad Marítima (*Maritime Security Operations*)

NuPIC: *Numenta Platform for Intelligent Computing*

OGC: *Open Geospatial Consortium*

OMI: Organización Marítima Internacional

OTAN: Organización del Tratado del Atlántico Norte

RDF: Radiogoniómetro (*Radio Direction Finders*)

SBADM: *State-Based Anomaly Detection Method*

SDR: Representación Distribuida Dispersa (*Sparse Distributed Representation*)

SOTDMA: Acceso Múltiple con División de Tiempo Auto-Organizada (*Self-Organized Time Division Multiple Access*)

SP: *Spatial Pooler*

TP: *Temporal Pooler*

UIT: Unión Internacional de Telecomunicaciones

UK: Reino Unido (*United Kingdom*)

VHF: Frecuencia Muy Alta (*Very High Frecuency*)

VTS: Servicios de Tráfico Marítimo (*Vessels Traffic Service*)

ANEXO II: APLICACIONES DE NUMENTA

Grok

Grok [46] es una aplicación de Numenta desarrollada junto a Avik para la detección de anomalías en indicadores de servicio de Amazon Web Services. Actualmente se encuentra disponible en el Amazon Web Services Marketplace. Grok permite un mayor conocimiento de los sistemas IT críticos para que los problemas se puedan solucionar antes de que ocurran. Grok aprende mediante HTM patrones complejos en el entorno de una aplicación para después identificar comportamientos inusuales en dichos sistemas. Finalmente muestra a través de una aplicación web sus predicciones y el flujo de datos permitiendo al usuario comprobar la salud de su sistema en cualquier momento y lugar. Esta aplicación se puede adquirir desde su propia página web [46].

Figura AII-1 Logo Grok [46]

Figura AII-2 Ejemplo de utilización de Grok con una anomalía en vista por día [46]

HTM for Stocks

HTM for Stocks [47] es una aplicación HTM que continuamente monitoriza cientos de compañías que cotizan en bolsa y envía una alerta si ocurre algo inusual con alguna de ellas. *HTM for Stocks* utiliza algoritmos HTM para modelar el precio de mercado, el volumen de acciones y datos de Twitter relacionados con 200 de las mayores empresas que cotizan en bolsa. Algunas de las compañías que monitoriza son Apple, Google, Amazon o Starbucks. *HTM for Stocks* es una aplicación móvil para Android que se puede adquirir en Google Play.

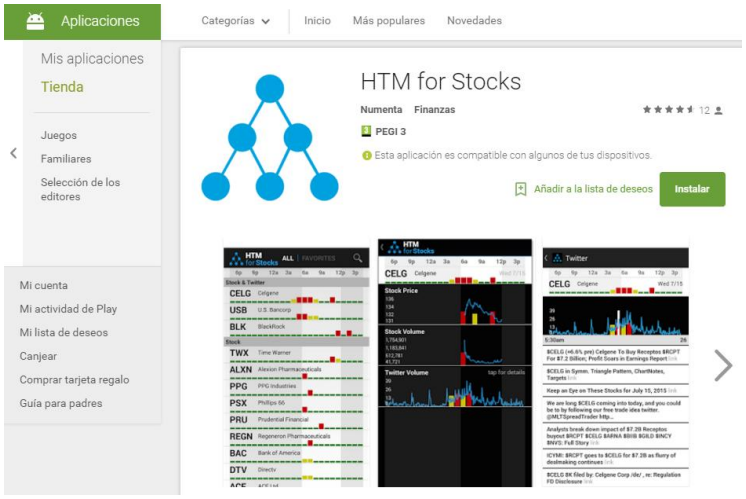

Figura AII-3 Captura de la página de descarga de *HTM for Stocks* en Google Play [48]

Rogue Behavior Detection

Rogue Behavior Detection [43] es una aplicación que utiliza HTM para, de manera automática, aprender y modelar el comportamiento de un individuo y así identificar actividades anormales o irregulares. La creación automática de modelos de comportamiento individuales proporciona una precisión, flexibilidad y escalabilidad superior. Además elimina la necesidad de decidir que es normal y anormal para cada individuo. Estos modelos se desarrollan codificando datos generados tanto por humanos como por máquinas en SDR que posteriormente son procesadas por los algoritmos de aprendizaje HTM. El código y los algoritmos utilizados para la realización de la aplicación se encuentran disponibles a través de NuPIC.

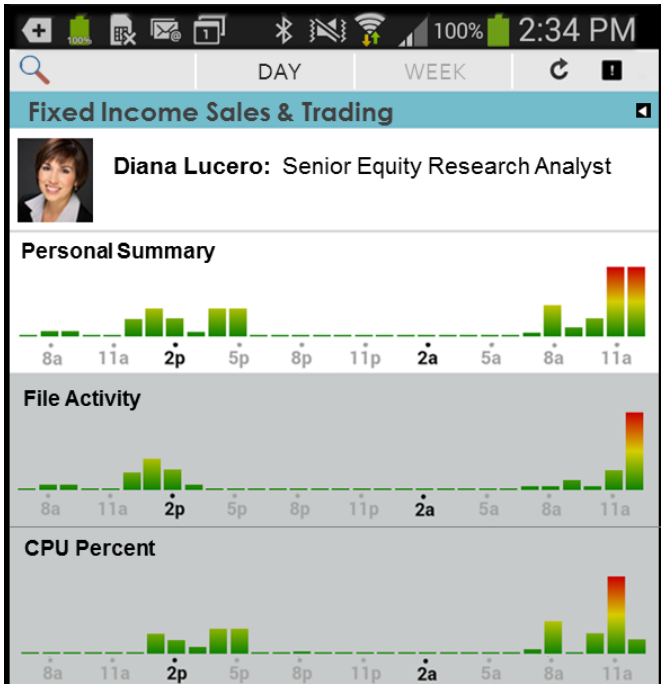

Figura AII-4 Presentación de la aplicación mostrando una visión general de las anomalías de un empleado [49]

ANEXO III: TRABAJOS PRESENTADOS EN LA 1ª EDICIÓN DEL *NUMENTA HTM CHALLENGE*

- *HTM for Adelaide Arterial Traffic Flow.* Usa algoritmos HTM de NuPIC para realizar una detección automática de incidentes en el tráfico arterial de Adelaida.
- *ATAD. Air Traffic Anomaly Detector*.
- *ECG + HTM.* Sistema detector de anomalías ECG usando HTM.
- *ComportexViz, but for Nupic.* Para ver tu HTM funcionar.
- *CloudSonar*. Encuentra los eventos que deberías ver.
- *Fitness Fortuna.* Una aplicación para Apple Watch contra la que compites para alcanzar tus objetivos físicos.
- *Clairvaux.* Predicción y visualización de conflictos armados.
- *AI-909.* Batería inteligente que genera nuevos ritmos.
- *Semantic Anomaly Detection.* Monitoriza *twits* de candidatos presidenciales y detecta errores semánticos.
- *Chatanomaly*. Detección de anomalías en la actividad de un grupo de conversación online.
- *Column Swarm Reinforcement Learning (CSRL).* Una arquitectura de refuerzo del lenguaje de propósito general basado en HTM e inteligencia *swarm*.
- *Brain Squared.* Clasifica los patrones de tus ondas cerebrales.
- *Hard Drive Lifeguard.* Predicción de fallo en un disco duro usando *SMART data set*.
- *Knowledge Graph with HTMs.* Bots inteligentes dirigidos por un gráfico de conocimiento construido con HTMs.

ANEXO IV: EJEMPLOS VTS

INDRA

Las soluciones de Indra en Sistemas de Gestión del Tráfico Marítimo (VTS) [50] proporcionan, en tiempo real, un escenario operacional integrado que permite la mejora de la seguridad marítima, la eficacia en la planificación del tráfico, la protección medioambiental del agua y la seguridad de las infraestructuras del puerto. Su solución tecnológica contempla la integración de una amplia variedad de sensores (radares, CCTV, AIS, RDF, comunicaciones radio e integración de voz, GMDSS, estaciones meteorológicas e hidrológicas y otros sensores bajo petición del cliente). Basadas en su software propio iMare, Indra ha instalado ya sus soluciones VTS en puertos como Southampton (UK), Mohammedia (Marruecos), Cádiz o Valencia (España). También fue elegida por la Autoridad Marítima de Polonia para gestionar el tráfico marítimo y la seguridad de la costa polaca.

International MarConsult

Sus sistemas de tráfico marítimo (VTS) [51] ofrecen a los operadores una visión general de sus áreas de interés, lo que les permite controlar e interactuar con situaciones marítimas de riesgo, y proporcionar los medios para comunicarse rápidamente con toda la autoridad pertinente

El núcleo de estos sistemas es el software modular y los componentes de hardware que se han desarrollado y perfeccionado durante más de 25 años. Esta tecnología utiliza una red de sensores que pueden incluir: radares, estaciones base AIS, RDF, cámaras y cámaras termográficas. Su software VTS supervisa la gestión de esta red de sensores y las diversas fuentes de seguimiento a un buque específico u otro objetivo. También puede vincular esta trayectoria del barco a la información de base de datos histórica existente, ya sea local o internacional. Todas las operaciones se archivan de forma continua incluyendo pistas, video radar, acciones del operador, comunicaciones de voz y cámara de vídeo / imágenes.

Xpert Solutions Technologiques

INNAV [52] es el sistema de gestión del tráfico utilizado por la *Canadian Coast Guard* para monitorizar los miles de buques que transitan por las rutas marítimas canadienses cada día. La arquitectura *Distributed Database* de INNAV está diseñada para un intercambio de información operacional a través del país en tiempo real.

Navielecktro

- **Maritime Traffic Authority System (MARITAS)**

El sistema MARITAS [53] está diseñado para apoyar al usuario en operaciones en curso proveyéndole una imagen de la situación precisa y fácil de comprender. MARITAS ofrece herramientas para el análisis que ayudan al usuario en el proceso de toma de decisiones.

- **Maritime Awareness Tactical Information System (MATIS)**

Navielecktro ha integrado sus avances en el campo de la vigilancia marítima en un sistema de apoyo multi-rol para aplicaciones militares, MATIS [54]. MATIS es una solución que cubre una miríada de bases con la flexibilidad operativa en mente. El sistema funciona tanto como sistema de vigilancia como de mando y control.

MATIS utiliza un sistema de detección de anomalías, MADIS, que, a partir de una serie de parámetros introducidos por el usuario y haciendo uso de datos históricos, facilita al usuario la vigilancia del entorno marítimo.

ANEXO V: MARITIMEANOMALIES.PY

```
1. #!/usr/bin/env python
2.
3. import os
4. import sys
5. from optparse import OptionParser
6. from run import run
7. from tools.convertion import convertion
8. from tools.anomaly_to_kml import anomalyrepresentation as KML
9.
10.      DEFAULT_OUTPUT_DIR = "output"
11.      verbose = False
12.      scriptDir = os.path.dirname(os.path.realpath(__file__))
13.
14.
15.      parser = OptionParser(
16.        usage="%prog <path/to/input/file> [options]\n\nRun NuPIC on
   specified "
17.              "location file, which should already be in the proper
   format "
18.              "(downloaded from the simulator)."
19.      )
20.
21.      parser.add_option(
22.        "-m",
23.        "--manual-sequence",
24.        action="store_true",
25.        default=False,
26.        dest="manualSequence",
27.        help="Automatically breaks into sequences based upon time gaps."
28.      )
29.      parser.add_option(
30.        "-i",
31.        "--initial",
32.        action="store_true",
33.        default=False,
34.        dest="initial",
35.        help="Starts a new model."
36.      )
37.      parser.add_option(
38.        "-t",
39.        "--time-encoders",
40.        action="store_true",
41.        default=False,
42.        dest="useTimeEncoders",
43.        help="Adds time of day encoder to model params."
44.      )
45.      parser.add_option(
46.        "-v",
47.        "--verbose",
48.        action="store_true",
49.        default=False,
50.        dest="verbose",
51.        help="Print debugging statements."
52.      )
```

```
53.     parser.add_option(
54.       "-o",
55.       "--output-dir",
56.       default=DEFAULT_OUTPUT_DIR,
57.       dest="outputDir",
58.       help="Where to write the output file."
59.     )
60.     parser.add_option(
61.       "-s",
62.       "--scale",
63.       default=False,
64.       dest="scale",
65.       help="Meter resolution for Geospatial Coordinate Encoder (default
   5m)."
66.     )
67.
68.     def maritimeanomalies(inputPath, outputDir, useTimeEncoders, scale,
   autoSequence, initial):
69.
70.       outputPath = os.path.abspath(outputDir)
71.       if not os.path.exists(outputPath):
72.         os.makedirs(outputPath)
73.
74.       convertedOutputPath
   = os.path.join(outputPath, "converted_data.csv")
75.       if verbose: print "Converting %s..." % inputPath
76.       convertion(inputPath, convertedOutputPath)
77.
78.       run(convertedOutputPath, outputPath,
   useTimeEncoders=useTimeEncoders, scale=scale,
   autoSequence=autoSequence,initial=initial, verbose=verbose)
79.
80.       representationDataPath
   = os.path.join(outputPath, "anomaly_scores.csv")
81.       representationOutputPath
   = os.path.join(outputPath, "anomaly_representation.kml")
82.       if verbose: print "Creating visualization at
   %s..." % representationOutputPath
83.       KML(representationDataPath,representationOutputPath)
84.
85.
86.     if __name__ == "__main__":
87.       (options, args) = parser.parse_args(sys.argv[1:])
88.       try:
89.         input_path = args.pop(0)
90.     except IndexError:
91.         parser.print_help(sys.stderr)
92.         sys.exit()
93.
94.       verbose = options.verbose
95.
96.       maritimeanomalies(
97.         input_path,
98.         options.outputDir,
99.         options.useTimeEncoders,
100.        options.scale,
101.        not options.manualSequence,
102.        options.initial)
```

ANEXO VI: RUN.PY

```
1. #!/usr/bin/env python
2.
3. import os
4. import sys
5.
6. from tools.preprocess_data import preprocess
7. from model.geospatial_anomaly import runGeospatialAnomaly
8.
9. verbose = False
10.      scriptDir = os.path.dirname(os.path.realpath(__file__))
11.
12.      def run(inputPath, outputPath, useTimeEncoders, scale,
   autoSequence, initial, verbose):
13.
14.        preProcessedOutputPath
   = os.path.join(outputPath, "preprocessed_data.csv")
15.        if verbose: print "Pre-processing %s..." % inputPath
16.        preprocess(inputPath, preProcessedOutputPath, verbose=verbose)
17.
18.        anomalyOutputPath
   = os.path.join(outputPath, "anomaly_scores.csv")
19.        modelOutputPath
   = os.path.join(os.path.abspath('model/model/anomaly'))
20.        if verbose: print "Running NuPIC on
   %s..." % preProcessedOutputPath
21.        runGeospatialAnomaly(preProcessedOutputPath,
22.                             anomalyOutputPath,
23.                             modelOutputPath,
24.                             scale=scale,
25.                             autoSequence=autoSequence,
26.                             useTimeEncoders=useTimeEncoders,
27.                             initial=initial,
28.                             verbose=verbose)
29.
30.
31.      if __name__ == "__main__":
32.        (options, args) = parser.parse_args(sys.argv[1:])
33.        try:
34.          input_path = args.pop(0)
35.      except IndexError:
36.          parser.print_help(sys.stderr)
37.          sys.exit()
38.
39.        verbose = options.verbose
40.
41.        run(input_path, outputDir, useTimeEncoders, scale,
   manualSequence, initial, verbose)
```

ANEXO VII: CONVERTION.PY

```
1. import csv
2. import time
3. import calendar
4.
5. def readais(dataPath):
6.   """It opens the original file, takes all the information and stores it
   in filelist. Then return file list"""
7.
8.   my_file = open(dataPath,'rb')
9.   reader = csv.reader(my_file)
10.       filelist = []
11.       for row in reader:
12.         rowlist = []
13.         for item in row:
14.           rowlist.append(item)
15.         filelist.append(rowlist)
16.         rowlist = []
17.       my_file.close()
18.       return filelist
19.
20.     def aisconvertion(filelist):
21.       """It takes filelist and changes it to the format required by
   nupic.geospatial"""
22.
23.       fileconverted = []
24.       for i in range(1,int(len(filelist))):
25.         rowlist = []
26.         rowlist.append(filelist[i][0])
27.         timetuple = time.strptime(str(filelist[i][6]),"%Y-%m-
   %dT%H:%M:%S")
28.         dt = calendar.timegm(timetuple)
29.         rowlist.append(int(dt))
30.         rowlist.append(filelist[i][2])
31.         rowlist.append(filelist[i][1])
32.         rowlist.append("")
33.         speed = float(filelist[i][3]) * 0.05144
34.         rowlist.append(speed)
35.         rowlist.append("")
36.         rowlist.append(1)
37.
38.         fileconverted.append(rowlist)
39.
40.         rowlist = []
41.
42.       return fileconverted
43.
44.     def writeais(outPath,fileconverted):
45.       """Writes the converted file returned by convertion in
   the file that we will run with nupic.geospatial"""
46.
47.       my_file = open(outPath,'wb')
48.       writer = csv.writer(my_file, delimiter=',', quotechar="",
   quoting=csv.QUOTE_NONE)
49.       for row in fileconverted:
50.         writer.writerow(row)
```

```
51.         my_file.close()
52.
53.       def convertion(dataPath, outPath):
54.
55.         filelist = readais(dataPath)
56.         fileconverted = aisconvertion(filelist)
57.         writeais(outPath, fileconverted)
58.
59.
60.       if __name__ == "__main__":
61.         if len(sys.argv) < 3:
62.           print ("Usage: {0} "
63.                  "/path/to/data.csv
    /path/to/outfile.csv").format(sys.argv[0])
64.           sys.exit(0)
65.
66.         dataPath = sys.argv[1]
67.         outPath = sys.argv[2]
68.         convertion(dataPath, outPath)
```

ANEXO VIII: PREPROCESS_DATA.PY

```
1. #!/usr/bin/env python
2.
3.
4. import csv
5. import datetime
6. import sys
7.
8.
9.
10.     def preprocess(dataPath, outPath, verbose=False):
11.       with open(dataPath) as csvfile:
12.         reader = csv.reader(csvfile)
13.         writer = csv.writer(open(outPath, "wb"))
14.         reader1 = []
15.         for row in reader:
16.           line = []
17.           for item in row:
18.             line.append(str(item))
19.           reader1.append(line)
20.
21.         reader2 = reader1
22.
23.         usedtracks = []
24.         for row in reader2:
25.           trackname = str(row[0])
26.           check = 0
27.           for track in usedtracks:
28.             if track == trackname:
29.               check += 1
30.           if check == 0:
31.             lastlat = 0
32.             lastlon = 0
33.             lasttimestamp = 0
34.             for i in reader1:
35.               keep = True
36.               if i[0] == trackname:
37.                 if i[2] == lastlon and i[3] == lastlat:
38.                  keep = False
39.                 if i[1] == lasttimestamp:
40.                   keep = False
41.                 if keep:
42.                   if verbose:
43.                     print "Keeping row:\t{0}".format(row)
44.                   writer.writerow(i)
45.               lastlon = i[2]
46.               lastlat = i[3]
47.               lasttimestamp = i[1]
48.             usedtracks.append(trackname)
49.         print len(usedtracks)
50.
51.
52.
53.     if __name__ == "__main__":
54.       if len(sys.argv) < 3:
```

```
55.          print ("Usage: {0} "
56.                 "/path/to/data.csv
   /path/to/outfile.csv").format(sys.argv[0])
57.          sys.exit(0)
58.
59.        dataPath = sys.argv[1]
60.        outPath = sys.argv[2]
61.        preprocess(dataPath, outPath)
```

ANEXO IX: GEOSPATIAL_ANOMALY.PY

```
1. #!/usr/bin/env python
2.
3. """
4. A simple client to create a CLA anomaly detection model for geospatial
   data.
5. """
6.
7. import csv
8. import datetime
9. import sys
10.
11.     from nupic.frameworks.opf.modelfactory import ModelFactory
12.
13.     import model_params
14.
15.
16.
17.     DEFAULT_DATA_PATH = "data/commute.csv"
18.     DEFAULT_OUTPUT_PATH = "anomaly_scores.csv"
19.
20.     ACCURACY_THRESHOLD = 80  # meters
21.     INTERVAL_THRESHOLD = 300  # seconds
22.
23.
24.
25.     def addTimeEncoders(params):
26.       params["modelParams"]["sensorParams"]["encoders"]["timestamp_time
   OfDay"] = {
27.         "fieldname": u"timestamp",
28.         "name": u"timestamp_timeOfDay",
29.         "timeOfDay": (51, 9.5),
30.         "type": "DateEncoder"
31.       }
32.       return params
33.
34.
35.
36.     def setEncoderScale(params, scale):
37.       params["modelParams"]["sensorParams"]["encoders"]["vector"]["scal
   e"] = \
38.         int(scale)
39.       return params
40.
41.
42.
43.     def createModel(useTimeEncoders, scale, verbose):
44.       params = model_params.MODEL_PARAMS
45.       if useTimeEncoders:
46.         params = addTimeEncoders(params)
47.       if scale:
48.         params = setEncoderScale(params, scale)
49.       if verbose:
50.         print "Model parameters:"
51.         print params
```

```
52.         model = ModelFactory.create(params)
53.         model.enableInference({"predictedField": "vector"})
54.         return model
55.
56.
57.
58.       def runGeospatialAnomaly(dataPath, outputPath, modelPath,
59.                                scale=False,
60.                                autoSequence=True,
61.                                useTimeEncoders=False,
62.                                initial=False,
63.                                verbose=False):
64.         if initial:
65.           model = createModel(useTimeEncoders, scale, verbose)
66.         else:
67.           model = ModelFactory.loadFromCheckpoint(modelPath)
68.           model.enableInference({"predictedField": "vector"})
69.
70.         with open (dataPath) as fin:
71.           reader = csv.reader(fin)
72.           csvWriter = csv.writer(open(outputPath,"wb"))
73.           csvWriter.writerow(["trackName",
74.                  "timestamp",
75.                              "longitude",
76.                              "latitude",
77.       "speed",
78.                              "anomaly_score",
79.                              "new_sequence"])
80.
81.           reader.next()
82.           reader.next()
83.           reader.next()
84.
85.           lastTimestamp = None
86.           lastTrackName = None
87.           outputFormat = "%Y-%m-%dT%H:%M:%S"
88.
89.           for _, record in enumerate(reader, start=1):
90.             trackName = record[0]
91.             timestamp = datetime.datetime.fromtimestamp(int(record[1]))
92.             longitude = float(record[2])
93.             latitude = float(record[3])
94.             speed = float(record[5])
95.             accuracy = float(record[7])
96.
97.             altitude = float(record[4]) if record[4] != "" else None
98.
99.             if accuracy > ACCURACY_THRESHOLD:
100.              continue
101.
102.            newSequence = False
103.            # Handle the automatic sequence creation
104.            if autoSequence:
105.              if lastTimestamp and (
106.                (timestamp -
   lastTimestamp).total_seconds() > INTERVAL_THRESHOLD):
107.                newSequence = True
108.            # Manual sequence resets depend on the track name
```

```
109.          else:
110.            if trackName != lastTrackName:
111.              newSequence = True
112.
113.          lastTimestamp = timestamp
114.          lastTrackName = trackName
115.
116.          if newSequence:
117.            if verbose:
118.              print "Starting new sequence..."
119.            model.resetSequenceStates()
120.
121.          modelInput = {
122.            "vector": (speed, longitude, latitude, altitude)
123.          }
124.
125.          if useTimeEncoders:
126.      modelInput["timestamp"] = timestamp
127.
128.          result = model.run(modelInput)
129.          anomalyScore = result.inferences["anomalyScore"]
130.
131.          csvWriter.writerow([trackName,
132.            timestamp.strftime(outputFormat),
133.                              longitude,
134.                              latitude,
135.                              speed,
136.                              anomalyScore,
137.                              1 if newSequence else 0])
138.          if verbose:
139.            print "[{0} - {1}] - Anomaly score: {2}.".format(trackName,
    timestamp, anomalyScore)
140.
141.        model.save(modelPath)
142.        print "Anomaly scores have been written to
    {0}".format(outputPath)
143.
144.
145.
146.      if __name__ == "__main__":
147.        dataPath = DEFAULT_DATA_PATH
148.        outputPath = DEFAULT_OUTPUT_PATH
149.
150.        if len(sys.argv) > 1:
151.          dataPath = sys.argv[1]
152.
153.        if len(sys.argv) > 2:
154.          outputPath = sys.argv[2]
155.
156.        runGeospatialAnomaly(dataPath, outputPath, modelPath)
```

ANEXO X: 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': 4096,
                        'w': 101,
                        '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).
```

```
47.                 # Valid keys is the desired combination of the
    following:
48.                 #  days, hours, minutes, seconds, milliseconds,
    microseconds, weeks
49.                 #
50.                 # Example for 1.5 days: sensorAutoReset =
    dict(days=1,hours=12),
51.                 'sensorAutoReset' : None,
52.             },
53.
54.             'spEnable': True,
55.
56.             'spParams': {
57.                 # SP diagnostic output verbosity control;
58.                 # 0: silent; >=1: some info; >=2: more info;
59.                 'spVerbosity' : 0,
60.
61.                 # Spatial Pooler implementation selector.
62.                 # Options: 'py', 'cpp' (speed optimized, new)
63.                 'spatialImp' : 'cpp',
64.
65.                 'globalInhibition': 1,
66.
67.                 # Number of columns in the SP (must be same as in TP)
68.                 'columnCount': 2048,
69.
70.                 'inputWidth': 0,
71.
72.                 # SP inhibition control (absolute value);
73.                 # Maximum number of active columns in the SP region's
    output (when
74.                 # there are more, the weaker ones are suppressed)
75.                 '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': 64,
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': 40,
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': 64,
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': 276,
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 XI: ANOMALY_TO_KML.PY

```
1. #!/usr/bin/env python
2.
3. import csv
4. import xml.dom.minidom
5. import sys
6.
7. import datetime
8. import time
9. import calendar
10.
11.     def extractCoordinates(row):
12.       # This extracts the coordinates from a row and returns it as a
   list. This requires knowing
13.       # ahead of time what the columns are that hold
   the address information.
14.       return '%s,%s' % (row[2], row[3])
15.
16.     def createPlacemark(kmlDoc, row, order):
17.       # This creates a <Placemark> element for a row of data.
18.       # A row is a dict.
19.       placemarkElement = kmlDoc.createElement('Placemark')
20.       extElement = kmlDoc.createElement('ExtendedData')
21.       placemarkElement.appendChild(extElement)
22.
23.       # Loop through the columns and create a <Data> element for
   every field that has a value.
24.       for i in range(0, len(order)):
25.           dataElement = kmlDoc.createElement('Data')
26.           dataElement.setAttribute('name', order[i])
27.           valueElement = kmlDoc.createElement('value')
28.           dataElement.appendChild(valueElement)
29.           valueText = kmlDoc.createTextNode(row[i])
30.           valueElement.appendChild(valueText)
31.           extElement.appendChild(dataElement)
32.
33.       if float(row[5]) <= 0.25:
34.         styleElement = kmlDoc.createElement('styleUrl')
35.         styleElement.appendChild(kmlDoc.createTextNode('#green'))
36.         placemarkElement.appendChild(styleElement)
37.       elif float(row[5]) <= 0.75:
38.         styleElement = kmlDoc.createElement('styleUrl')
39.         styleElement.appendChild(kmlDoc.createTextNode('#yellow'))
40.         placemarkElement.appendChild(styleElement)
41.       else:
42.         styleElement = kmlDoc.createElement('styleUrl')
43.         styleElement.appendChild(kmlDoc.createTextNode('#red'))
44.         placemarkElement.appendChild(styleElement)
45.       pointElement = kmlDoc.createElement('Point')
46.       placemarkElement.appendChild(pointElement)
47.       coordinates = extractCoordinates(row)
48.       coorElement = kmlDoc.createElement('coordinates')
49.       coorElement.appendChild(kmlDoc.createTextNode(coordinates))
50.       pointElement.appendChild(coorElement)
51.       return placemarkElement
```

```
52.
53.
54.     def createKML(csvReader, csvreader1, fileName, order):
55.       # This constructs the KML document from the CSV file.
56.       kmlDoc = xml.dom.minidom.Document()
57.
58.       kmlElement =
   kmlDoc.createElementNS('http://earth.google.com/kml/2.2', 'kml')
59.       kmlElement.setAttribute('xmlns', 'http://earth.google.com/kml/2.2
   ')
60.       kmlElement = kmlDoc.appendChild(kmlElement)
61.
62.       documentElement = kmlDoc.createElement('Document')
63.
64.       styleElement = kmlDoc.createElement('Style')
65.       styleElement.setAttribute('id', 'green')
66.       iconstyleElement = kmlDoc.createElement('IconStyle')
67.       scaleElement = kmlDoc.createElement('scale')
68.       scaleText = kmlDoc.createTextNode('0.5')
69.       scaleElement.appendChild(scaleText)
70.       iconstyleElement.appendChild(scaleElement)
71.       iconElement = kmlDoc.createElement('Icon')
72.       hrefElement = kmlDoc.createElement('href')
73.       hrefText =
   kmlDoc.createTextNode('http://maps.google.com/mapfiles/kml/paddle/grn-
   blank.png')
74.       hrefElement.appendChild(hrefText)
75.       iconElement.appendChild(hrefElement)
76.       iconstyleElement.appendChild(iconElement)
77.       styleElement.appendChild(iconstyleElement)
78.       documentElement.appendChild(styleElement)
79.
80.       styleElement = kmlDoc.createElement('Style')
81.       styleElement.setAttribute('id', 'yellow')
82.       iconstyleElement = kmlDoc.createElement('IconStyle')
83.       scaleElement = kmlDoc.createElement('scale')
84.       scaleText = kmlDoc.createTextNode('0.6')
85.       scaleElement.appendChild(scaleText)
86.       iconstyleElement.appendChild(scaleElement)
87.       iconElement = kmlDoc.createElement('Icon')
88.       hrefElement = kmlDoc.createElement('href')
89.       hrefText =
   kmlDoc.createTextNode('http://maps.google.com/mapfiles/kml/paddle/ylw-
   blank.png')
90.       hrefElement.appendChild(hrefText)
91.       iconElement.appendChild(hrefElement)
92.       iconstyleElement.appendChild(iconElement)
93.       styleElement.appendChild(iconstyleElement)
94.       documentElement.appendChild(styleElement)
95.
96.       styleElement = kmlDoc.createElement('Style')
97.       styleElement.setAttribute('id', 'red')
98.       iconstyleElement = kmlDoc.createElement('IconStyle')
99.       scaleElement = kmlDoc.createElement('scale')
100.      scaleText = kmlDoc.createTextNode('0.8')
101.      scaleElement.appendChild(scaleText)
102.      iconstyleElement.appendChild(scaleElement)
103.      iconElement = kmlDoc.createElement('Icon')
```

```
104.        hrefElement = kmlDoc.createElement('href')
105.        hrefText =
    kmlDoc.createTextNode('http://maps.google.com/mapfiles/kml/paddle/red-
    blank.png')
106.        hrefElement.appendChild(hrefText)
107.        iconElement.appendChild(hrefElement)
108.        iconstyleElement.appendChild(iconElement)
109.        styleElement.appendChild(iconstyleElement)
110.        documentElement.appendChild(styleElement)
111.
112.        documentElement = kmlElement.appendChild(documentElement)
113.
114.        # Skip the header line.
115.        csvReader.next()
116.        usedtracks = ["0"]
117.        for row in csvReader:
118.          dt = datetime.date.today()
119.          timetuple = time.strptime(str(dt),"%Y-%m-%d")
120.          dtp = calendar.timegm(timetuple)
121.          timetuple1 = time.strptime(str(row['timestamp']),"%Y-%m-
    %dT%H:%M:%S")
122.          timestamp = calendar.timegm(timetuple1)
123.          timediference = int(timestamp) - int(dtp)
124.          if timediference >= 0:
125.            trackName = str(row['trackName'])
126.            check = 0
127.            for track in usedtracks:
128.              if track == trackName:
129.                check += 1
130.            if check == 0:
131.              folderElement = kmlDoc.createElement('Folder')
132.        nameElement = kmlDoc.createElement('name')
133.        nameText = kmlDoc.createTextNode(trackName)
134.        nameElement.appendChild(nameText)
135.        folderElement.appendChild(nameElement)
136.              csvreader1.remove(csvreader1[0])
137.        for item in csvreader1:
138.                timetuple2 = time.strptime(str(item[1]),"%Y-%m-
    %dT%H:%M:%S")
139.                timestamp1 = calendar.timegm(timetuple2)
140.                timediference1 = int(timestamp1) - int(dtp)
141.                if timediference1 >= 0:
142.           if item[0] == trackName:
143.             placemarkElement = createPlacemark(kmlDoc, item, order)
144.             folderElement.appendChild(placemarkElement)
145.              usedtracks.append(trackName)
146.        documentElement.appendChild(folderElement)
147.
148.        kmlFile = open(fileName, 'w')
149.        kmlFile.write(kmlDoc.toprettyxml('  ', newl = '\n', encoding
    = 'utf-8'))
150.
151.      def anomalyrepresentation(dataPath, outPath):
152.        # This reader opens up 'anomaly_scores.csv'.
153.        # It creates a KML file called 'anomaly_representation.kml'.
154.
155.        # If an argument was passed to the script, it splits the argument
    on a comma
```

```
156.       # and uses the resulting list to specify an order for
   when columns get added.
157.       # Otherwise, it defaults to the order used in the sample.
158.
159.
160.       order
   = ['trackName','timestamp','longitude','latitude','speed','anomaly_score'
   ,'new_sequence']
161.       csvreader = csv.DictReader(open(dataPath),order)
162.       csvreader1 = []
163.       for row in csvreader:
164.         line = []
165.         for key in order:
166.           line.append(str(row[key]))
167.         csvreader1.append(line)
168.       csvreader = csv.DictReader(open(dataPath),order)
169.       kml = createKML(csvreader, csvreader1, outPath, order)
170.
171.     if __name__ == "__main__":
172.       if len(sys.argv) < 3:
173.         print ("Usage: {0} "
174.                "/path/to/data.csv
   /path/to/outfile.csv").format(sys.argv[0])
175.         sys.exit(0)
176.
177.       dataPath = sys.argv[1]
178.       outPath = sys.argv[2]
179.       anomalyrepresentation(dataPath, outPath)
```

ANEXO XII: EXTRACTO EJEMPLO ARCHIVO CSV DE ENTRADA

MMSI, LAT, LON, SPEED, COURSE, STATUS, TIMESTAMP
111224102,42.904730,-10.006710,1410,358,97,2016-02-18T17:42:28
111224102,42.956280,-10.012870,1390,355,97,2016-02-18T17:43:48
111224102,42.982880,-10.015920,1410,355,97,2016-02-18T17:44:30
111224102,43.034970,-10.022600,1420,354,97,2016-02-18T17:45:48
111224102,43.155030,-10.034690,1430,355,97,2016-02-18T17:48:49
111224102,43.181050,-10.031950,1420,15,97,2016-02-18T17:49:29
111224102,43.226100,-10.011570,1460,17,97,2016-02-18T17:50:39
111224102,43.257540,-9.995622,1490,30,97,2016-02-18T17:51:29
111224102,43.403320,-9.857282,1630,35,97,2016-02-18T17:55:29
111224102,43.447410,-9.815110,1630,34,97,2016-02-18T17:56:39
111224102,43.496350,-9.768659,1650,34,97,2016-02-18T17:57:59
209098000,42.918670,-9.789690,100,1,0,2016-02-18T01:47:03
209098000,42.955860,-9.788981,103,2,0,2016-02-18T02:00:03
209098000,42.961380,-9.788968,106,3,0,2016-02-18T02:01:56
209098000,42.966690,-9.789037,107,357,0,2016-02-18T02:03:43
209098000,42.979930,-9.788268,100,7,0,2016-02-18T02:08:23
209098000,42.977180,-9.788609,101,8,0,2016-02-18T02:10:41
209098000,42.990350,-9.787425,102,353,0,2016-02-18T02:12:03
209098000,42.994580,-9.787157,106,358,0,2016-02-18T02:16:41
209098000,43.012490,-9.785080,104,16,0,2016-02-18T02:22:41
209098000,43.012490,-9.785080,104,16,0,2016-02-18T02:25:35
209098000,43.029890,-9.783120,101,356,0,2016-02-18T02:28:44
209098000,43.051750,-9.780073,103,2,0,2016-02-18T02:33:16
209098000,43.047740,-9.780666,103,7,0,2016-02-18T02:34:37
209098000,43.047740,-9.780666,103,7,0,2016-02-18T02:37:30
209098000,43.064890,-9.779068,104,359,0,2016-02-18T02:40:26
209098000,43.064890,-9.779068,104,359,0,2016-02-18T02:43:19
209098000,43.128580,-9.776631,108,2,0,2016-02-18T02:59:57
209098000,43.135750,-9.776417,107,8,0,2016-02-18T03:07:21
209098000,43.154070,-9.776013,106,357,0,2016-02-18T03:13:23

ANEXO XIII: EXTRACTO EJEMPLO CONVERTED_DATA.CSV

314208000,1455529769,-9.710267,43.273370,,2.67488,,1
212208000,1455530900,-10.046670,43.367550,,4.01232,,1
257174000,1455530904,-10.124600,42.988440,,6.27568,,1
220557000,1455530918,-10.199750,43.039830,,6.12136,,1
305251000,1455530932,-10.080330,43.278830,,4.32096,,1
538003287,1455530946,-9.801666,42.895670,,3.90944,,1
271002685,1455530959,-10.042000,43.334170,,5.70984,,1
205521000,1455530978,-9.767899,43.135760,,2.0576,,1
314208000,1455531018,-9.691983,43.300530,,2.88064,,1
305251000,1455531648,-10.079670,43.248660,,4.6296,,1
538003287,1455531661,-9.799167,42.920500,,3.54936,,1
477764600,1455531667,-9.935351,43.528550,,11.36824,,1
564182000,1455531671,-10.060050,43.499620,,5.55552,,1
271002685,1455531674,-10.064830,43.300500,,5.76128,,1
205521000,1455531694,-9.765892,43.151030,,2.36624,,1
212208000,1455531799,-10.065260,43.330360,,4.06376,,1
314208000,1455531802,-9.680634,43.315930,,2.72632,,1
257174000,1455531803,-10.130360,42.944960,,6.06992,,1
205521000,1455532056,-9.764762,43.159230,,2.46912,,1
255925000,1455532075,-10.085540,42.965910,,4.01232,,1
314208000,1455532119,-9.676017,43.322810,,2.67488,,1
257174000,1455532163,-10.132550,42.922940,,5.96704,,1
305251000,1455532189,-10.079330,43.233000,,4.78392,,1
477764600,1455532196,-9.970500,43.481670,,11.26536,,1
538003287,1455532204,-9.798833,42.932330,,3.4464799999999998,,1
564182000,1455532213,-10.072880,43.483260,,6.01848,,1
255925000,1455532435,-10.086700,42.950580,,4.83536,,1
212208000,1455532520,-10.073410,43.312830,,4.21808,,1
257174000,1455532523,-10.134300,42.902400,,6.32712,,1
314208000,1455532529,-9.671017,43.330350,,2.16048,,1
305251000,1455532549,-10.079170,43.217830,,4.78392,,1
538003287,1455532563,-9.798333,42.944830,,3.90944,,1
564182000,1455532573,-10.085310,43.467540,,5.4012,,1

ANEXO XIV: EXTRACTO EJEMPLO PREPROCESSED_DATA.CSV

314208000,1455529769,-9.710267,43.273370,,2.67488,,1
314208000,1455531018,-9.691983,43.300530,,2.88064,,1
314208000,1455532119,-9.676017,43.322810,,2.67488,,1
314208000,1455533922,-9.653967,43.358680,,2.16048,,1
314208000,1455534615,-9.646167,43.371930,,2.26336,,1
212208000,1455530900,-10.046670,43.367550,,4.01232,,1
212208000,1455531799,-10.065260,43.330360,,4.06376,,1
212208000,1455532520,-10.073410,43.312830,,4.21808,,1
212208000,1455535222,-10.093260,43.211720,,4.47528,,1
212208000,1455542600,-10.098330,42.914750,,4.3724,,1
212208000,1455542960,-10.098440,42.899820,,4.42384,,1
257174000,1455532163,-10.132550,42.922940,,5.96704,,1
305251000,1455531648,-10.079670,43.248660,,4.6296,,1
305251000,1455534532,-10.080000,43.124830,,4.57816,,1
305251000,1455535071,-10.080000,43.109660,,4.57816,,1
205521000,1455532056,-9.764762,43.159230,,2.46912,,1
477764600,1455533444,-10.050500,43.372760,,10.90528,,1
255925000,1455532435,-10.086700,42.950580,,4.83536,,1
305148000,1455542639,-10.102350,42.969070,,8.2304,,1
305148000,1455800303,-9.677868,43.350870,,7.81888,,1
305148000,1455800387,-9.674423,43.356250,,7.8703199999999995,,1
305148000,1455800495,-9.670130,43.363200,,7.81888,,1
305148000,1455800609,-9.665604,43.370520,,7.81888,,1
304994000,1455535158,-9.793070,42.941190,,3.03496,,1
304994000,1455542532,-9.800213,43.151910,,3.70368,,1
304994000,1455546126,-9.741042,43.270050,,4.21808,,1
304994000,1455549712,-9.651637,43.390650,,3.6008,,1
304994000,1455550665,-9.620519,43.432580,,4.06376,,1
304994000,1455551164,-9.612930,43.442270,,4.42384,,1
538005389,1455542897,-10.195030,43.099910,,5.81272,,1
304927000,1455542805,-10.056100,43.496460,,4.73248,,1

ANEXO XV: EXTRACTO EJEMPLO ANOMALY_SCORES.CSV

trackName,timestamp,longitude,latitude,speed,anomaly_score,new_sequence
224127000,2016-02-22T17:17:11,-13.85666,28.74722,7.35592,1.0,0
224127000,2016-02-22T17:18:11,-13.85535,28.75113,7.716,1.0,0
224127000,2016-02-22T17:19:11,-13.85411,28.75507,7.61312,1.0,0
224127000,2016-02-22T17:20:11,-13.85287,28.75897,7.66456,1.0,0
224127000,2016-02-22T17:21:11,-13.85161,28.76291,7.716,1.0,0
224127000,2016-02-22T17:22:11,-13.85026,28.76688,7.35592,1.0,0
224127000,2016-02-22T17:23:11,-13.84892,28.77083,7.92176,1.0,0
224127000,2016-02-22T17:24:11,-13.84757,28.77479,7.66456,1.0,0
224127000,2016-02-22T17:25:17,-13.84604,28.77926,7.66456,1.0,0
224127000,2016-02-22T17:26:17,-13.84463,28.78323,7.76744,1.0,0
224127000,2016-02-22T17:27:17,-13.8433,28.78726,7.61312,1.0,0
224127000,2016-02-22T17:28:17,-13.84209,28.79132,7.66456,1.0,0
224127000,2016-02-22T17:29:17,-13.84091,28.79537,7.61312,1.0,0
224127000,2016-02-22T17:30:23,-13.83957,28.79976,7.61312,1.0,0
224127000,2016-02-22T17:31:23,-13.83835,28.80382,7.76744,1.0,0
224127000,2016-02-22T17:32:23,-13.83711,28.80788,7.92176,1.0,0
224127000,2016-02-22T17:33:23,-13.83588,28.81196,7.76744,1.0,0
224127000,2016-02-22T17:34:23,-13.8347,28.81605,7.716,1.0,0
224127000,2016-02-22T17:35:23,-13.83325,28.8201,7.76744,1.0,0
224127000,2016-02-22T17:36:23,-13.83157,28.82407,7.81888,1.0,0
224127000,2016-02-22T17:37:23,-13.82994,28.82809,7.66456,1.0,0
224127000,2016-02-22T17:38:23,-13.82927,28.83227,7.716,1.0,0
224127000,2016-02-22T17:39:23,-13.82905,28.83644,7.716,1.0,0
224127000,2016-02-22T17:40:23,-13.82893,28.84063,7.716,1.0,0
224127000,2016-02-22T17:41:24,-13.82893,28.84495,7.716,1.0,0
224127000,2016-02-22T17:42:28,-13.82906,28.84937,7.61312,1.0,0
224127000,2016-02-22T17:43:28,-13.8295,28.85347,7.56168,1.0,0
224127000,2016-02-22T17:44:28,-13.83029,28.8569,5.81272,1.0,0
224127000,2016-02-22T17:45:33,-13.83091,28.85974,3.3436,1.0,0
224127000,2016-02-22T17:46:43,-13.83083,28.8602,1.69752,1.0,0
224127000,2016-02-22T17:49:34,-13.8329,28.8597,0.0,1.0,0
224127000,2016-02-22T18:10:13,-13.83272,28.85974,2.72632,1.0,1
224127000,2016-02-22T18:41:41,-13.86147,28.73984,1.64608,1.0,1

ANEXO XVI: EXTRACTO EJEMPLO ANOMALY_REPRESENTATION.KML

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <kml xmlns="http://earth.google.com/kml/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>219019073</name>
30.           <Placemark>
31.             <ExtendedData>
32.               <Data name="trackName">
33.                 <value>219019073</value>
34.               </Data>
35.               <Data name="timestamp">
36.                 <value>2016-02-25T03:15:42</value>
37.               </Data>
38.               <Data name="longitude">
39.                 <value>-9.75602</value>
40.               </Data>
41.               <Data name="latitude">
42.                 <value>42.90977</value>
43.               </Data>
44.               <Data name="speed">
45.                 <value>5.9155999999999995</value>
46.               </Data>
47.               <Data name="anomaly_score">
48.                 <value>0.625</value>
49.               </Data>
```

```
50.                 <Data name="new_sequence">
51.                   <value>1</value>
52.                 </Data>
53.               </ExtendedData>
54.               <styleUrl>#red</styleUrl>
55.               <Point>
56.                 <coordinates>-9.75602,42.90977</coordinates>
57.               </Point>
58.             </Placemark>
59.           </Folder>
60.         </Document>
61.       </kml>
```